

---

**Educational Services**

**digital™**

**VMS Internals II:  
Memory Management, I/O, and  
Advanced Topics  
Source Listings Book  
EY-9769E-DA-0002**

**October 1989**

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

Copyright ©1989 by Digital Equipment Corporation

All Rights Reserved.  
Printed in U.S.A.

The following are trademarks of Digital Equipment Corporation:

DEC  
DEC/CMS  
DEC/MMS  
DECsystem-10  
DECSYSTEM-  
20  
DECUS  
DECwriter  
DIBOL

EduSystem  
IAS  
MASSBUS  
PDP  
PDT  
RSTS  
RSX  
TK50

UNIBUS  
VAX  
VAXcluster  
VAX DOCUMENT  
VMS  
VT

**digital™**

## CONTENTS

|                                       |     |
|---------------------------------------|-----|
| 1 PAGEFAULT.LIS .....                 | 1   |
| 2 WRTMFYPAG.LIS .....                 | 62  |
| 3 SYSQIOREQ.LIS .....                 | 99  |
| 4 SYSIMGACT.LIS .....                 | 129 |
| 5 CHECK_VERSION.LIS .....             | 204 |
| 6 PERMANENT_DEVICE_DATABASE.LIS ..... | 212 |
| 7 LDAT.LIS .....                      | 234 |
| 8 MMDAT.LIS .....                     | 256 |
| 9 DOINIT.LIS .....                    | 262 |
| 10 PTALLOC.LIS .....                  | 277 |
| 11 SMPROUT.LIS .....                  | 287 |
| 12 SPINLOCKS.LIS .....                | 328 |





# 1 PAGEFAULT.LIS

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 0

## Table of contents

|      |      |   |            |
|------|------|---|------------|
| (1)  | 36   | HISTORY   | ; DETAILED |
| (2)  | 195  | DECLARATIONS  |            |
| (3)  | 280  | Initialization Routine                                  |            |
| (4)  | 297  | PAGE FAULT HANDLER                                      |            |
| (5)  | 352  | SYSTEM PAGE FAULT, ESTABLISH PAGE TYPE                  |            |
| (6)  | 417  | EXCEPTION ENTRY POINT - PAGE TYPE DISPATCHER            |            |
| (7)  | 547  | PAGE FILE, SECTION TABLE INDEX, OR GLOBAL PAGE          |            |
| (8)  | 622  | PAGE NOT RESIDENT, QUEUE A READ REQUEST                 |            |
| (9)  | 1102 | FORM A CLUSTER OF PAGES TO READ                         |            |
| (10) | 1270 | DEMAND ZERO PAGE  |            |
| (11) | 1381 | FREE, MODIFIED, OR BAD PAGE LIST, RELEASE PENDING       |            |
| (12) | 1538 | SCANDADPT - SCAN A DEAD PAGE TABLE FOR TRANSITION PAGES |            |
| (13) | 1650 | WSLEPFN - FETCH PFN FROM WORKING SET LIST ENTRY         |            |
| (14) | 1701 | FRWSLE - FREE A WORKING SET LIST ENTRY                  |            |
| (16) | 1996 | DELWSLEX - DELETE WORKING SET LIST ENTRY BY INDEX       |            |
| (18) | 2081 | ININWPFN - ALLOCATE AND INIT A NEW PFN                  |            |
| (20) | 2151 | MAKEWSLE - MAKE A WORKING SET LIST ENTRY                |            |
| (22) | 2251 | LOCKPGTB - LOCK PAGE TABLE                              |            |
| (23) | 2305 | INCPTRF - INCREMENT PAGE TABLE REFERENCE COUNT          |            |
| (25) | 2385 | DECPTRF - DECREMENT PAGE TABLE REFERENCE COUNT          |            |
| (27) | 2467 | DECPHDREF - DECREMENT PROCESS HEADER REFERENCE COUNT    |            |
| (28) | 2539 | INIBLDPKT - INIT FOR CALLING BUILDPKT                   |            |
| (30) | 2617 | MMG\$SWITCH_PRCPGFL - Switch process pagefile           |            |

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 1  
X-39 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (1)

```
1 .TITLE PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLER
2 .IDENT 'X-39'
3 ;
4 ;*****
5 ;*
6 ;* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
7 ;* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
8 ;* ALL RIGHTS RESERVED.
9 ;*
10 ;* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
11 ;* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
12 ;* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
13 ;* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
14 ;* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
15 ;* TRANSFERRED.
16 ;*
17 ;* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
18 ;* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
19 ;* CORPORATION.
20 ;*
21 ;* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
22 ;* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
23 ;*
24 ;*
25 ;*****
26
27 ;++
28 ; FACILITY: EXECUTIVE, TRANSLATION NOT VALID EXCEPTION HANDLER
29 ;
30 ; ABSTRACT:
31 ;
32 ; ENVIRONMENT:
33 ;
34 ;--
35 ;
36 .SBTTL HISTORY ; DETAILED
37 ;
38 ; AUTHOR: PETER H. LIPMAN , CREATION DATE: 14-SEP-76
39 ;
40 ; MODIFIED BY:
41 ;
42 ; X-39 SSA0013 Stan Amway 14-Sep-1988
43 ; Count system page fault I/Os.
44 ;
45 ; X-38 SSA0012 Stan Amway 1-Jul-1988
46 ; Remove debugging code.
47 ;
48 ; X-37 SSA0011 Stan Amway 21-Jun-1988
49 ; Collect statistics for CRF page faults.
50 ;
51 ; X-36 SSA0010 Stan Amway 12-Apr-1988
52 ; In SCANDEADPT, call MMG$PURGEMPL to selectively
53 ; flush modified page list.
54 ;
55 ; In MMG$FREWSLE, always use current process PCB address
56 ; when checking for owned mutexes. On entry to the routine,
57 ; R4 may contain the system PCB address.
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 2

X-39 HISTORY ; DETAILED 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (1)

|       |      |         |                  |  |
|-------|------|---------|------------------|--|
| 58 ;  |      |         |                  |  |
| 59 ;  | X-35 | WMC0035 | Wayne Cardoza    | 22-Feb-1988  |
| 60 ;  |      |         |                  | Use longword instructions to modify invalid global slave PTEs. |
| 61 ;  |      |         |                  | INSV lets CI see bad PTEs.                                     |
| 62 ;  |      |         |                  |  |
| 63 ;  | X-34 | SSA0009 | Stan Amway       | 1-Feb-1988   |
| 64 ;  |      |         |                  | Re-instate full MPL flush removed in X-30, but still           |
| 65 ;  |      |         |                  | place process in MPBUSY (rather than MPLEEMPTY) state.         |
| 66 ;  |      |         |                  |  |
| 67 ;  | X-33 | SJF     | Stu Farnham      | 21-Jan-1988  |
| 68 ;  |      |         |                  | Close window in which a modify bit set by a write by           |
| 69 ;  |      |         |                  | one CPU can be lost if another CPU is concurrently             |
| 70 ;  |      |         |                  | replacing the same page.                                       |
| 71 ;  |      |         |                  | Also:  |
| 72 ;  |      | SSA0008 | Stan Amway       | 18-Jan-1988  |
| 73 ;  |      |         |                  | Prevent processes from being placed into a permanent           |
| 74 ;  |      |         |                  | PFW wait state due to an intrinsic timing window.              |
| 75 ;  |      |         |                  |  |
| 76 ;  | X-32 | SSA0007 | Stan Amway       | 17-Nov-1987  |
| 77 ;  |      |         |                  | Use page count specified by MMG\$GL_RSRVPAGCNT when            |
| 78 ;  |      |         |                  | reserving process page file pages in MMG\$SWITCH_PRCPGFL.      |
| 79 ;  |      |         |                  |  |
| 80 ;  | X-31 | SSA0006 | Stan Amway       | 6-Oct-1987   |
| 81 ;  |      |         |                  | Check for installed page file(s) before placing                |
| 82 ;  |      |         |                  | process in MPWBUSY wait state.                                 |
| 83 ;  |      |         |                  |  |
| 84 ;  | X-30 | SSA0005 | Stan Amway       | 8-Sep-1987   |
| 85 ;  |      |         |                  | In SCANDEADPT, move pages on modified list to the              |
| 86 ;  |      |         |                  | beginning of the list, flush at most 128 pages, and            |
| 87 ;  |      |         |                  | place process in MPWBUSY state instead of MPLEEMPTY.           |
| 88 ;  |      |         |                  | These changes are meant to avoid (and in practice,             |
| 89 ;  |      |         |                  | eliminate) possible deadlocks.                                 |
| 90 ;  |      |         |                  |  |
| 91 ;  |      |         |                  | Check for process holding mutex in MMG\$FREWSLE.               |
| 92 ;  |      |         |                  |  |
| 93 ;  | X-29 | SSA0004 | Stan Amway       | 14-Aug-1987  |
| 94 ;  |      |         |                  | Swap file allocation changes.                                  |
| 95 ;  |      |         |                  |  |
| 96 ;  |      |         |                  | Remove DEAD code in demand-zero page fault path.               |
| 97 ;  |      |         |                  |  |
| 98 ;  | X-28 | SJF     | Stu Farnham      | 13-Aug-1987  |
| 99 ;  |      |         |                  | Fix DZRO faults.   |
| 100 ; |      | SF      | Steve Fiorelli   |  |
| 101 ; |      |         |                  | Fix errors in X-27   |
| 102 ; |      |         |                  |  |
| 103 ; | X-27 | SF00027 | Stephen Fiorelli | 4-Aug-1987   |
| 104 ; |      |         |                  | Large working set support.                                     |
| 105 ; |      |         |                  |  |
| 106 ; | X-26 | SJF     | Stu Farnham      | 21-July-1987   |
| 107 ; |      |         |                  | Make TBIS comply with SRM rev H.                               |
| 108 ; |      |         |                  |  |
| 109 ; | X-24 | RNG5024 | Rod Gamache      | 13-Mar-1987  |
| 110 ; |      |         |                  | Put back synchronization changed in last edit. Add 1           |
| 111 ; |      |         |                  | more performance optimization.                                 |
| 112 ; |      |         |                  |  |
| 113 ; | X-23 | RNG5023 | Rod Gamache      | 12-Mar-1987  |
| 114 ; |      |         |                  | Add some performance optimizations.                            |

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 3

X-39 HISTORY ; DETAILED 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (1)

115 ;  
116 ;       X-22     SF04001           Stephen Fiorelli           09-Mar-1987  
117 ;           Access syspcb through pointer. Use standard name  
118 ;           for system\_primitives\_data.  
119 ;  
120 ;       X-21     WCT0032           Ward C. Travis           27-Feb-1987  
121 ;           Update remaining old lookaside listhead references  
122 ;           to reflect that they are now interlocked queues.  
123 ;  
124 ;       X-20     SSA0003           Stan Amway           23-Feb-1987  
125 ;           Correctly assign backing store in clustering loop.  
126 ;           Correctly return process page file backing store  
127 ;           in logic that handles concurrent zeroing of a  
128 ;           global, demand zero page.  
129 ;  
130 ;       X-19     SSA0002           Stan Amway           11-Nov-1986  
131 ;           -18     Add support for multiple pagefiles per process.  
132 ;  
133 ;       X-17     RNG0017           Rod N. Gamache           28-Oct-1986  
134 ;           Fix branch errors.  
135 ;  
136 ;       X-16     RNG0016           Rod N. Gamache           29-Sep-1986  
137 ;           Release the MMG spinlock before calling EXE\$BUILDPKTR.  
138 ;           The WCB is protected separately by a lock of FILSYS  
139 ;           in IOC\$MAPVBLK.  
140 ;  
141 ;       X-15     SSA0001           Stan Amway           12-Sep-1986  
142 ;           In MMG\$FREWSLE, when checking the modified page list  
143 ;           count, use an adaptive threshold based on whether  
144 ;           the modified page writer is active. If not active,  
145 ;           use MPW\$GL\_WAITLIM, else use MPW\$GL\_LOWAITLIM. Since  
146 ;           the page write I/O completion routine uses the latter  
147 ;           value to determine when to reawaken proceses in the  
148 ;           MPWBUSY wait state, this change prevents processes  
149 ;           from becoming permanently stuck in the wait state due  
150 ;           to continual modified page generation.  
151 ;  
152 ;       X-14     MSH0276           Michael S. Harvey       15-Aug-1986  
153 ;           Don't try to release MMG more than once on a pagefault  
154 ;           exit path.  
155 ;  
156 ;       X-13     MSH0274           Michael S. Harvey       10-Aug-1986  
157 ;           Hold MMG across call to EXE\$BUILDPKTR to ensure, for  
158 ;           now anyway, that the WCB remains protected. This  
159 ;           necessitates moving the MMG spinlock down in rank  
160 ;           to be less than IOLOCK8 which the routine being  
161 ;           called will attempt to acquire.  
162 ;  
163 ;       X-12     WMC0006           Wayne Cardoza           28-Jul-1986  
164 ;           Get rid of re-executeable switch.  
165 ;  
166 ;       X-11     RNG0011           Rod Gamache           25-Jul-1986  
167 ;           Change most of the synchronization around - release  
168 ;           MMG spinlock as early as possible, acquire SCHED as late  
169 ;           as possible and don't hold onto MMG while doing I/O thru  
170 ;           EXE\$BUILDPKTR.  
171 ;

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 4

X-39 HISTORY ; DETAILED 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (1)

|       |         |         |  |             |
|-------|---------|---------|--|-------------|
| 172 ; | X-10    | WMC0005 | Wayne Cardoza  | 21-Jul-1986 |
| 173 ; |         |         | Change many JSBs to BSBWs.                               |             |
| 174 ; |         |         |  |             |
| 175 ; | X-9     | SJF     | Stu Farnham  | 7-Jul-1986  |
| 176 ; |         |         | Manually merge X-4 and X-2H3 to form new mainline.       |             |
| 177 ; |         |         |  |             |
| 178 ; | X-7     | SJF     | Stu Farnham  | 5-Jul-1986  |
| 179 ; |         |         | Fix more merge errors. ***WARNING*** The merge           |             |
| 180 ; |         |         | SERIOUSLY SCARBELD this file.                            |             |
| 181 ; |         |         |  |             |
| 182 ; | X-4     | WMC0004 | Wayne Cardoza  | 18-Jun-1986 |
| 183 ; |         |         | Initialization routine should return status.             |             |
| 184 ; |         |         |  |             |
| 185 ; | X-3     | WMC0003 | Wayne Cardoza  | 20-May-1986 |
| 186 ; |         |         | Fix initialization macro.                                |             |
| 187 ; |         |         |  |             |
| 188 ; | X-1D2   | WMC0001 | Wayne Cardoza  | 21-Jan-1986 |
| 189 ; |         |         | Add initialization routine for SCB.                      |             |
| 190 ; |         |         |  |             |
| 191 ; | V04-001 | TCM0002 | Trudy C. Matthews  | 29-Mar-1985 |
| 192 ; |         |         | Move global data cells in psect \$\$\$210 to SYSDATA.MAR |             |
| 193 ; |         |         |  |             |

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 5

X-39 DECLARATIONS 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (2)

```

195          .SBTTL  DECLARATIONS
196 ;
197 ; INCLUDE FILES:
198 ;
199          $CADEF          ;CONDITIONAL ASSEMBLY DEFINITIONS
200          $CPUDEF        ;DEFINE PER-CPU DATA BLOCK OFFSETS
201          $IPLDEF        ;PROCESSOR PRIORITY LEVEL DEFINITIONS
202          $IREPDEF       ;I/O REQUEST PACKET DEFINITIONS
203          $MPWDEF        ; Define MPW structures and constants
204          $PCBDEF        ;PROCESS CONTROL BLOCK DEFINITIONS
205          $PHDEF         ;PROCESS HEADER DEFINITIONS
206          $PRDEF         ;PROCESSOR REGISTER DEFINITIONS
207          $PFLDEF        ;PAGE FILE CONTROL BLOCK DEFINITIONS
208          $PFNDEF        ;PFN DATA BASE DEFINITIONS
209          $PRTDEF        ;PROTECTION FIELD DEFINITIONS
210          $PSLDEF        ;PROCESSOR STATUS LONG WORD DEFINITIONS
211          $PTEDEF        ;PAGE TABLE ENTRY OFFSETS
212          $RSNDEF        ;RESOURCE NAME DEFINITIONS
213          $SECDEF        ;SECTION TABLE DEFINITIONS
214          $SSDEF         ;SYSTEM STATUS DEFINITIONS
215          $$SYSTEM_PRIM_DATADEF ;SYSTEM PRIMITIVES LOCAL DATA
216          $VADEF         ;VIRTUAL ADDRESS VIELDS
217          $WQHDEF       ;WAIT QUEUE HEADER DEFINITIONS
218          $WSLDEF       ;WORKING SET LIST DEFINITIONS
219 ;
220 ; EXTERNAL SYMBOLS:
221 ;
222 ;
223 ;
224 ; MACROS:
225 ;
226 ;
227 ;
228 ; EQUATED SYMBOLS:
229 ;
230 ;
231          $OFFSET <4*6>, POSITIVE, <-
232          FLTCTL, -          ;OFFSET TO FAULT CONTROL BITS
233          FLTVA, -          ;OFFSET TO FAULT VIRTUAL ADDRESS
234          FLTPC, -          ;OFFSET TO FAULT PC
235          FLTPSL, -        ;OFFSET TO FAULT PSL
236          >
          FLTCTL:
          FLTVA:
          FLTPC:
          FLTPSL:
237 ;
238          $VFIELD  PGF, 0, <-          ;DEFINE PAGE FAULT CONTROL BITS
239          LENVIO, -          ;LENGTH VIOLATION
240          PGTBFLT, -        ;PAGE TABLE FAULT
241          WRTACC, -        ;REFERENCE WAS WRITE OR MODIFY
242          >
243 ;
244 ; OFFSETS INTO I/O PACKET WHILE BEING USED AS SCRATCH STORAGE FOR CLUSTERING
245 ;
246          $OFFSET 0, POSITIVE, <-
247          PTEDAT, -          ;MASTER PTE CONTENTS

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 6

X-39 DECLARATIONS 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (2)

|     |                         |   |
|-----|-------------------------|---|
| 248 | SVAPTE, -               | ;MASTER PTE ADDRESS                     |
| 249 | SIZE_TYPE, -            | ;RESERVED FOR SIZE AND TYPE             |
| 250 | VA, -                   | ;VIRTUAL ADDRESS                        |
| 251 | AST, -                  | ;AST INFO                               |
| 252 | ASTPRM, -               | ; AND PARAMETER                         |
| 253 | GPTX, -                 | ;PROCESS PTE CONTENT FOR GLOBAL PAGE    |
| 254 | GPTX PTE, -             | ;PROCESS PTE ADR FOR GLOBAL PAGE        |
| 255 | <CLUSTER, 1>, -         | ;DESIRED CLUSTER SIZE                   |
| 256 | <COUNT, 1>, -           | ;CURRENT COUNT OF PAGES                 |
| 257 | <STATE, 1>, -           | ;SAVED PFN STATE BYTE                   |
| 258 | <PRI, 1>, -             | ;PRIORITY OF I/O TRANSFER               |
| 259 | BAK, -                  | ;PFN BACKING STORE ADDRESS              |
| 260 | INC1, -                 | ;+ OR - 1                               |
| 261 | INC4, -                 | ;+ OR - 4                               |
| 262 | INC512, -               | ;+ OR - 512                             |
| 263 | VCN, -                  | ;VIRTUAL BLOCK NUMBER                   |
| 264 | WINDOW, -               | ;WINDOW CONTROL BLOCK ADR               |
| 265 | FP_SAV, -               | ;SAVED FP                               |
| 266 | <PCB_SAV, 8>, -         | ;SAVED PCB, PHD ADDRESS                 |
| 267 | PHVREFCADR, -           | ;PROCESS HEADER REFERENCE COUNT ADDRESS |
| 268 | <CLU_SCRATCH_SIZ, 0>, - | ;SIZE OF THIS SCRATCH AREA              |
| 269 | >                       |   |

PTEDAT:  
 SVAPTE:  
 SIZE\_TYPE:  
 VA:  
 AST:  
 ASTPRM:  
 GPTX:  
 GPTX PTE:  
 CLUSTER:  
 COUNT:  
 STATE:  
 PRI:  
 BAK:  
 INC1:  
 INC4:  
 INC512:  
 VCN:  
 WINDOW:  
 FP\_SAV:  
 PCB\_SAV:  
 PHVREFCADR:  
 CLU\_SCRATCH\_SIZ:

270  
 271       ASSUME CLU\_SCRATCH\_SIZ LE IRP\$C\_LENGTH  
 272       ASSUME SIZE\_TYPE EQ IRP\$W\_SIZE  
 273       ASSUME AST EQ IRP\$L\_AST  
 274       ASSUME ASTPRM EQ IRP\$L\_ASTPRM  
 275       ASSUME PRI EQ IRP\$B\_PRI  
 276 ;  
 277 ; OWN STORAGE:  
 278 ;

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 7

X-39 Initialization Routine 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (3)

```
280          .SBTTL  Initialization Routine
281 ;
282 ; Initialize the SCB with the pagefault handler address.
283 ;
284 ;
285          DECLARE_PSECT EXEC$INIT_CODE
286
287 INITIALIZATION_ROUTINE -
288          PAGEFAULT_INIT
289
290 PAGEFAULT_INIT:
291          MOVL   G^EXE$GL_SCB,R0                ; SCB address
292          MOVAB  W^MMG$PAGEFAULT,^X24(R0)        ; Fill in TNV vector
293          MOVL   #SS$ _NORMAL,R0
294          RSB
295
```



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 8

X-39 PAGE FAULT HANDLER 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (4)

```
297         .SBTTL PAGE FAULT HANDLER
298
299 ;++
300 ; FUNCTIONAL DESCRIPTION:
301 ;
302 ; THIS MODULE CONTAINS THE PAGEFAULT HANDLER. IT IS ENTERED VIA A
303 ; TRANSLATION-NOT-VALID FAULT. AT THE TIME OF A FAULT, THE KERNEL
304 ; STACK CONTAINS THE FOLLOWING INFORMATION:
305 ;
306 ;      +-----+
307 ;      ! REASON MASK ! --> BIT 0 - ALWAYS 0 FOR
308 ;      +-----+                                TRANS-NOT-VALID FAULTS
309 ;      ! INVALID VIRTUAL ADDRESS ! BIT 1 - 0 INDIC VIRT ADR NOT VALID
310 ;      +-----+                                1 INDIC ASSOC PTE NOT VALID
311 ;      ! PC OF FAULTING INSTRUCTION ! BIT 2 - 0 INDIC READ ACCESS
312 ;      +-----+                                1 INDIC MODIFY/WRITE ACCESS
313 ;      ! PSL OF FAULTING INSTRUCTION !
314 ;      +-----+
315 ;
316 ; CALLING SEQUENCE:
317 ;     NONE
318 ;
319 ;
320 ; INPUT PARAMETERS:
321 ;     NONE
322 ;
323 ; IMPLICIT INPUTS:
324 ;     NONE
325 ;
326 ; OUTPUT PARAMETERS:
327 ;     NONE
328 ;
329 ; IMPLICIT OUTPUTS:
330 ;     NONE
331 ;
332 ; COMPLETION CODES:
333 ;     NONE
334 ;
335 ; SIDE EFFECTS:
336 ;     NONE
337 ;
338 ;--
339
340 ;
341 ; *****
342 ;
343 ; ***** THIS ENTIRE MODULE MUST BE RESIDENT *****
344 ;
345 ;     DECLARE_PSECT EXEC$NONPAGED_CODE
346 ;
347 ; *****
348 ;
349 ;     .LIST MEB
350
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 9

X-39 SYSTEM PAGE FAULT, ESTABLISH PAGE TYPE 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (5

```

352      .SBTTL  SYSTEM PAGE FAULT, ESTABLISH PAGE TYPE
353
354      .ENABL  LSB
355 ;
356 ; BAD SYSTEM PAGE - PROCESS HEADER OR PAGE TABLE PAGE FOR ANOTHER PROCESS
357 ;
358 ; IF THE PROCESS HEADER HAS JUST BEEN INSWAPPED (PHD$V_NOACCVIO IS SET),
359 ; SIMPLY DISMISS THE FAULT. IN ALL OTHER CASES, REPORT AN ACCESS VIOLATION
360 ; EXCEPTION.
361 ;
362 ; IF THE PROCESS WAS OUTSWAPPED WHILE ACCESSING ITS OWN HEADER,
363 ; DISMISSING THE EXCEPTION WILL CAUSE THE REFERENCE TO OCCUR AGAIN,
364 ; BUT THIS TIME TO THE CORRECT BALANCE SLOT.
365 ;
366 ; IF THE PROCESS WAS MAKING AN ILLEGAL REFERENCE TO THE HEADER OF ANOTHER
367 ; PROCESS, DISMISSING THE EXCEPTION WILL CAUSE THE SAME ILLEGAL REFERENCE
368 ; TO OCCUR AGAIN, BUT NOW WITH PHD$V_ACCVIO CLEAR, CAUSING AN ACCESS
369 ; VIOLATION TO BE REPORTED.
370
371 BADSYSPAG:
372      BBSC      #PHD$V_NOACCVIO,PHD$W_FLAGS(R5),10$ ;BRANCH IF HEADER JUST INSWAPPED
373      BRW      ACVIOLAT ;FAKE AN ACCESS VIOLATION
374
375 10$:      BRW      PGFCOMPLETE ;SIMPLY DISMISS THE PAGE FAULT
376
377 ;
378 ; SEE IF PAGE IS A GLOBAL PAGE TABLE PAGE, OTHERWISE ERROR
379 ;
380 GPGTBL:
381      CMPW     R0,G^SGN$GL_BALSETCT ;SYSTEM BALANCE SET SLOT?
382      BLSS     BADSYSPAG ;BRANCH IF NOT
383      CML     R2,G^MMG$GL_MAXGPT ;LEGAL GPT ADDRESS
384      BGEQU    BADSYSPAG
385      MOV     #WSL$C_GPGTBL,R2 ;PAGE IS GLOBAL PAGE TABLE
386      BRB     50$
387 ;
388 ; PAGE IS NOT SYSTEM PAGE TYPE, COULD BE GLOBAL PAGE TABLE, PROCESS PAGE TABLE
389 ; OR PROCESS HEADER PAGE
390 ;
391 NOTSYSTEM:
392      ASHL     #-9,R0,R0 ;SCALE VA DIFFERENCE TO PAGE NUMBER
393      MOVL     PCB$L_PHD(R4),R5 ;ADDRESS OF PROCESS HEADER
394      DIVL     G^SWP$GL_BSLLOTSZ,R0 ;PROCESS HEADER INDEX
395      CMPW     R0,PHD$W_PHVINDEX(R5) ;THIS PROCESS' HEADER?
396      BNEQ     GPGTBL ;BRANCH IF NOT, MAYBE GLOBAL PAGE TABLE
397      MOV     #WSL$C_PPGTBL,R2 ;ASSUME PROCESS PAGE TABLE
398      BRB     70$
399 ;
400 ; PAGE FAULT FOR SYSTEM SPACE VIRTUAL ADDRESS
401 ; R2 = FAULT VA, LOW BITS CLEARED
402 ; R4 = PROCESS PCB ADDRESS
403 ;
404 SYSTEMSPACE:
405      EXTZV    #VA$V_VPN,#VA$S_VPN,R2,R3 ;PAGE NUMBER IN SYSTEM SPACE
406      SUBL3    G^SWP$GL_BALBASE,R2,R0 ;ABOVE BASE OF BALANCE SET SLOTS?
407      BGEQ     NOTSYSTEM ;BRANCH IF NOT SYSTEM PAGE TYPE
408 40$:      MOV     #WSL$C_SYSTEM,R2 ;SYSTEM PAGE

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 10  
X-39 SYSTEM PAGE FAULT, ESTABLISH PAGE TYPE 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (5  
409 50\$:    MOVL    G^MMG\$AR\_SYSPCB,R4           ;ADDRESS OF SYSTEM PCB  
410            MOVL    PCB\$P\_L\_PHD(R4),R5       ;ADDRESS OF SYSTEM PROCESS HEADER  
411 70\$:    MOVAL    @W^MMG\$GL\_SPTBASE[R3],R3 ;ADDRESS OF PAGE TABLE ENTRY  
412            BRB     GETPAGELOC  
413  
414            .DSABL   LSB  
415

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 11  
X-39 EXECPTION ENTRY POINT - PAGE TYPE DISPA 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```

417      .SBTTL  EXECEPTION ENTRY POINT - PAGE TYPE DISPATCHER
418
419 ;
420 ; PAGE FAULT MONITORING ENABLED FOR THIS PROCESS
421 ;
422 PGFMONITOR:
423      JSB      G^PFM$MON          ;CALL THE RECORDING ROUTINE
424      BRB      PGFMONITOR1       ;RETURN IN LINE
425
426 IPLHI:  BUG_CHECK PGFIPLHI,FATAL      ;IPL TOO HIGH FOR PAGE FAULT
          .WORD  ^XFEFF
          .IIF  IDN <FATAL>,<FATAL> , .WORD      BUG$ PGFIPLHI!4

427
428 ;
429 ; THIS IS A PROCESS PAGE TABLE FAULT
430 ;
431 PPGTBL:
432      BICL3   $VA$M_BYTE,R3,R2      ;R2 = FAULT VA
433      BRB     SYSTEMSPACE
434
435      .ALIGN  LONG
436      UNIVERSAL_SYMBOL      MMG$PAGEFAULT
MMG$PAGEFAULT::
437 ;MMG$PAGEFAULT::
438      PUSHL   R5                  ;SAVE R5
439      PUSHL   R4                  ;SAVE R4
440      CMPZV   $PSL$V_IPL,$PSL$$_IPL,<12+8>(SP), $IPL$_ASTDEL ;CHECK FAULT IPL
441      BGTR    IPLHI              ;BRANCH IF IPL IS TOO HIGH
442      PUSHL   R3                  ;SAVE R3
443      PUSHL   R2                  ;SAVE R2
444      PUSHL   R1                  ;SAVE R1
445      PUSHL   R0                  ;SAVE R0
446      LOCK    LOCKNAME=MMG,-      ;LOCK MMG DATABASE
447      PRESERVE=NO                ;DON'T PRESERVE R0

      .SAVE    LOCAL_BLOCK
      .PSECT   $ABS$,ABS
      .=0
      .RESTORE
      .SAVE    LOCAL_BLOCK
      .PSECT   $ABS$,ABS
      .=0
      .RESTORE
      BLBC     G^SMP$GL_FLAGS,30002$
      MOVZBL   S^$SPL$C_MMG,R0
      JSB      G^SMP$ACQUIRE
      BRB      30003$

30002$:
          MTPR    S^$IPL$_MMG,S^$PR$_IPL

30003$:
448      MOVL   G^CTL$GL_PCB,R4      ;R4 = ADDRESS OF PCB
449      BICL3   $VA$M_BYTE,FLTVA(SP),R2 ;R2 = VA OF FAULT (LOW BITS CLEAR)
450      BLSS   SYSTEMSPACE          ;BRANCH IF SYSTEM SPACE ADDRESS
451      MOVL   PCB$L_PHD(R4),R5     ;R5 = ADDRESS OF HEADER
452      ASSUME  PHD$V_PFMFLG EQ 0
453      BLBS   PHD$W_FLAGS(R5),PGFMONITOR ;BRANCH IF PAGE FAULT MONITORING
454 PGFMONITOR1:
455      EXTZV   $VA$V_VPN,$VA$$_VPN,R2,R3 ;VIRTUAL PAGE IN P0 OR P1 SPACE

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 12  
X-39 EXECEPTION ENTRY POINT - PAGE TYPE DISPA 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```

456         BBC      #VA$V_P1,R2,POADDR      ;BRANCH IF P0 SPACE
457         MOVAL    @PHD$L_P1BR(R5)[R3],R3  ;GET SYS VIRT ADR OF PTE FOR P1 SPACE
458         BRB      GETPAGELOC
459
460 VALID:   BRW     PGFCOMPLETE              ;IF VALID, JUST EXIT
461
462 DZRO_PTE_0:
463         BRW     DZRO_PTE
464
465 RSRWAIT_3:
466         ADDL    #3*4,SP                    ;CLEAN OFF 3 LONG WORDS
467         BRW     RESOURCEWAIT              ;AND GO WAIT FOR A RESOURCE
468
469 POADDR:  MOVAL    @PHD$L_POBR(R5)[R3],R3  ;GET SYS VIRT ADR OF PTE FOR P0 SPACE
470 ;
471 ; R2 = VA (LOW BITS = PAGTYP)
472 ; R3 = SVAPTE
473 ;
474 GETPAGELOC:
475         EXTZV   #VA$V_VPN,#VA$$_VPN,R3,R0 ;INDEX TO SPT ENTRY
476         MOVAL    @W^MMG$GL_SPTBASE[R0],R1 ;ADDRESS OF SPTE FOR PAGE TABLE
477         TSTL    (R1)                       ;IS SPTE VALID?
478         BGEQ    PPGTBL                      ;BRANCH IF NOT, FAULT IT
479         BICL3   #^C<PTE$M_VALID            -;CHECK VALID BIT
480         ! PTE$M_TYP1 ! PTE$M_TYP0 -;GET PTE TYPE BITS
481         ! PTE$M_BAKX>, (R3),R0            ;AND PFN or PGFLVBN/PRCPGFLX BITS TO R0
482         BLSS    VALID                      ;BRANCH IF VALID
483 ;
484 ; R0 = TYP1 ! TYP0 ! BAKX, VALID IS KNOWN TO BE OFF AT THIS POINT
485 ; R1 = SPT ENTRY ADDRESS FOR PAGE TABLE PAGE
486 ; R2 = VA (LOW BITS = PAGTYP)
487 ; R3 = SVAPTE
488 ; R4 = PCB ADDRESS
489 ; R5 = PHD ADDRESS
490 ;
491         INCL    G^PMS$GL_FAULTS            ;COUNT ALL THE PAGE FAULTS
492         INCL    PHD$L_PAGEFLTS(R5)        ;COUNT PROCESS' PAGE FAULTS
493         PUSHL   R3                          ;SAVE NEEDED VOLATILE REGISTERS
494         PUSHL   R2
495         PUSHL   R1
496         BSBW    MMG$FREWSLE                ;FREE A WORKING SET LIST ENTRY
497         BLBC    R0,RSRCWAIT_3              ;BRANCH IF HAVE TO WAIT
498         MOVL    8(SP),R3                   ;SVAPTE
499 ;
500 ; MUST RECHECK VALIDITY OF PAGE TABLE PAGE, SINCE FREWSLE MIGHT HAVE DISCARDED IT
501 ;
502         TSTL    @(SP)+                      ;IS PAGE TABLE PAGE VALID?
503         BGEQ    PPGTBL_2                  ;NO, GO FAULT THE PAGE TABLE
504 ;
505 ; ***** ALL POINTS DISPATCHED TO FROM HERE MUST REMEMBER THAT
506 ; ***** 0(SP) = VA (LOW BITS = PAGTYP), 4(SP) = SVAPTE
507 ;
508 ;
509 ; MUST FETCH PAGE TABLE ENTRY CONTENTS AGAIN SINCE DEAD PAGE TABLE SCAN
510 ; IN FREWSLE MIGHT HAVE DISCARDED THE PAGE. NOTE THAT VALID IS KNOWN 0.
511 ;
512         BICL3   #^C<PTE$M_VALID ! -      ;FETCH VALID BIT

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 13  
X-39 EXECPTION ENTRY POINT - PAGE TYPE DISPA 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```

513          PTE$M_TYP1 ! PTE$M_TYPO !- ;PTE TYPE BITS
514          PTE$M_BAKX>, (R3), R0      ;AND PGFLVBN/PRCPGFLX, GPTX, SECX, PFN
515          BEQL      DZRO_PTE_0      ;BRANCH IF PAGE IS DEMAND ZERO
516 ;
517 ; FORM R1 = 4 BIT SIGN EXTENDED VIELD LOW BIT = TYPO, SIGN = TYP1
518 ;
519          EXTV      #PTE$V_TYPO, #<PTE$V_TYP1+1-PTE$V_TYPO>, R0, R1
520          BNEQ      NOTTRANSITION    ;BRANCH IF NOT TRANSITION PAGE
521 ;
522 ; THIS IS A PAGE IN TRANSITION
523 ; R0 = PFN, R2 = VA (LOW BITS = PAGTYP), R3 = SVAPTE
524 ;
525 TRANSITION:
526          EXTZV     #PFN$V_LOC, #PFN$S_LOC, @W^PFN$AB_STATE[R0], R2 ;PAGE LOCATION
527
528          .IF      GT, CA$_MEASURE
529          INCL     G^PMS$AL_TRANSFLT[R2] ;COUNT VARIOUS TRANSITION FAULTS
530          .ENDC
531
532          CASE     R2, <-
533                  PFNLIST, -           ;ON THE FREE PAGE LIST
534                  PFNLIST, -           ;ON THE MODIFIED PAGE LIST
535                  PFNLIST, -           ;ON THE BAD PAGE LIST
536                  RELEASEPEND, -      ;RELEASE PENDING
537                  READERR, -          ;PAGE READ ERROR
538                  WRITEINPROG, -      ;PAGE WRITE IN PROGRESS
539                  READINPROG-         ;PAGE READ IN PROGRESS
540          >
          CASEW     R2, #0, S^#<<30007$-30006$>/2>-1
30006$:
          .SIGNED_WORD   PFNLIST-30006$
          .SIGNED_WORD   PFNLIST-30006$
          .SIGNED_WORD   PFNLIST-30006$
          .SIGNED_WORD   RELEASEPEND-30006$
          .SIGNED_WORD   READERR-30006$
          .SIGNED_WORD   WRITEINPROG-30006$
          .SIGNED_WORD   READINPROG-30006$
30007$:
541 LOCBAD: BUG_CHECK PGFLOCBAD, FATAL      ;BAD PAGE LOCATION FIELD
          .WORD      ^XFEFF
          .IIF IDN <FATAL>, <FATAL> , .WORD      BUG$ PGFLOCBAD!4
542
543 PPGTBL_2:
544          ADDL     #2*4, SP              ;CLEAN OFF 2 LONG WORDS
545          BRW      PPGTBL                ;FAULT A PROCESS PAGE TABLE

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 14  
X-39 PAGE FILE, SECTION TABLE INDEX, OR GLOBAL 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```

547      .SBTTL PAGE FILE, SECTION TABLE INDEX, OR GLOBAL PAGE
548 ;
549 ; GLOBAL PAGE, MASTER PTE VALID.
550 ; R0 = MASTER PTE CONTENTS (VALID, MODIFY AND PFN BITS)
551 ; 0(SP) = PROCESS VA (LOW BITS = PAGTYP), 4(SP) = SLAVE PTE ADDRESS
552 ;
553 GBLVALID:
554     INCL     G^PMS$GL_GVALID           ;UPDATE GLOBAL VALID COUNTER
555     MOVQ    (SP),R2                    ;R2=VA (LOW BITS = PAGTYP), R3=SVAPTE
556     PUSHL   RO                          ;SAVE MASTER PTE
557     ASSUME  PTE$V_PFN EQ 0
558     BICL   #^C<PTE$M_PFN>,R0          ;GET PAGE FRAME NUMBER
559     BSBW   MMG$MAKEWSLE                ;MAKE A WORKING SET LIST ENTRY
560     POPL   R1                            ;R1=MASTER PTE
561     MOVQ   (SP)+,R2                      ;R2=VA, R3=SVAPTE
562     BRW    SETSLAVEPTE                  ;SET PROCESS' PTE AND EXIT
563 ;
564 ; DEMAND ZERO GLOBAL SECTION WITH PAGE FILE BACKING STORE
565 ;
566 GBLDZRO_PGFL:
567     MOVL   #PTE$M_TYPO,R0                ;ADD A TYPO BIT TO THE MASTER PTE (0)
568     BRB   GBLDZRO                        ;GO JOIN COMMON CODE
569 ;
570 ; PAGE IS NOT A TRANSITION OR DEMAND ZERO PAGE
571 ; R0 = LOW 23 BITS OF PTE AND TYPE BITS (PAGE FILE VBN, GPTX OR STX)
572 ; R1 = RESULT OF EXTV ABOVE, CONDITION CODES SET FROM EXTV
573 ; 0(SP) = VA (LOW BITS = PAGTYP), 4(SP) = SVAPTE
574 ;
575 NOTTRANSITION:
576     BLSS   NOTGLOBAL                     ;BRANCH IF TYP1 SET, NOT A GLOBAL PAGE
577     BICL   #PTE$M_TYPO,R0                ;LEAVE JUST GLOBAL PAGE TABLE INDEX
578     MOVAL  @W^MMG$GL_GPTBASE[R0],R3      ;ADDRESS OF MASTER PTE
579     ROTL   #<32-<PTE$V_OWN-WLS$V_PAGTYP>>, (R3),R1 ;OWNER FIELD TO LOW BITS
580 ;
581 ; MASTER PTE OWNER FIELD CONTAINS THE VALUE PFN$C_GLOBAL OR PFN$C_GBLWRT
582 ;
583     BICB3  #^C<WLS$M_PAGTYP>,R1,(SP)     ;SET PAGE TYPE FIELD
584     BICL3  #^C<PTE$M_VALID ! -           ;GET THE VALID BIT
585           PTE$M_TYP1 ! PTE$M_TYPO ! - ;THE PTE TYPE BITS
586           PTE$M_BAKX>,(R3),R0          ;AND PGFLVBN/PRCPGFLX, OR STX
587     BLSS   GBLVALID                       ;BRANCH IF MASTER PTE IS VALID
588     BEQL   GBLDZRO_PGFL                   ;MASTER PTE IS DEMAND ZERO
589     EXTV   #PTE$V_TYPO,#<PTE$V_TYP1+1-PTE$V_TYPO>,R0,R1
590 ;
591 ; R1 = 0 IF TYP1 AND TYPO ARE BOTH ZERO
592 ; R1 = NEGATIVE IF TYP1 IS SET
593 ; R1<0> = 1 IF TYPO IS SET
594 ;
595     BGTR   GBLBAD                          ;BRANCH IF GLOBAL AGAIN, ERROR
596     BEQL   TRANSITION                      ;BRANCH IF GLOBAL TRANSITION
597 ;;;     BLSS   20$                          ;BRANCH IF SECTION OR PAGE FILE ADDRESS
598 ;
599 ; MASTER PAGE TABLE ENTRY IS A SECTION OR PAGE FILE ADDRESS
600 ;
601 20$:   BLBC   R1,30$                        ;BRANCH IF PAGE FILE
602     BBS    #PTE$V_DZRO,R0,GBLDZRO        ;BRANCH IF DEMAND ZERO GLOBAL SECTION
603     BBS    #PTE$V_CRF,R0,GBLCRF         ;BRANCH IF COPY ON REFERENCE

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 15  
X-39 PAGE FILE, SECTION TABLE INDEX, OR GLOBA 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```
604 ;  
605 ; GLOBAL SECTION (NOT CRF OR DZRO) OR PAGE FILE BACKING STORE ADR  
606 ;  
607 30$:   MOVL    4(SP),R0           ;SAVE SLAVE PTE ADR INDICATING GLOBAL  
608       BRB     GBLNOTRESIDENT    ;  
609 ;  
610 ; BAD MASTER PAGE TABLE ENTRY FORMAT FOR A GLOBAL PAGE  
611 ;  
612 GBLBAD: BUG_CHECK PGFGBLBAD,FATAL ;BAD MASTER PTE FORMAT FOR GLOBAL PAGE  
          .WORD   ^XFEFF  
          .IIF IDN <FATAL>,<FATAL> , .WORD   BUG$_PGFGBLBAD!4  
  
613  
614 ;  
615 ; GLOBAL COPY ON REFERENCE PAGE  
616 ;  
617 GBLCRF:  
618       CLRB    (SP)               ;SAY PAGE IS PROCESS PAGE  
619       MOVL    R3,R0              ;MASTER PTE ADDRESS  
620       BRB     GBLNOTRESIDENT    ;
```



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 16  
X-39 PAGE NOT RESIDENT, QUEUE A READ REQUEST 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```

622      .SBTTL PAGE NOT RESIDENT, QUEUE A READ REQUEST
623      .ENABL LSB
624
625 ;
626 ; MUST WAIT FOR AN I/O REQUEST PACKET
627 ;
628 IRPWAIT_3:
629      POPL R1 ;CLEAN OFF 1 LONG WORD
630      MOVZBL #RSN$_NPDYMEM,R1 ;NON PAGED DYNAMIC MEMORY RESOURCE NUMBER
631 10$: ADDL #2*4,SP ;CLEAN OFF 2 LONG WORDS
632      BRW RESOURCEWAIT ;WAIT FOR RESOURCE IN R1
633 ;
634 ; THIS PAGE READ WOULD EXCEED THIS PROCESS' DIRECT I/O QUOTA.
635 ; WAIT UNTIL SOME OF HIS OUTSTANDING I/O COMPLETES.
636 ;
637 DIOCNTWAIT_2:
638      MOVZBL #RSN$_ASTWAIT,R1 ;AST WAIT RESOURCE NUMBER
639      BRB 10$
640
641      .DSABL LSB
642
643 ;
644 ; DEMAND ZERO GLOBAL SECTION PAGE
645 ;
646 GBLDZRO:
647      MOVL (R3),R1 ;MASTER PTE CONTENTS
648      BRW DZRO_GBL_SEC
649 ;
650 ; DEMAND ZERO PROCESS SECTION PAGE
651 ;
652 DZRO_PROC_SEC:
653      BRW DZRO_PTE
654 ;
655 ; PAGE IS NOT A GLOBAL PAGE
656 ; R1<31>=TYP1, R1<0>=TYP0, R0 = TYP1 ! TYP0 ! BAKX
657 ;
658 NOTGLOBAL:
659      BLBC R1,10$ ;BRANCH IF NOT SECTION PAGE
660      BBS #PTE$V_DZRO,R0,DZRO_PROC_SEC ;BRANCH IF DEMAND ZERO PROCESS SECTION
661 10$: CLRL R0 ;INDICATE NO SLAVE PAGE TABLE ENTRY
662 ;
663 ; 0(SP) = VA (LOW BITS = PAGTYP), 4(SP) = SVAPTE (SLAVE IF GLOBAL)
664 ; R0 = MASTER PTE ADDRESS IF GLOBAL CRF
665 ; = SLAVE PTE ADDRESS IF GLOBAL NOT CRF
666 ; = 0 IF NOT GLOBAL
667 ;
668      .ENABL LSB
669 GBLNOTRESIDENT:
670      CMPB PHD$B_PGTEPFC(R5),#1 ;IF CLUSTERING PAGE TABLE PAGES
671      BLEQ 40$
672      EXTZV #VA$V_VPN,#VA$S_VPN,4(SP),R2 ;SEE IF ADJACENT PAGE TABLES
673      MOVAL @W^MMG$GL_SPTBASE[R2],R1 ;NEED TO BE FAULTED, GET SPT ENTRY ADR
674      BITL #<PTE$M_VALID ! -
675      PTE$M_TYP1 ! PTE$M_TYP0 ! -
676      PTE$M_BAKX>,-(R1) ;CHECK PREVIOUS SPT ENTRY
677      BGTR 10$ ;BRANCH IF NOT VALID, NOT DZRO
678      BITL #<PTE$M_VALID ! -

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 17

X-39 PAGE NOT RESIDENT, QUEUE A READ REQUEST 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```

679          PTE$M_TYP1 ! PTE$M_TYPO ! -
680          PTE$M_BAKX>,8(R1)          ;CHECK NEXT SPT ENTRY
681          BLEQ      40$                ;BRANCH IF IT IS VALID
682 ;
683 ; NEXT PAGE TABLE NEEDS TO BE FAULTED
684 ;
685          INCL      R2                  ;SET NEXT SPT INDEX
686          BRB       20$
687 ;
688 ; NO I/O PACKETS ON THE SIDE LIST, MUST ALLOCATE ONE FROM NON-PAGED POOL
689 ; 0(SP) = PLACE TO STORE ADDRESS OF PACKET, TOTAL OF 3 LONG WORDS ON STACK
690 ;
691 GET_IRP:
692          MOVL      R0, (SP)            ;SAVE REGISTER
693          MOVZBL   #IRP$C_LENGTH,R1    ;SIZE OF I/O PACKET
694          JSB      G^EXE$ALONONPAGED  ;ALLOCATE THE PACKET
695          BLBC     R0,IRPWAIT_3        ;BRANCH IF NONE AVAILABLE
696          MOVL     (SP),R0             ;RESTORE SAVED REGISTER
697          MOVL     R2,(SP)            ;SAVE I/O PACKET ADDRESS
698          BRB      GOT_IRP            ;REJOIN THE MAIN FLOW
699 ;
700 ; PREVIOUS PAGE TABLE NEEDS TO BE FAULTED
701 ;
702 10$:      DECL     R2                  ;SET PREVIOUS SPT INDEX
703 20$:      ROTL     #9,R2,R2          ;TURN SPT INDEX BACK INTO
704          BISL     #VA$M_SYSTEM,R2    ;SYSTEM VIRTUAL ADDRESS
705          ADDL     #2*4,SP            ;CLEAN OFF 2 LONG WORDS
706          BRW      SYSTEMSPACE       ;GO FAULT THE PAGE TABLE
707 40$:      MOVL     G^CTL$GSL_PCB,R1   ;COULD HAVE SYSTEM PCB IN R4
708          TSTW    PCB$W_DI0CNT(R1)   ;ENOUGH DIRECT I/O QUOTA FOR THIS READ?
709          ;NOTE THAT BUILDPKT WILL CHARGE THE READ
710          BLEQ    DI0CNTWAIT_2       ;BRANCH IF NO, MUST WAIT.
711 ;
712          ASSUME   IOC_G0_IRPIQ EQ 0
713          MOVL     G^EXE$AR_SYSTEM_PRIMITIVES_DATA,R2 ;IRP LIST HEAD IS 1ST CELL IN DAT
714          CLRL    -(SP)                ;MAKE ROOM ON STACK FOR IRP ADDRESS
715          $REMQHI (R2),(SP),R3        ;GET AN I/O PACKET FROM THE SIDE LIST
          CLRL     R3
30008$:
          REMQHI   (R2),(SP)
          BCC      30009$
          AOBLSS   #900000,R3,30008$
          .WORD   ^XFEFF
          .IIF IDN <FATAL>,<FATAL> , .WORD      BUG$_BADQHDR14
30009$:
30010$:
716          ; NOTE: R3 is scratch in $REMQHI macro
717          BVS     GET_IRP             ;BRANCH IF NEED TO GET ONE FROM THE POOL
718 ;
719          .DSABL  LSB
720 ;
721 ;
722 ; R0 = MASTER PTE ADDRESS IF GLOBAL CRF
723 ; = SLAVE PTE ADDRESS IF GLOBAL NOT CRF
724 ; = 0 IF NOT GLOBAL
725 ; 0(SP) = I/O REQUEST PACKET ADDRESS
726 ;

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 18  
X-39 PAGE NOT RESIDENT, QUEUE A READ REQUEST 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```

727      .ENABL  LSB
728 GOT_IRP:
729      MOVQ   4(SP),R2                ;R2=VA (LOW BITS = PAGTYP), R3=SVAPTE
730      CLRL  -(SP)                  ;INIT CRF INDICATOR TO NOT CRF PAGE
731      PUSHL RO                      ;SAVE GLOBAL, GBLCRF INDICATOR
732      BSBW  MMG$ININNEWPFN         ;ALLOCATE AND INIT A NEW PFN
733      TSTL  RO                      ;PFN ALLOCATED SUCCESSFULLY?
734      BLSS  FREPAGWAIT_5           ;BRANCH IF NOT, MUST WAIT
735      MOVL  @W^PFN$AL_PTE[R0],R3   ;GET MASTER PTE ADDRESS
736                                     ;SAME AS SLAVE UNLESS GLOBAL
737      MOVL  R3,16(SP)              ;SAVE FOR LATER USE
738      INCW  @W^PFN$AW_REFCNT[R0]   ;2ND REFERENCE FOR PAGE I/O
739 ;
740 ; FORM R2 = BACKING STORE ADDRESS
741 ; 0(SP) = MASTER PAGE TABLE ENTRY ADDRESS IF GLOBAL CRF
742 ;      = SLAVE PAGE TABLE ENTRY ADDRESS IF GLOBAL NOT CRF
743 ;      = 0 IF NOT GLOBAL PAGE
744 ; 4(SP) = 0 INITIALIZED TO NOT COPY ON REFERENCE
745 ;      SET THIS TO CORRECT BACKING STORE ADDRESS IF CRF PAGE
746 ; 8(SP) = I/O REQUEST PACKET ADDRESS
747 ; 12(SP) = VIRTUAL ADDRESS (LOW BITS = PAGTYP)
748 ; 16(SP) = SVAPTE, GLOBAL IF NOT GBL CRF, PROCESS IF NOT GBL OR IF GBL CRF
749 ;
750      BICL3  #^C<PTE$M_PROT ! PTE$M_OWN>, (R3),R1 ;R1 = PROT AND OWN
751      BICL3  R1, (R3),R2           ;R2 = TYP1 ! TYPO ! BAKX
752      BISL3  R0,R1, (R3)           ;PTE = PROT ! OWN ! PFN = TRANSITION PTE
753      BBSC  #PTE$V_TYP1,R2,20$    ;BRANCH IF PAGE FILE OR SECTION
754 ;
755 ; GLOBAL COPY ON REFERENCE PAGE
756 ;
757      MOVL  (SP)+,R3                ;GET MASTER PAGE TABLE ENTRY ADDRESS
758      BISL3 #PFN$M_GBLBAK,R2,(SP)  ;GBL BACKING STORE ADR IN CRF INDICATOR
759      PUSHL (R3)                   ;SAVE MASTER PAGE TABLE ENTRY CONTENTS
760      ASSUME PFN$V_BAKO EQ 0
761      BICL3 #^C<PFN$M_BAKO>, (R3),R2 ;GET ADDR FROM MASTER PTE
762      BRB   25$                    ;TO COPY ON REFERENCE SECTION LOGIC
763 ;
764 ; MUST WAIT FOR A FREE PAGE, 5 LONG WORDS ON STACK, FIRST 2 ARE GARBAGE
765 ; 8(SP) = I/O REQUEST PACKET ADDRESS TO BE DEALLOCATED, LAST 2 ARE SCRATCH
766 ;
767 FREPAGWAIT_5:
768      MOVL  8(SP),RO                ;I/O PACKET ADDRESS TO RO
769      ADDL  #<5*4>,SP              ;CLEAN OFF 5 LONG WORDS
770      MOVZBL #IRP$C_LENGTH,IRP$W_SIZE(R0) ;SET PACKET SIZE AND CLEAR TYPE
771      JSB  G^EXE$DEANONPAGED       ;AND DEALLOCATE IT
772      BRW  FREPAGWAIT
773 ;
774 ; PAGE FILE OR SECTION ADDRESS
775 ;
776 20$:  BBC   #PTE$V_TYP0,R2,40$    ;BRANCH IF PAGING FILE
777      BBC   #PTE$V_CRF,R2,50$     ;BRANCH IF NOT CRF
778 ;
779 ; COPY ON REFERENCE SECTION TABLE ENTRY
780 ;
781      MOVL  R2,4(SP)                ;SAVE BACKING ADDRESS IN CRF INDICATOR
782 25$:
783      ASSIGN_BACKING_STORE BAK=@W^PFN$AL_BAK[R0], FAIL=27$, RETRY=26$

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 19  
X-39 PAGE NOT RESIDENT, QUEUE A READ REQUEST 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```

                DECW   PHD$W_PRCPGFLPAGES(R5)
                BLSS   27$
26$:
                MOVL   PHD$L_PAGFIL(R5),@W^PFN$AL_BAK[R0]
784             BISB   #<PFN$M_MODIFY ! PFN$C_RDINPROG>,-
785             @W^PFN$AB_STATE[R0] ;FORCE MODIFY BIT, READ IN PROGRESS
786             BRB    60$
787
788 27$:        SWITCH_PROCESS_PAGEFILE R5,26$
                PUSHAB B^26$
                BRW    MMG$SWITCH_PRCPGFL
789 ;
790 ; PAGING FILE BACKING STORE ADDRESS
791 ;
792 30$:
793             BUG_CHECK BADPRCPGFLX,FATAL ; Bad process page file index in PTE
794 ;
795 ; PAGING FILE BACKING STORE ADDRESS
796 ;
797 40$:        MOVL   R5,R1
798             TSTL   (SP) ;IS IT GLOBAL
799             BEQL   45$ ;NO
800             MOVL   G^MMG$GL_SYSPHD,R1 ;GET SYSTEM HEADER
801
802 45$:        PUSHL  R2
803             EXTZV  #PTE$V_PRCPGFLX,- ; R2 = PROCESS page file index from PTE
804             #PTE$S_PRCPGFLX,R2,R2
805
806             ASSUME PFN$V_PGFLX EQ 24
807             ASSUME PFN$S_PGFLX EQ 8
808             MOVB   PHD$B_PRCPGFL(R1)[R2],- ; Set SYSTEM page file index
809             3(SP) ; in backing store
810             BEQL   30$ ; Bugcheck if not a valid page file
811             POPL  R2
812 ;
813 ; SECTION TABLE BACKING STORE ADDRESS
814 ;
815 50$:        BISB   #PFN$C_RDINPROG,@W^PFN$AB_STATE[R0] ;READ IN PROGRESS
816             MOVL   R2,@W^PFN$AL_BAK[R0] ;STORE BACKING STORE ADDRESS
817 ;
818 ; R0 = PFN
819 ; R2 = BACKING STORE ADDRESS
820 ; R3 = PAGE TABLE ENTRY ADDRESS, PROCESS ADR IF NOT GLOBAL,
821 ; GLOBAL ADDRESS IF GLOBAL OR GLOBAL CRF
822 ; R4 = PROCESS PCB IF PROCESS PAGE, PROCESS PAGE TABLE, OR GLOBAL PAGE,
823 ; = SYSTEM PCB IF SYSTEM PAGE OR GLOBAL PAGE TABLE
824 ; R5 = PROCESS HEADER ADDRESS CORRESPONDING TO THE ABOVE PCB ADDRESS
825 ; 0(SP) = MASTER PTE CONTENTS IF GLOBAL CRF (>0)
826 ; = SLAVE PTE ADDRESS IF GLOBAL NOT CRF (<0)
827 ; = 0 IF NOT GLOBAL
828 ; 4(SP) = 0 IF PAGE IS NOT COPY ON REFERENCE
829 ; = BACKING STORE ADDRESS (FOR GBL CRF TOO) IF CRF PAGE
830 ; 8(SP) = I/O REQUEST PACKET ADDRESS
831 ; 12(SP) = VIRTUAL ADDRESS (LOW BITS = PAGTYP)
832 ; 16(SP) = SVAPTE, GLOBAL IF NOT GBL CRF, PROCESS IF NOT GBL OR IF GBL CRF
833 ;
834 60$:        MOVL   8(SP),R1 ;ADDRESS OF I/O REQUEST PACKET

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 20

X-39 PAGE NOT RESIDENT, QUEUE A READ REQUEST 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```

835      MOVL      FP,FP_SAV(R1)          ;SAVE A REGISTER
836      MOVL      R1,FP                  ;USE THIS FOR CLUSTER CONTEXT
837      MOVQ      R4,PCB_SAV(FP)         ;SAVE PCB, PHD ADDRESSES
838      TSTL      (SP)                   ;LESS THAN 0 IF GLOBAL PAGE
839      BEQL      70$                    ;BRANCH IF NOT GLOBAL PAGE
840      BGTR      65$                    ;BRANCH IF GLOBAL CRF
841      MOVL      G^MMG$AR_SYSPCB,R4    ;USE SYSTEM PCB FOR GLOBAL PAGES
842      ;                                     ;WANT THE PRIORITY FROM IT
843 65$:   MOVL      G^MMG$GL_SYSPHD,R5    ;SYSTEM PROCESS HEADER ADDRESS
844 ;
845 ; IF THE BACKING STORE ADDRESS IN R2 IS A GLOBAL ADDRESS, THEN R3 IS THE MASTER PTE
846 ; AND R5 IS THE SYSTEM PROCESS HEADER, OTHERWISE R3 IS THE PROCESS PTE
847 ; ADDRESS AND R5 IS THE PROCESS HEADER ADDRESS.
848 ;
849 70$:   MOVQ      R2,PTEDAT(FP)         ;SAVE PTE DATA, AND ADDRESS
850
851      ASSUME     ASTPRM EQ AST+4
852      MOVQ      (SP)+,AST(FP)          ;STORE PARAMETERS TO IOPOST IN IRP
853 ;
854 ; FETCH THE TRANSFER PRIORITY FROM PROCESS PCB IF PROCESS PAGE,
855 ; PROCESS PAGE TABLE, OR GLOBAL CRF PAGE. USE SYSTEM PRIORITY
856 ; IF SYSTEM PAGE, GLOBAL PAGE, OR GLOBAL PAGE TABLE.
857 ;
858      MOVB      PCB$B_Prib(R4),PRI(FP) ;STORE PRIORITY OF TRANSFER IN IRP
859      MOVL      @W^PFN$AL_BAK[R0],BAK(FP) ;SAVE BACKING STORE ADDRESS
860      BICB3     #^C<PFN$M_MODIFY ! PFN$M_LOC>,- ;AND STATE BYTE INFORMATION
861      @W^PFN$AB_STATE[R0],STATE(FP) ;FROM PFN DATA OF FIRST PAGE
862      CLRL      (SP)                   ;THROUGH WITH IRP ADDRESS, USE FOR SCRATCH
863      ;                                     ;WILL BE PAGE TABLE FAULT CLUSTER IF PPGTBL
864      CLRL      PHVREFCADR(FP)         ;ADDRESS OF PROCESS HEADER REF CNT IF PPGTBL
865
866      ASSUME     PFN$C_PROCESS EQ 0
867      ASSUME     PFN$C_SYSTEM EQ 1
868      ASSUME     PFN$C_GLOBAL EQ 2
869      ASSUME     PFN$C_GBLWRT EQ 3
870      ASSUME     PFN$C_PPGTBL EQ 4
871      ASSUME     PFN$C_GPGTBL EQ 5
872      CMPZV     #WSL$V_PAGTYP,#WSL$S_PAGTYP,- ;IF PAGE TABLE PAGE
873      4(SP),#PFN$C_PPGTBL              ;THEN SEPARATE CLUSTER FACTOR
874      BLSS      90$                    ;BRANCH IF NOT PAGE TABLE
875      BGTR      80$                    ;BRANCH IF GLOBAL PAGE TABLE
876 ;
877 ; MUST RECORD A PROCESS HEADER REFERENCE FOR PAGE READ OF PROCESS HEADER PAGE
878 ;
879      MOVZWL     PHD$W_PHVINDEX(R5),R1 ;PROCESS HEADER VECTOR INDEX
880      MOVAW      @PHV$GL_REFEBAS[R1],R1 ;ADDRESS OF PROCESS HEADER REF CNT
881      INCW      (R1)                   ;COUNT ANOTHER REFERENCE
882      MOVL      R1,PHVREFCADR(FP)      ;SAVE ADDRESS FOR CLUSTERING CODE
883      MOVB      PHD$B_PGTBPF(CR5),(SP) ;GET PAGE TABLE CLUSTER FACTOR
884      BNEQ      90$                    ;BRANCH IF SPECIFIED
885 80$:   MOVL      #1,(SP)              ;INDICATE NO CLUSTERING
886 90$:   BSBW      MMG$INIBLDPKT        ;SET UP REGISTERS TO CALL BUILDPKT
887      BBS       #EXE$V_NOCLUSTER,G^EXE$GL_FLAGS,110$ ;BRANCH IF CLUSTERING DISABLED
888      MOVQ      PCB_SAV(FP),R4        ;RECOVER PCB, PHD ADDRESSES
889
890      ASSUME     SEC$B_PFC EQ PFL$B_PFC
891      MOVB      SEC$B_PFC(R1),1(SP)    ;PAGE FAULT CLUSTER FROM

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 21

X-39 PAGE NOT RESIDENT, QUEUE A READ REQUEST 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```

892                                ;SECTION OR PAGE FILE CONTROL BLOCK
893      MOVZBL  (SP),R1           ;SEE IF PAGE TABLE CLUSTER SPECIFIED
894      BGTR    100$              ;BRANCH IF SPECIFIED
895      MOVZBL  1(SP),R1         ;SEE IF PAGE FILE OR SECTION TABLE
896                                ;CLUSTER WAS SPECIFIED
897      BGTR    100$              ;BRANCH IF IT WAS
898      MOVZBL  PHD$B_DFPFC(R5),R1 ;DEFAULT FROM PROCESS HEADER
899                                ;PROCESS IF PROCESS OR GLOBAL PAGE
900                                ;SYSTEM IF SYSTEM PAGE
901 100$:  CMLP    R1,#1           ;CLUSTER OF 1?
902      BLEQ    110$              ;BRANCH IF YES
903      ADDL    #4,SP             ;CLEAN OFF CLUSTER FACTOR SCRATCH
904      BRW     TRY_TO_CLUSTER    ;GO TRY TO CLUSTER
905 110$:  MOVL   #1,R1           ;ONE PAGE READ
906      ADDL    #4,SP             ;CLEAN OFF CLUSTER FACTOR SCRATCH
907      POPL   VA(FP)            ; Save VA (for page type bits)
908      MOVL   (SP)+,R3          ;SVAPTE FROM PFN$AL_PTE
909
910      .DSABL  LSB
911 ;
912 ; R0 = VBN, R1 = PAGE COUNT, R2 = WINDOW, R3 = SVAPTE, FP = IRP
913 ; VA(FP) = page type (in low 9 bits)
914 ;
915 QUEUE_PAGE_READ:
916
917      INCL   G^PMS$GL_PREADIO    ;COUNT PAGE READ I/O REQUESTS (SYSTEM)
918      ADDL   R1,G^PMS$GL_PREADS  ;AND THE NUMBER OF PAGES READ
919 ;
920 ; If this is a system or global page table page, count the I/O in the system header
921 ;
922      ASSUME  PFN$C_PROCESS EQ 0
923      ASSUME  PFN$C_SYSTEM  EQ 1
924      ASSUME  PFN$C_GLOBAL  EQ 2
925      ASSUME  PFN$C_GBLWRT  EQ 3
926      ASSUME  PFN$C_PPGTBL  EQ 4
927      ASSUME  PFN$C_GPGTBL  EQ 5
928
929      EXTV   #WSL$V_PAGTYP,#WSL$S_PAGTYP,- ; R4 = sign-extended page type
930      VA(FP),R4
931      BLBC   R4,5$              ; BR if process, global, or process page tab
932      BLSS   1$                 ; BR if global page table page
933      BBS    #1,R4,5$           ; BR if global writable page
934 1$:      MOVL  G^MMG$GL_SYSPHD,R5 ; Fetch system PHD address
935      INCL   PHD$L_PGFLTIO(R5)   ; Count page read I/O requests (SYSTEM)
936
937 5$:      PUSHL  R3              ; Save SVAPTE
938
939      .IF    GT,CA$ MEASURE
940      TSTL   ASTPRM(FP)         ; CRF page ?
941      BEQL   10$                ; BR if not
942      EXTZV  #PFN$V_GBLBAK,#1,ASTPRM(FP),R3 ; Differentiate private and global CR
943      INCL   G^PMS$AL_CRFIO[R3] ; Count CRF page I/O requests
944      ADDL2  R1,G^PMS$AL_CRF[R3] ; and the number of CRF pages
945      MOVL   (SP),R3           ; Restore SVAPTE
946 10$:
947      .ENDC
948

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 22  
X-39 PAGE NOT RESIDENT, QUEUE A READ REQUEST 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```

949      MOVL      G^CTL$GL_PCB,R4          ;PCB ADDRESS
950      MOVL      PCB$SL_PHD(R4),R5       ;PHD ADDRESS
951      INCL      PHD$SL_PGFLTIO(R5)      ;COUNT PAGE READ I/O REQUESTS (PROCESS)
952      ASHL      #9,R1,R1                ;FORM BYTE COUNT TO TRANSFER
953      MOVL      FP,R5                    ;I/O PACKET ADDRESS
954      MOVL      FP_SAV(R5),FP           ;RESTORE SAVED REGISTER
955
956 ;
957 ; 0(SP) =          SVAPTE (Master if global, slave if process or GBL CRF)
958 ;
959 ; R0 = VBN
960 ; R1 = NUMBER OF BYTES TO READ
961 ; R2 = WINDOW ADDRESS
962 ; R3 = SVAPTE (MASTER IF GLOBAL, SLAVE IF GLOBAL CRF)
963 ; R4 = PROCESS PCB ADDRESS
964 ; R5 = I/O REQUEST PACKET ADDRESS
965 ;
966 ; IRP$SL_AST(R5)
967 ;   = MASTER PTE CONTENTS IF GLOBAL CRF (>0)
968 ;   = SLAVE PTE ADDRESS IF GLOBAL NOT CRF (<0)
969 ;   = 0 IF NOT GLOBAL
970 ; IRP$SL_ASTPRM(R5)
971 ;   = BACKING STORE ADDRESS IF CRF PAGE (GBLBAK SET IF GBL CRF)
972 ;   = 0 IF NOT CRF PAGE
973 ; IRP$B_PRI(R5) = DESIRED TRANSFER PRIORITY
974 ;
975      UNLOCK    LOCKNAME=MMG             ;UNLOCK MMG ACCESS
976                                           ;** DON'T CHANGE IPL
977      JSB      G^EXE$BUILDPKTR         ;BUILD AND QUEUE THE I/O PACKET
978 ;
979 ; THE FOLLOWING WAITS THE PROCESS AT THE FAULTING MODE
980 ;
981 ; 0(SP) =          SVAPTE (Master if global, slave if process or GBL CRF)
982 ;
983 PROCPAG:
984      LOCK     LOCKNAME=SCHED,-         ;LOCK SCHED DATA BASE
985           PRESERVE=NO                 ;DON'T PRESERVE R0
986 ;
987 ; Test the PTE contents to see if the page is valid. This can
988 ; happen if the page fault I/O completes after releasing the
989 ; MMG and SCHED spin locks. Not testing the page can result in
990 ; the process entering a persistent PFW wait state.
991 ;
992 ; This test depends upon the following assumptions:
993 ;
994 ; 1. Page read completion code in IOCIOPST has the following
995 ;    processing template.
996 ;
997 ;           LOCK     MMG
998 ;           <Do all MMG page read completion processing, SETTING PTE VALID BIT L
999 ;           LOCK     SCHED
1000 ;           Declare PFCOM event for process
1001 ;           If required, declare COLPGA event for all processes in the COLPG que
1002 ;           UNLOCK   SCHED
1003 ;           UNLOCK   MMG
1004 ;
1005 ; 2. IPL remains at SYNCH after releasing the MMG lock. Thus, the faulting

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 23

X-39 PAGE NOT RESIDENT, QUEUE A READ REQUEST 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```

1006 ;           process remains CURrent, which implies that the process cannot be
1007 ;           affected by the SWAPPER in any way.
1008 ;
1009     TSTL     @(SP)+           ; Was page concurrently made valid ?
1010     BLSS     10$             ; BR if yes to avoid waiting process
1011 ;
1012     MOVAQ    G^SCH$GQ_PFWQ,R0 ;PAGE FAULT WAIT QUEUE ADDRESS
1013     PUSHAB   B^PGFEXIT       ;Stack subroutine return address
1014     BRW      MMG$PGFLTWAIT_1 ;PUT PROCESS ON PAGE FAULT WAIT QUEUE
1015 ;
1016 10$:     UNLOCK  LOCKNAME=SCHED ;UNLOCK SCHED ACCESS
1017 ;           ;** DON'T CHANGE IPL
1018     BRW      PGFCOMPLETE2    ;Branch helper
1019 PGFEXIT:
1020     MOVQ     (SP)+,R0         ;RESTORE THE REGISTERS
1021     MOVQ     (SP)+,R2         ;
1022     MOVQ     (SP)+,R4         ;
1023     ADDL    #8,SP           ;CLEAN OFF THE EXCEPTION PARAMETERS
1024 ;
1025 ; STACK NOW CONTAINS JUST THE FAULT PC, PSL PAIR
1026 ;
1027     UNIVERSAL_SYMBOL      MMG$SVPCTX
1028 ;MMG$SVPCTX:
1029     SVPCTX                ;SAVE PROCESS CONTEXT, WITH R4 SAME
1030 ;           ; AS WHEN PAGEFAULT OCCURRED
1031     MOVL     G^CTL$GL_PCB,R4 ;NOW GET PCB ADDRESS
1032     JMP      G^SCH$WAITM     ;JOIN COMMON WAIT CODE FOLLOWING SVPCTX
1033 ;           ; SCH$WAITM will release SCHED spinlock
1034 ;
1035 ; WAIT FOR RESOURCE IN R1 TO BECOME AVAILABLE
1036 ;
1037 RESOURCEWAIT:
1038     BSBB     MMG$RESRCWAIT   ;SET UP TO WAIT FOR THE RESOURCES
1039 ;           ; - RELEASE MMG LOCK, ACQUIRE SCHED LOCK
1040     BRB      PGFEXIT        ;AND EXIT TO THE SCHEDULER
1041 ;
1042 ; FAULT FOR PAGE WHICH IS ALREADY ON THE WAY INTO MEMORY
1043 ;
1044 READINPROG:
1045     MOVQ     (SP)+,R2         ;R2=VA (LOW BITS = PAGTYP), R3=SVAPTE
1046     ASSUME   PFN$C_PROCESS EQ 0
1047     BITB     #PFN$M_PAGTYP,@W^PFN$AB_TYPE[R0] ;PROCESS PAGE?
1048     BNEQ    5$              ;BRANCH IF NO
1049     UNLOCK   LOCKNAME=MMG,-   ;UNLOCK MMG ACCESS
1050     PRESERVE=NO             ;DON'T PRESERVE R0
1051 ;           ;** DON'T CHANGE IPL
1052     PUSHL   R3               ; SVAPTE must be on top of stack
1053     BRB      PROCPAG         ;CONTINUE
1054 5$:      ;
1055 ; COLLIDED PAGE (NOT A PROCESS PAGE)
1056 ;
1057     BISB     #PFN$M_COLLISION,@W^PFN$AB_TYPE[R0] ;COLLISION OCCURRED
1058     MOVAQ    G^SCH$GQ_COLPGWQ,R0 ;COLLISION PAGE WAIT QUEUE
1059 FREEPAGEWAIT1:
1060     LOCK     LOCKNAME=SCHED   ;LOCK SCHED DATA BASE
1061     UNLOCK   LOCKNAME=MMG     ;UNLOCK MMG ACCESS
1062     BSBB     MMG$PGFLTWAIT_1 ;PLACE PROCESS ON THE COLLISION QUEUE

```



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 24

X-39 PAGE NOT RESIDENT, QUEUE A READ REQUEST 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```

1063         BRB         PGFEXIT                ;EXIT TO THE SCHEDULER
1064 ;
1065 ; NO FREE PAGES AVAILABLE ON THE FREE PAGE LIST, MUST WAIT
1066 ;
1067 FREEPAGEWAIT:
1068         MOVQA        G^SCH$GQ_FPGWQ,R0      ;WAIT ON FREE PAGE WAIT QUEUE
1069         BRB         FREEPAGEWAIT1
1070 ;
1071 ; WAIT FOR RESOURCE IN R1 TO BECOME AVAILABLE
1072 ; R0,R1,R2,R3 ALTERED, R4 RETURNED WITH CURRENT PCB ADDRESS
1073 ;
1074 ; WE MUST HOLD ONTO THE MMG SPINLOCK UNTIL AFTER THE RESMASK BIT IS SET!
1075 ;
1076         .ENABL      LSB
1077         UNIVERSAL_SYMBOL          MMG$RESRCWAIT
1078 ;MMG$RESRCWAIT::
1079         LOCK        LOCKNAME=SCHED,-        ;LOCK SCHED DATA BASE
1080         PRESERVE=NO                ;DON'T PRESERVE R0
1081         MOVL        G^CTL$GL_PCB,R4        ;R4 = CURRENT PCB ADDRESS
1082         MOVL        R1,PCB$E_FWM(R4)      ;SET RESOURCE NEEDED
1083         BBSI        R1,G^SCH$GL_RESMASK,10$ ;NOTE SOMEONE WAITING
1084 10$:
1085         UNLOCK      LOCKNAME=MMG,-        ;UNLOCK MMG ACCESS
1086         PRESERVE=NO                ;DON'T PRESERVE R0
1087         ;** DON'T CHANGE IPL
1088         MOVQA        G^SCH$GQ_MWAIT,R0     ;WAIT ON MISCELLANEOUS QUEUE
1089         BRB         20$                ;GO WAIT THIS PROCESS
1090 MMG$PGFLTWAIT_1:
1091         MOVL        G^CTL$GL_PCB,R4        ;MUST WAIT THE PROCESS PCB
1092
1093         UNIVERSAL_SYMBOL          MMG$PGFLTWAIT
1094 ;MMG$PGFLTWAIT::
1095 20$:
1096         INCW        WQH$W_WQCNT(R0)        ;COUNT THIS PROCESS WAITING
1097         INSQUE      (R4),(R0)            ;QUEUE THIS PCB
1098         MOVW        WQH$W_WQSTATE(R0),PCB$W_STATE(R4) ;SET WAIT STATE IN PCB
1099         RSB
1100

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 25

X-39 FORM A CLUSTER OF PAGES TO READ 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (9)

```

1102      .SBTTL  FORM A CLUSTER OF PAGES TO READ
1103 ;
1104 ; R0 = VBN IN FILE OF FIRST PAGE TO READ
1105 ; R1 = DESIRED CLUSTER SIZE
1106 ; R2 = WINDOW CONTROL BLOCK ADDRESS
1107 ; R4 = PCB ADDRESS, PROCESS IF PROCESS OR GLOBAL PAGE, SYSTEM IF SYSTEM PAGE
1108 ; R5 = PHD ADDRESS, PROCESS IF PROCESS OR GLOBAL PAGE, SYSTEM IF SYSTEM PAGE
1109 ; FP = I/O REQUEST PACKET ADDRESS
1110 ; 0(SP) = VIRTUAL ADDRESS (LOW BITS = PAGTYP)
1111 ; 4(SP) = SVAPTE FROM PFN$AL_PTE
1112 ;
1113
1114      .ENABL  LSB
1115
1116 TRY_TO_CLUSTER:
1117      MOVL    (SP)+,VA(FP)          ;SAVE VIRTUAL ADDRESS
1118      MOVL    R0,VBN(FP)           ;SET VIRTUAL BLOCK NUMBER
1119
1120      ASSUME  COUNT EQ CLUSTER+1
1121      BISW3   #^X0100,R1,CLUSTER(FP) ;SET COUNT AND CLUSTER
1122      MOVL    R2,WINDOW(FP)        ;WINDOW ADDRESS
1123 ;
1124 ; PUT PTEDAT INTO FORM OF TYP1 ! TYPO ! BAKX
1125 ;
1126      ASSUME  PFN$V_PGFLX GE 24
1127      ASSUME  PTE$V_TYP1 GE 24
1128      MOVBL   #PTE$M_TYP1@-24,PTEDAT+3(FP) ;TURN TYP1 BACK ON, CLEAR PAGE FILE IND
1129      MOVL    AST(FP),R3           ;PROCESS PTE ADR IF GBL NOT CRF
1130      BEQL    30$                 ;BRANCH IF NOT GLOBAL PAGE
1131      BLSS    10$                 ;BRANCH IF GBL NOT CRF
1132 ;
1133 ; GLOBAL COPY ON REFERENCE PAGE
1134 ;
1135      MOVL    (SP),R3             ;PROCESS PTE ADR WHEN GBL CRF
1136      MOVL    ASTPRM(FP),R2       ;GPTX PTE CONTENTS FOR THIS CASE
1137      BRB     20$
1138
1139 CLU_END1:
1140      BRW     CLU_END
1141
1142 ;
1143 ; GLOBAL PAGE NOT COPY ON REFERENCE
1144 ;
1145 10$:   MOVL    (R3),R2           ;GPTX FROM PROCESS PTE
1146 20$:   BICL    #^C<PTE$M_TYP1 ! PTE$M TYPO !- ;ISOLATE PAGE TYPE
1147      PTE$M_GPTX>,R2           ;AND GPTX BITS
1148
1149      ASSUME  GPTX_PTE EQ GPTX+4
1150 30$:   MOVQ    R2,GPTX(FP)       ;SET GPTX AND GPTX_PTE
1151      MOVL    #1,INC1(FP)        ;INIT TO SCAN FORWARDS
1152
1153      .DSABL  LSB
1154
1155 CLU_INI_INC:
1156      ASHL    #2,INC1(FP),INC4(FP) ;+ OR - 4
1157      ASHL    #9,INC1(FP),INC512(FP) ;+ OR - 512
1158 CLU_NXT:

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 26

X-39 FORM A CLUSTER OF PAGES TO READ 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (9)

```

1159      ADDL3   INC4(FP),SVAPTE(FP),R3 ;NEXT PTE TO CHECK
1160      MOVL    R3,SVAPTE(FP)           ;UPDATE CONTEXT
1161      BBS     #PTE$V_TYPO,PTEDAT(FP),20$ ;BRANCH IF SECTION ADDRESS
1162      ADDL    INC1(FP),PTEDAT(FP)       ;INCREMENT PAGE FILE ADDRESS
1163 20$:   EXTZV  #VA$V_VPN,#VA$$_VPN,R3,R1 ;CHECK THAT PTE IS RESIDENT
1164      TSTL    @W^MMG$GL_SPTBASE[R1]    ;BY MAKING SURE ITS SPTTE IS VALID
1165      BGEQ    CLU_END1                  ;BRANCH IF IT ISN'T
1166      BICL3   #^C<PTE$M_VALID !-      ;GET VALID BIT
1167      PTE$M_TYP1 ! PTE$M_TYPO ! - ;PAGE TYPE BITS
1168      PTE$M_BAKX>, (R3),R0            ;AND PGFLVBN/PRCPGFLX or GPTX FROM PTE
1169      CMLP    R0,PTEDAT(FP)            ;MUST AGREE IF THIS PAGE IS IN THE CLUSTER
1170      BNEQ    CLU_END1                  ;BRANCH IF AT END OF CLUSTER
1171      TSTL    GPTX_PTE(FP)              ;WAS THAT THE MASTER PTE FOR A GLOBAL?
1172      BEQL    60$                       ;BRANCH IF NO, IT WAS PROCESS PTE
1173 ;
1174 ; MUST TEST THAT PROCESS PTE POINTS AT THE GPTX
1175 ;
1176      ADDL3   INC4(FP),GPTX_PTE(FP),R3 ;NEXT PROCESS PTE ADR
1177      EXTZV  #VA$V_VPN,#VA$$_VPN,R3,R1 ;CHECK THAT THIS PTE IS ACCESSIBLE
1178      TSTL    @W^MMG$GL_SPTBASE[R1]    ;BY MAKING SURE ITS SPTTE IS VALID
1179      BGEQ    CLU_END1                  ;BRANCH IF IT ISN'T
1180      BICL3   #^C<PTE$M_VALID ! -      ;GET VALID BIT
1181      PTE$M_TYP1 ! PTE$M_TYPO ! - ;PAGE TYPE BITS
1182      PTE$M_BAKX>, (R3),R0            ;AND PGFLVBN/PRCPGFLX or GPTX FROM PTE
1183      ADDL3   INC1(FP),GPTX(FP),R2     ;NEXT GLOBAL PAGE TABLE INDEX
1184      CMLP    R0,R2                     ;IN THE CLUSTER?
1185      BNEQ    CLU_END1                  ;BRANCH IF NOT
1186 ;
1187      ASSUME  GPTX_PTE EQ GPTX+4
1188      MOVQ    R2,GPTX(FP)               ;UPDATE GPTX
1189 ;
1190 ; R1 = SPT INDEX FOR PAGE TABLE PAGE
1191 ; R3 = PROCESS PTE ADDRESS
1192 ;
1193 60$:   PUSHL  R3                       ;SAVE PROCESS PTE ADR
1194      PUSHL  R1                       ;SAVE SPT INDEX
1195      BSBW  MMG$FREWSLE                 ;GET A FREE WORKING SET LIST ENTRY
1196      MOVQ  (SP)+,R2                   ;RESTORE PROCESS PTE ADR, SPT INDEX
1197      BLBC  R0,CLU_END_RESRC1         ;IF CANNOT GET ONE, END THE CLUSTER
1198 ;
1199 ; MUST CHECK THE SPT ENTRY FOR PROCESS PAGE TABLE IS STILL VALID
1200 ; FREWSLE MIGHT HAVE DISCARDED IT FROM THE WORKING SET.
1201 ;
1202      TSTL    @W^MMG$GL_SPTBASE[R2]    ;IS SPT ENTRY FOR PT STILL VALID
1203      BGEQ    CLU_END_RESRC1           ;BRANCH IF NOT
1204      ADDL3   INC512(FP),VA(FP),R2     ;NEXT VIRTUAL ADDRESS
1205      MOVL    R2,VA(FP)                 ;UPDATE THE CONTEXT
1206      BSBW  MMG$ININWPFN                 ;ALLOC AND INIT A PFN
1207      TSTL    R0                         ;IF NO PFN'S AVAILABLE
1208      BLSS  CLU_END_RESRC1             ;THEN END THE CLUSTER
1209      INCB   COUNT(FP)                  ;COUNT ANOTHER PAGE IN THE CLUSTER
1210      MOVL    PHVREFCADR(FP),R1        ;PROCESS HEADER REF CNT ADR
1211      ;
1212      BEQL    70$                       ;BRANCH IF NOT A PROCESS PAGE TABLE PAGE
1213      INCW   (R1)                       ;COUNT ANOTHER PROCESS HEADER REFERENCE
1214 70$:   MOVL    @W^PFN$AL_PTE[R0],R3   ;PROCESS PTE ADR IF NOT GLOBAL OR IF GBL CRF
1215      ;
1216      GLOBL  PTE_ADR IF GLOBAL NOT CRF

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 27

X-39 FORM A CLUSTER OF PAGES TO READ 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (9)

```

1216      INCW      @W^PFN$AW_REFcnt[R0]      ;SECOND REFERENCE FOR I/O IN PROGRESS
1217      BICL3     #^C<PTE$M_PROT ! PTE$M_OWN>, (R3), R1 ;PROTECTION AND OWNER FIELDS
1218      BISL3     R0, R1, (R3)                ;FORM TRANSITION PTE FORMAT
1219      MOVL      BAK (FP), R2                ;BACKING STORE FROM PREV PFN
1220      BBS       #PTE$V_TYPO, R2, 80$        ;BRANCH IF SECTION ADDRESS
1221      BITL      #PFN$M_PGFLVBN, R2         ;IF NOT A NULL PAGE FILE ADDRESS
1222      BEQL      75$
1223      ADDL      INC1 (FP), R2                ;THEN INCREMENT THE ADDRESS
1224      MOVL      R2, BAK (FP)                ;AND UPDATE THE CONTEXT
1225      BRB       80$
1226 75$:   ASSIGN  BACKING STORE BAK=R2, FAIL=90$, RETRY=77$
          DECW    PHD$W_PRCPGFLPAGES (R5)
          BLSS   90$
          77$:
1227 80$:   MOVL    PHD$L_PAGFIL (R5), R2
          MOVL    R2, @W^PFN$AL_BAK [R0]      ;SET BACKING STORE ADR FOR THIS PFN
1228      BISB     STATE (FP), @W^PFN$AB_STATE [R0] ;USE STATE FROM PREV PFN
1229      CMPB     COUNT (FP), CLUSTER (FP)    ;IS CLUSTER FULL?
1230      BGEQ     CLU_END_RESRC                ;BRANCH IF YES, QUEUE THE READ
1231      BRW      CLU_NXT                       ;NO, TRY FOR ANOTHER PAGE
1232
1233 90$:   SWITCH_PROCESS_PAGEFILE R5, 77$
          PUSHAB B^77$
          BRW     MMG$SWITCH_PRCPGFL
1234
1235 CLU_END_RESRC1:
1236      BRB     CLU_END_RESRC
1237 ;
1238 ; END OF CLUSTER
1239 ;
1240 CLU_END:
1241      CMPB     COUNT (FP), #1                ;IF AT LEAST 2 PAGES IN CLUSTER
1242      BGTR     CLU_END_RESRC                ;THEN READ THE CLUSTER
1243      MNEGL   INC1 (FP), INC1 (FP)          ;OTHERWISE TRY TO SCAN BACKWARDS
1244      BGTR     CLU_END_RESRC                ;UNLESS ALREADY TRIED THAT
1245      SUBL     #4, SVAPTE (FP)              ;BACK TO STARTING SVAPTE
1246      BBS     #PTE$V_TYPO, PTEDAT (FP), 20$ ;BRANCH IF SECTION PAGE
1247      DECL    PTEDAT (FP)                  ;BACK TO ORIG PAGE FILE VBN
1248 20$:   BRW     CLU_INI_INC
1249 ;
1250 ; SET UP TO DO THE PAGE READ
1251 ;
1252 CLU_END_RESRC:
1253      MOVL     (SP)+, R3                      ;GET PTE ADR OF FIRST PFN IN CLUSTER
1254      MOVL     VBN (FP), R0                  ;AND ITS ASSOCIATED VBN IN THE FILE
1255      MOVZBL   COUNT (FP), R1                ;NUMBER OF PAGES IN THE CLUSTER
1256      TSTL    INC1 (FP)                      ;IF CLUSTER WENT BACKWARDS
1257      BGTR     20$
1258      SUBL3    R1, #1, R2                    ;-(COUNT-1)
1259      ADDL     R2, R0                        ;ADJUST FILE VBN
1260      MOVAL    (R3) [R2], R3                ;AND PTE ADR
1261      MOVL     AST (FP), R4                  ;PROCESS PTE ADDRESS FOR GLOBAL PAGE?
1262      BEQL     20$                           ;BRANCH IF NOT GLOBAL
1263      BLSS    10$                             ;BRANCH IF GLOBAL BUT NOT CRF
1264      ADDL     R2, ASTPRM (FP)              ;ADJUST GLOBAL BACKING STORE ADDRESS
1265      BRB     20$
1266 10$:   MOVAL    (R4) [R2], AST (FP)        ;ADJUST PROCESS PTE ADDRESS FOR GLOBAL PAGE

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

**PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 28**

**X-39 FORM A CLUSTER OF PAGES TO READ 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (9)**

|                   |             |                        |                                |
|-------------------|-------------|------------------------|--------------------------------|
| <b>1267 20\$:</b> | <b>MOVL</b> | <b>WINDOW(FP),R2</b>   | <b>;GET WINDOW ADDRESS</b>     |
| <b>1268</b>       | <b>BRW</b>  | <b>QUEUE_PAGE_READ</b> | <b>;GO QUEUE THE PAGE READ</b> |

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 29

X-39 DEMAND ZERO PAGE 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (10)

```

1270      .SBTTL DEMAND ZERO PAGE
1271 ;
1272 ; MUST WAIT FOR FREE PAGES TO BECOME AVAILABLE
1273 ;
1274 DZROFPWAIT 5:
1275      ADDL    #<5*4>,SP          ;CLEAN OFF 5 LONG WORDS
1276      BRW    FREEPAGEWAIT      ;AND GO WAIT FOR A FREE PAGE
1277 ;
1278 ; THIS IS A DEMAND ZERO FORMAT PAGE TABLE ENTRY, R0 = 0
1279 ;
1280 DZRO_PTE:
1281      CLRL   R3                  ;NO GLOBAL MASTER PTE ADDRESS
1282      CLRL   R1                  ;NO MASTER PTE CONTENTS
1283 ;
1284 ; R0 = BACKING STORE ADDRESS, TYP1 ! TYPO ! BAKX
1285 ; R1 = MASTER PAGE TABLE ENTRY CONTENTS IF GLOBAL, 0 IF NOT
1286 ; R3 = GLOBAL PAGE TABLE ENTRY ADDRESS IF GLOBAL, 0 IF NOT
1287 ; 0(SP) = FAULT VA (LOW BITS = PAGTYP)
1288 ; 4(SP) = CORRESPONDING SVAPTE
1289 ;
1290 DZRO_GBL_SEC:
1291      PUSHL  R3                  ;SAVE GBL PTE ADDR
1292      PUSHL  R1                  ;SAVE MAGSTER PTE CONTENTS
1293      PUSHL  R0                  ;SAVE BACKING STORE ADDRESS
1294      MOVQ   12(SP),R2          ;R2=VA, R3=SVAPTE
1295      BSBW   MMG$ININEWPFN      ;ALLOCATE AND INIT A PFN
1296      TSTL   R0                  ;SEE IF A PFN WAS ALLOCATED
1297      BLSS   DZROFPWAIT_5      ;BRANCH IF HAVE TO WAIT
1298      BISB   #PFN$C_ACTIVE,@W^PFN$AB_STATE[R0] ;MARK PAGE ACTIVE
1299      MOVL   (SP)+,R1          ;GET BACKING STORE ADDRESS
1300      BEQL   15$                ; Branch if demand zero format
1301      BBS    #PTE$V_CRF,R1,15$  ; Branch if DZRO, CRF section
1302      BBS    #PTE$V_TYP1,R1,10$ ;CHECK FOR GLOBAL WITH PAGE FILE BACING STOR
1303      BBCC   #PTE$V TYPO,R1,10$
1304 ; PAGE FILE BACKING STORE GLOBAL SECTION
1305 3$:      MOVL   G^MMG$GL SYSPHD,R2 ;GET SYSTEM HEADER
1306      ASSIGN _BACKING_STORE PHD=R2, BAK=R2, SUCCESS=20$, FAIL=5$, RETRY=4$
      DECW   PHD$W_PRCPGFLPAGES(R2)
      BLSS   5$
      4$:
      MOVL   PHD$L_PAGFIL(R2),R2
      BRB    20$
1307
1308 ; Failed to reserve pagefile using system header
1309 5$:      PUSHL  R5
1310      MOVL   R2,R5
1311      BSBB   9$
1312      POPL   R5
1313      BRB    4$                ; Make reservation using new pagefile
1314
1315 ; Failed to reserve pagefile using process header
1316 7$:      PUSHAB B^17$         ; Fall through to local subroutine
1317
1318 ;
1319 ; Try to switch to a new pagefile
1320 ; PHD address (process or system) in R5
1321 ;

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 30

X-39 DEMAND ZERO PAGE 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (10)

```

1322 9$: SWITCH_PROCESS_PAGEFILE R5
      BSBW MMG$SWITCH_PRCPGFL
1323 RSB
1324
1325 10$: BICL3 #<PTE$M_DZRO ! ^C<PFN$M_BAKO>>,R1,R2 ;BACKING STORE ADR
1326 ;WITH DZRO SHUT OFF
1327 BRB 20$
1328
1329 15$: ASSIGN_BACKING_STORE BAK=R2, FAIL=7$, RETRY=17$
      DECW PHD$W_PRCPGFLPAGES(R5)
      BLSS 7$
17$:
      MOVL PHD$L_PAGFIL(R5),R2
1330
1331 20$: MOVL R2,@W^PFN$AL_BAK[R0] ;STORE THE BACKING STORE ADDRESS
1332 ;
1333 ; 0(SP) = MASTER PTE CONTENTS IF GLOBAL, 0 IF NOT
1334 ; 4(SP) = MASTER PTE ADDRESS IF GLOBAL, 0 IF NOT
1335 ; 8(SP) = VIRTUAL ADDRESS (LOW BITS = PAGE TYPE)
1336 ; 12(SP) = SYSTEM VIRTUAL ADDRESS OF PROCESS PAGE TABLE ENTRY
1337 ;
1338 MOVQ 8(SP),R2 ;R2=VA (LOW BITS = PAGTYP), R3=SVAPTE
1339 BICL3 #^C<PTE$M_PROT ! PTE$M_OWN>, -
1340 (R3),-(SP) ;PROTECTION AND OWNER FROM PTE
1341 BISL3 #<PTE$M_VALID ! PTE$C_KW ! -
1342 PTE$M_MODIFY>,R0,(R3) ;MAKE PAGE KERNEL WRITE FOR ZEROING
1343 .IF CA$ MEASURE
1344 INCL G^PMS$GL_DZROFLTS ;COUNT DEMAND ZERO PAGE FAULTS
1345 .ENDC
1346
1347 ;
1348 ; ***** BE AWARE THAT THE FOLLOWING CLRБ ASSUMES THAT BIT 8 IS NOT
1349 ; ***** IN USE FOR ANY OF THE PAGE TYPE FLAGS, ETC.
1350 ;
1351 CLRБ R2 ;CLEAR OUT THE PAGE TYPE
1352 PUSHL R3 ;SAVE AROUND MOVС5
1353 MOVС5 #0,(R4),#0,#^X200,(R2) ;ZERO THE PAGE, PCB ADDRESS TO R1
1354 MOVL (SP)+,R3 ;RESTORE REGISTER
1355 MOVL R1,R4 ;RECOVER PCB ADDRESS
1356 MOVL (SP)+,R1 ;PICK UP SAVED PROT AND OWNER
1357 MOVL 8(SP),R2 ;VA TO INVALIDATE
1358 BICL3 #^C<PTE$M_VALID ! PTE$M_MODIFY ! -
1359 PTE$M_PFN>,(R3),R0 ;CLEAR OUT ALL BUT VALID,MODIFY, AND PFN
1360 INVALIDATE TB R2,- ;INVALIDATE TRANSLATION BUFFER
1361 INST1=<BISL3 R0,R1,(R3)>;SET PTE WITH CORRECT PROT AND OWNER

; ++
; NB: Co-routine address + 2 LWs are on top of stack during execution
; --
1362 MOVQ (SP),R1 ;SEE IF DZRO GLOBAL
1363 BEQL 60$ ;BRANCH IF NOT, BOTH ARE ZERO
1364 ;
1365 ; GLOBAL DEMAND ZERO PAGE, MAKE MASTER PTE VALID TOO
1366 ;
1367 40$: BICL #^C<PTE$M_PROT ! PTE$M_OWN>,R1 ;MASTER PTE PROTECTION AND OWNER
1368 BISL3 R0,R1,(R2) ;SET MASTER PTE VALID
1369 60$: ADDL #4*4,SP ;CLEAN OFF 4 LONG WORDS

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 31  
X-39 DEMAND ZERO PAGE 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (10)

```
1370      .dsabl  lsb
1371 PGFCOMPLETE:
1372      UNLOCK  LOCKNAME=MMG,-      ;RELEASE MMG LOCK/PRESERVE IPL
1373      PRESERVE=NO                  ;
1374 PGFCOMPLETE2:
1375      MOVQ    (SP)+,R0              ;RESTORE THE REGISTERS
1376      MOVQ    (SP)+,R2              ;      ...
1377      MOVQ    (SP)+,R4              ;      ...
1378      ADDL    #8,SP                 ;CLEAN OFF THE EXCEPTION PARAMETERS
1379      REIMAC                        ;AND RETURN FROM THE EXCEPTION
```



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 32  
X-39 FREE, MODIFIED, OR BAD PAGE LIST, RELEAS 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```

1381      .SBTTL  FREE, MODIFIED, OR BAD PAGE LIST, RELEASE PENDING
1382 ;
1383 ; THIS IS A FAULT OFF THE FREE, MODIFIED, OR BAD PAGE LISTS
1384 ; R0 = PFN, R2 = LISTID,
1385 ; 0(SP) = VA (LOW BITS = PAGTYP), 4(SP) = SVAPTE
1386 ;
1387      .ENABL  LSB
1388 PFNLIST:
1389      BSBW    MMG$REMPFN          ;REMOVE PFN FROM LIST
1390 WRITEINPROG:
1391 RELEASEPEND:
1392      MOVQ    (SP),R2              ;R2 = VA, R3 = SVAPTE
1393      BSBW    MMG$MAKEWSLE        ;MAKE A WORKING SET LIST ENTRY
1394      MOVQ    (SP)+,R2            ;R2=VA, R3=SVAPTE
1395 ;
1396 ; SET PAGE ACTIVE AND VALID
1397 ;
1398      MOVL    @W^PFN$AL_PTE[R0],R1 ;GET MASTER PTE ADDRESS
1399
1400      ASSUME  PFN$V_DELCON EQ PFN$V_LOC+PFN$$_LOC+1 ;DELCON IS 2ND BIT TO LEFT OF
1401                ;BIT IN BETWEEN IS FOR LOC EXPANSION
1402      INSV    #PFN$C_ACTIVE,#PFN$V_LOC,#PFN$$_LOC+2,- ;SET PAGE ACTIVE
1403                @W^PFN$AB_STATE[R0] ;AND CLEAR DELCON
1404      BISB    #<PTE$M_VALID@-24>,3(R1) ;SET VALID BIT
1405      CMPB    R2,#WSL$C_GLOBAL      ;GLOBAL OR PAGE TABLE PAGE?
1406      BGEQ    100$                  ;BRANCH IF YES
1407 60$:      UNLOCK  LOCKNAME=MMG,-   ;RELEASE MMG LOCK/PRESERVE IPL
1408                PRESERVE=NO        ;DON'T PRESERVE R0
1409      MOVQ    (SP)+,R0              ;RESTORE THE REGISTERS
1410      MOVQ    (SP)+,R2              ;
1411      MOVQ    (SP)+,R4              ;
1412      ADDL    #8,SP                 ;CLEAN OFF THE EXCEPTION PARAMETERS
1413      REIMAC                          ;AND RETURN FROM FAULT
1414 ;
1415 ; GLOBAL PAGE OR PAGE TABLE PAGE
1416 ;
1417 100$:     CMPB    R2,#WSL$C_PPGTBL ;PROCESS PAGE TABLE PAGE?
1418      BGEQ    120$                  ;BRANCH IF PROCESS OR GLOBAL PAGE TABLE
1419 ;
1420 ; GLOBAL PAGE
1421 ;
1422      BICL3   #^C<PTE$M_VALID ! PTE$M_MODIFY ! PTE$M_PFN>,(R1),R1 ;MASTER PTE
1423 ;
1424 ; R1 = VALID AND PFN BITS TO STORE INTO SLAVE PTE
1425 ; R3 = SLAVE PTE ADDRESS
1426 ;
1427 SETSLAVEPTE:
1428      BICL3   #^C<PTE$M_PROT ! PTE$M_OWN>,(R3),R2 ;PROTECTION AND OWNER FROM SLAVE
1429      BISL3   R1,R2,(R3)            ;STORE THE NEW SLAVE PTE
1430      BRB     60$                    ;AND EXIT THROUGH COMMON CODE
1431 ;
1432 ; PROCESS OR GLOBAL PAGE TABLE
1433 ;
1434 120$:     BGTR    60$                ;BRANCH IF GLOBAL PAGE TABLE
1435      PUSHL   R1                      ;SAVE REGISTER AROUND THE CALL
1436      BSBW    MMG$DECPHDREF          ;ONE LESS LOCK ON HDR SPTE
1437      POPL    R1                      ;RESTORE REGISTER

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 33

X-39 FREE, MODIFIED, OR BAD PAGE LIST, RELEAS 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```

1438         BRB      60$                ;AND EXIT THROUGH COMMON CODE
1439
1440         .DSABL  LSB
1441 ;
1442 ; FAILED TO READ THE DESIRED PAGE, RELEASE THE PFN AND ISSUE AN EXCEPTION
1443 ; R0      = PAGE FRAME NUMBER
1444 ; R4      = PROCESS CONTROL BLOCK ADDRESS
1445 ; R5      = SYSTEM ADDRESS OF PROCESS HEADER
1446 ; 0(SP)  = VIRTUAL ADDRESS (LOW BITS = PAGE TYPE)
1447 ; 4(SP)  = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE ENTRY
1448 ;
1449 READERR:
1450         MOVQ     (SP)+,R2                ;R2=VA (LOW BITS = PAGTYP), R3=SVAPTE
1451         PFN REFERENCE -
1452         MOVZWL  <@W^PFN$AX WSLX[R0],R1>,- ;GET WORKING SET LIST INDEX IF NOT G
1453         LONG_OPCODE=MOVL,-
1454         IMAGE=SYS_NONPAGED
1455
1456         ASSUME   PFN$C_PROCESS EQ 0
1457         ASSUME   PFN$C_SYSTEM EQ 1
1458         ASSUME   PFN$C_GLOBAL EQ 2
1459         ASSUME   PFN$C_GBLWRT EQ 3
1460         ASSUME   PFN$C_PPGTBL EQ 4
1461         ASSUME   PFN$C_GPGTBL EQ 5
1462         CMPV    #WSL$V_PAGTYP,#WSL$S_PAGTYP,R2,#PFN$C_GLOBAL ;IS PAGE GLOBAL?
1463         BLSS   20$                ;BRANCH IF NOT
1464 ;
1465 ; GLOBAL PAGE DOES NOT HAVE WORKING SET LIST INDEX IN WSLX ARRAY
1466 ; MUST SCAN THE PROCESS' WORKING SET LIST FOR THE VIRTUAL ADDRESS
1467 ;
1468         SUBL    R5,R3                ;UNBIAS SVAPTE, COULD BE SWAPPED HERE
1469         MOVL   G^CTL$GL PHD,R5        ;USE P1 SPACE HEADER WINDOW
1470         JSB    G^MMG$SCNWSLX         ;SCAN FOR THE WORKING SET LIST INDEX
1471 ;
1472 ; COULD HAVE BEEN SWAPPED IN THE ABOVE ROUTINE, BUT IPL IS BACK AT SYNCH NOW
1473 ;
1474         MOVL   PCB$S_L_PHD(R4),R5     ;RECOVER SYSTEM ADDRESS OF PHD
1475         ADDL   R5,R3                ;REBIAS PTE ADDRESS
1476         TSTL   R1                    ;SEE IF FOUND WORKING SET LIST ENTRY
1477         BEQL   40$                ;IF NOT, PAGE WENT AWAY OR SOME OTHER
1478         ;PROCESS WAS THE ORIGINATOR OF THE I/O
1479 20$:     MOVL   (R5)[R1],R2         ;FETCH WORKING SET LIST ENTRY
1480         MOVQ   R2,-(SP)              ;SAVE VIRTUAL ADDRESS AND PTE ADDRESS
1481         BSBW   MMG$FREWSLX           ;FREE THIS WORKING SET LIST ENTRY
1482         MOVQ   (SP)+,R2              ;RECOVER VA AND SVAPTE
1483         BLBC   R0,39$                ;BRANCH IF ERROR
1484 ;
1485 ; IF THIS PAGE FAULT IS FROM USER OR SUPER MODE THEN ISSUE A
1486 ; PAGE READ ERROR EXCEPTION.
1487 ;
1488 30$:     CMPZV  #PSL$V_CURMOD,#PSL$S_CURMOD,- ;IF FAULTING MODE IS
1489         FLTPSL(SP),#PSL$C_SUPER ;USER OR SUPER
1490         BGEQ   40$                ;THEN PAGE READ ERROR EXCEPTION
1491 ;
1492 ; THIS IS A BAD SITUATION NOW, AN EXCEPTION IN EXEC OR KERNEL MODE WILL
1493 ; CRASH THE SYSTEM. IF THIS PAGE IS OWNED BY USER OR SUPER THEN TRY
1494 ; SUBSTITUTING A PAGE OF ZEROS. THIS SHOULD SATISFY THE SYSTEM CODE WHICH

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 34

X-39 FREE, MODIFIED, OR BAD PAGE LIST, RELEAS 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```

1495 ; IS ACCESSING THE PROCESS PAGE SINCE IT IS PARANOID ABOUT USER SUPPLIED
1496 ; DATA. THE NEW PAGE WILL BE EXEC READ WRITE BUT OWNED BY THE ORIGINAL
1497 ; OWNER. THIS WILL RESULT IN AN ACCESS VIOLATION WHEN THE PAGE IS TOUCHED
1498 ; IN USER OR SUPER MODE.
1499 ;
1500 EXTZV #PTE$V_OWN,#PTE$S_OWN,(R3),R0 ;GET THE PAGE OWNER
1501 CMPL R0,#PSL$C_SUPER ;OWNED BY USER OR SUPER?
1502 BLSS 40$ ;BRANCH IF NOT, READ ERROR FOR
1503 ;A CRUCIAL PAGE, ISSUE THE PAGE
1504 ;READ ERROR EXCEPTION, DOWN WE GO.
1505 UNLOCK LOCKNAME=MMG,- ;UNLOCK MMG DATABASE
1506 NEWIPL=#IPL$ASTDEL ;LOWEST POSSIBLE FAULT IPL
1507 ;
1508 ; FORM ARGUMENT LIST FOR CRETVA
1509 ;
1510 PUSHL R2 ;VIRTUAL ADDRESS TO CREATE
1511 PUSHL R2 ;ANOTHER COPY TO FORM RANGE
1512 PUSHL R0 ;SAVE ACCESS MODE PARAMETER
1513 MOVB S^#PRT$C_EW,1(SP) ;SET DESIRED PAGE PROTECTION
1514 PUSHL #0 ;NULL RETURN ADDRESS
1515 PUSHAL 8(SP) ;ADDRESS OF RANGE TO CREATE
1516 CALLS #5,G^MMG$CRETVA ;KERNEL MODE ENTRY TO CRETVA
1517 ;PRESERVES IPL
1518 ;STRIP OFF INPUT RANGE WHEN DONE
1519 BRW PGFCOMPLETE2 ;FAULT THIS PAGE FROM SCRATCH
1520 39$:
1521 ;
1522 ; FREWSLX CAN ONLY FAIL IF PAGE FILE NEEDED TO BE ALLOCATED AND IT COULDN'T BE
1523 ; THIS CASE IS NOT POSSIBLE HERE.
1524 ;
1525 BUG_CHECK FREWSLX,FATAL
1526
1527 40$: INCW FLTCTL+2(SP) ;INDICATE PAGE READ ERROR
1528 ACVIOLAT:
1529 MOVZWL FLTPSL+2(SP),R1 ;GET IPL FROM FAULT PSL
1530 UNLOCK LOCKNAME=MMG,- ;RELEASE MMG LOCK
1531 NEWIPL=R1,- ; RESTORE IPL
1532 PRESERVE=NO ; DON'T PRESERVE R0
1533 POPR #^M<R0,R1,R2,R3,R4,R5> ;RESTORE REGISTERS SAVED BY PAGE FAULT
1534 BBCC #16,(SP),10$ ;BRANCH IF ACCESS VIOLATION
1535 JMP G^EXE$PAGRDERR ;ISSUE THE EXCEPTION
1536 10$: JMP G^EXE$ACVIOLAT ;ACCESS VIOATION

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 35

X-39 SCANDEADPT - SCAN A DEAD PAGE TABLE FOR 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```

1538      .SBTTL  SCANDEADPT - SCAN A DEAD PAGE TABLE FOR TRANSITION PAGES
1539 ;
1540 ; INPUTS:
1541 ;
1542 ;      R2 = VIRTUAL ADDRESS OF PAGE TABLE (LOW BITS = PAGE TYPE)
1543 ;      R5 = PROCESS HEADER ADDRESS
1544 ;      IPL = SYNCH
1545 ;
1546 ; OUTPUTS:
1547 ;
1548 ;      NONE
1549 ;
1550      .ENABL  LSB
1551 BRW_50$:
1552      BRW      50$              ;LONG BRANCH TO 50$
1553
1554 SCANDEADPT:
1555      SUBW3    PHD$W_PTCNTVAL(R5),PHD$W_PTCNTACT(R5),R0 ;ACTIVE PAGE TABLES
1556                      ;THAT DON'T CONTAIN VALID WSLE'S
1557                      ;ARE DEAD PAGE TABLES
1558      BLEQ     BRW_50$          ;BR IF NO DEAD PAGE TABLES
1559      ADDW     PHD$W_PTCNTLCK(R5),R0 ;ADD IN THE LOCKED PAGE TABLES
1560      MOVL     PHD$L_WSFUID(R5),-(SP) ; Get ready to add in fluid
1561      ADDL     (SP),R0          ;NEED TWICE FLUID EXTRA
1562      ADDL     (SP)+,R0
1563      MOVL     PHD$L_WSSIZE(R5),-(SP) ;GET CURRENT WS SIZE
1564      SUBL     (SP),R0          ;SUBTRACT OUT CURRENT WSL SIZE
1565      SUBL3    PHD$L_WSLIST(R5),PHD$L_WSDYN(R5),(SP) ;GET LOCKED PORTION OF WS
1566      ADDL     (SP)+,R0          ;ADD IN LOCKED ENTRY COUNT
1567      BLSS    50$              ;BRANCH IF SAFE TO POSTPONE DEAD PAGE
1568                      ;TABLE SCAN
1569      SUBL3    PHD$L_POBR(R5),R2,R0 ;BYTE OFFSET FROM FIRST PO PAGE TABLE
1570      BLSS    50$              ;BRANCH IF PROCESS HEADER PAGE
1571      ASHL     #-9,R0,R0        ;FORM PAGE NUMBER
1572      ADDL     R5,R0            ;ADD IN PHD BASE
1573      ADDL     PHD$L_PTWSLEVAL(R5),R0 ;ADD IN OFFSET TO BYTE ARRAY OF COUNTS
1574                      ;OF VALID WSLE'S IN EACH PAGE TABLE
1575      TSTB     (R0)              ;IS THIS A DEAD PAGE TABLE
1576      BGEQ    50$              ;BRANCH IF NOT
1577 ;
1578 ; R1,R2,R3 ARE PRESERVED UP TO THIS POINT
1579 ;
1580      MOVL     R1,PHD$L_WSNEXT(R5) ;UPDATE NEXT POINTER
1581      INCL     G^PMS$GL_DPTSCN    ;COUNT THESE SCANS
1582      EXTZV    #VA$V_VPN,#VA$$_VPN,R2,R0 ;PAGE NUMBER OF THE PT IN SYSTEM SPACE
1583      BICL3    #^C<PTE$M_VALID ! PTE$M_PFN>,- ;GET PFN AND VALID BIT
1584      @W^MMG$GL_SPTBASE[R0],R0    ;FROM SPT ENTRY FOR THE PAGE TABLE
1585      BBCC     #PTE$V_VALID,R0,40$ ;CLEAR VALID BRANCH IF IT WAS CLEAR
1586      PUSHL   #0                ;FLAG FOR MODIFIED PAGE WRITER NEEDED
1587                      PFN REFERENCE -
1588      MOVZWL   <@W^PFN$AX_SHRCNT[R0],-(SP) >,- ;NUMBER OF TRANSITION PAGES
1589                      LONG_OPCODE=MOVL,-
1590                      IMAGE=SYS_NONPAGED
1591      BEQL     40$                ;IF NONE, INCONSISTENT
1592      BICW     #VA$M_BYTE,R2      ;START SCANNING PT AT BEGINNING
1593      MOVZBL   #128,R3           ;AT MOST 128 PTE'S
1594 20$:      BICL3    #^C<PTE$M_VALID ! - ;GET THE VALID BIT

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 36  
X-39 SCANDEADPT - SCAN A DEAD PAGE TABLE FOR 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```

1595          PTE$M_TYP1 ! PTE$M_TYPO ! - ;PTE TYPE BITS
1596          PTE$M_PFN>, (R2)+, R0 ;AND THE PFN FROM THE PTE
1597          BEQL      30$ ;BRANCH IF DEMAND ZERO PAGE
1598          ASHL      #-PTE$V_TYPO, R0, R1 ;VALID, TYPO1, TYPO ALL 0 IF TRANSITION
1599          BEQL      60$ ;BRANCH IF TRANSITION PAGE
1600 30$:      SOBGTR   R3, 20$ ;LOOP THROUGH THE PAGE TABLE
1601          BRB       100$ ;ALL DONE, CNT=# I/O REQ OUTSTANDING
1602 40$:      BUG_CHECK SCANDEADPT, FATAL ;SPT ENTRY FOR PAGE TABLE NOT VALID
1603
1604 50$:      MOVL      #1, R0 ;SET CONTINUE RATHER THAN RESTART
1605          RSB
1606
1607          ;SHRCNT FOR PAGE TABLE IS 0
1608          ;DIDN'T FIND SHRCNT TRANSITION PAGES
1609          ;BEFORE RUNNING OFF THE END OF THE PT
1609 ;
1610 ; THIS IS A TRANSITION PAGE
1611 ;
1612 60$:      MOVQ      R2, -(SP) ;SAVE THESE REGISTERS
1613          EXTZV     #PFN$V_PAGTYP, #PFN$S_PAGTYP, - ;GET PAGE LOCATION
1614          @W^PFN$AB_STATE[R0], R2 ;FROM THE STATE BYTE
1615          CMLP      R2, #PFN$C_MFYPAGLST ;ON MODIFIED OR FREE PAGE LIST
1616          BGTR      90$ ;BRANCH IF NOT ON EITHER
1617          BEQL      80$ ;BRANCH IF ON MODIFIED PAGE LIST
1618 ;
1619 ; PAGE IS ON THE FREE PAGE LIST
1620 ;
1621          BSBW      MMG$REMPFN ;ON FREE LIST, REMOVE IT
1622          BISB      #PFN$M_DELCON, @W^PFN$AB_STATE[R0] ;FORCE DELETE CONTENTS
1623          BSBW      MMG$RELPFN ;AND RELEASE THE PAGE
1624          BRB       90$
1625 ;
1626 ; PAGE IS ON MODIFIED PAGE LIST
1627 ;
1628 80$:      BISB      #PFN$M_DELCON, @W^PFN$AB_STATE[R0] ;DELETE CONTENTS AFTER WRITING
1629          INCL      12(SP) ; Count another modified page
1630          BSBW      MMG$REMPFN ; Remove page from modified list
1631          BSBW      MMG$INSPFNH ; Insert at head of modified list
1632 90$:      MOVQ      (SP)+, R2 ;RESTORE SAVE REGISTERS
1633          SOBGTR   (SP), 30$ ;COUNT DOWN THE TRANSITION COUNT
1634 100$:     MOVQ      (SP)+, R0 ;CLEAN OFF THE EXHAUSTED COUNT AND FLAG
1635          TSTL      R1 ;R1 = count of modified pages found
1636          BEQL      110$ ;BRANCH IF NO MODIFIED PAGE WRITING
1637          MOVL      #MPW$C_SVAPTE, R0 ; Range-based purge request
1638          MOVAL     -128(R2), R1 ; Low SVAPTE
1639          MOVAL     -4(R2), R2 ; High SVAPTE (inclusive range)
1640          JSB       G^MMG$PURGEMPL ; Initiate MPL purge
1641          TSTL      G^MMG$GL_MAXPFIDX ; Any page files installed ?
1642          BEQL      110$ ; None - don't wait the process
1643          MOVZWL   #RSN$_MPWBUSY, R1 ;SET RESOURCE TO WAIT FOR
1644          ADDL      #4, SP ;RETURN TO ORIGINAL CALLER
1645 110$:     CLRL      R0 ;SET FAILURE (OR RESTART)
1646          RSB
1647
1648          .DSABL   LSB

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 37  
X-39 WSLEPFN - FETCH PFN FROM WORKING SET LIS 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```
1650      .SBTTL  WSLEPFN - FETCH PFN FROM WORKING SET LIST ENTRY
1651 ;
1652 ; CALLING SEQUENCE:
1653 ;
1654 ;      JSB      G^MMG$WSLEPFN
1655 ;
1656 ; INPUTS:
1657 ;
1658 ;      R3 = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE ENTRY
1659 ;           FOR A PAGE THAT IS IN THE WORKING SET LIST
1660 ;
1661 ; OUTPUTS:
1662 ;
1663 ;      R0 = PFN
1664 ;      R2,R3 PRESERVED
1665 ;
1666 WSLEPFNMSK:
1667      .LONG   ^C<PTE$M_VALID ! PTE$M_TYPO ! PTE$M_TYP1 ! PTE$M_BAKK>
1668
1669      UNIVERSAL_SYMBOL      MMG$WSLEPFN
1670 ;MMG$WSLEPFN::
1671      BICL3   B^WSLEPFNMSK,(R3),R0      ;GET VALID, TYPO, TYP1, PFN/GPTX
1672      ASHL   #-PTE$V_TYPO,R0,R1      ;SEE IF TRANSITION OR VALID PAGE
1673      BGTR   FRE_GBLTRANS              ;BRANCH IF NEITHER
1674      ASSUME PTE$V_PFN EQ 0
1675      BICL   #^C<PTE$M_PFN>,R0        ;GET PAGE FRAME NUMBER
1676      RSB                    ;AND RETURN
1677 ;
1678 ; INPUTS:
1679 ;
1680 ;      R0 = PAGE TABLE ENTRY WITH TYPO, TYP1 AND GPTX BITS
1681 ;      R1 = RESULT OF SHIFTING R0 BY -PTE$V_TYPO
1682 ;           = 1 IF PAGE IS GLOBAL
1683 ;           > 1 IF PAGE TYPE IS INVALID FOR THIS CONTEXT
1684 ;
1685 ; OUTPUTS:
1686 ;
1687 ;      R0 = PFN IF GLOBAL PAGE IN TRANSITION
1688 ;      BUGCHK IF NOT
1689 ;
1690 FRE_GBLTRANS:
1691      SOBGTR  R1,WSLVANVAL              ;BRANCH IF NOT GLOBAL FORMAT
1692      BBCC   #PTE$V_TYPO,R0,WSLVANVAL ;CLEAR TYPO, MUST HAVE BEEN SET
1693      BICL3  B^WSLEPFNMSK,@W^MMG$GL_GPTBASE[R0],R0 ;FETCH MASTER PTE
1694      ASHL   #-PTE$V_TYPO,R0,R1      ;MAKE SURE THIS IS IN TRANSITION
1695      BNEQ   WSLVANVAL                ;BRANCH IF IT IS NOT
1696      RSB                    ;ELSE, R0 = PFN
1697 WSLVANVAL:
1698      BUG_CHECK WSLVANVAL,FATAL        ;WORKING SET LIST ENTRY VIRTUAL
1699                                     ;ADDRESS IS NOT VALID
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 38  
X-39 FREWSLE - FREE A WORKING SET LIST ENTRY 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```
1701      .SBTTL  FREWSLE - FREE A WORKING SET LIST ENTRY
1702 ;++
1703 ; FUNCTIONAL DESCRIPTION:
1704 ;
1705 ;     THIS ROUTINE CHOOSES A WORKING SET LIST ENTRY, RELEASES THE
1706 ; PAGE WHICH OCCUPIES IT (IF ANY), MARKS THE ENTRY AVAILABLE, AND
1707 ; LEAVES THE WSNEXT POINTER POINTING TO THE AVAILABLE ENTRY.
1708 ;     IN RELEASING A PAGE, IF ITS BACKING STORE ADDRESS IS A
1709 ; NOT YET ALLOCATED PAGING FILE ADDRESS, THEN A PAGING FILE VBN
1710 ; IS ALLOCATED AT THIS TIME.  IT IS POSSIBLE THAT NO VBN'S ARE AVAILABLE
1711 ; AND THUS THIS ROUTINE CAN RETURN UNSUCCESSFULLY.
1712 ;
1713 ; CALLING SEQUENCE:
1714 ;
1715 ;     JSB      G^MMG$FREWSLE
1716 ;
1717 ; INPUT PARAMETERS:
1718 ;
1719 ;     R4 = PCB ADDRESS
1720 ;     R5 = PROCESS HEADER ADDRESS - MAY BE P1 SPACE ADDRESS
1721 ;     IF WORKING WITH PROCESS WORKING SET LIST
1722 ;     IPL = SYNCH
1723 ;
1724 ; IMPLICIT INPUTS:
1725 ;     NONE
1726 ;
1727 ; OUTPUT PARAMETERS:
1728 ;
1729 ;     IF SUCCESSFUL
1730 ;         R0 LOW BIT IS SET
1731 ;     IF NOT SUCCESSFUL
1732 ;         R0 LOW BIT IS CLEAR AND
1733 ;         R1 = RESOURCE TO WAIT FOR (#RSN$_XXXXX)
1734 ;
1735 ; IMPLICIT OUTPUTS:
1736 ;
1737 ;     IF A WORKING SET ENTRY WAS FREED,  IT IS PLACED ON THE FREE LIST
1738 ;
1739 ; COMPLETION CODES:
1740 ;     NONE
1741 ;
1742 ; SIDE EFFECTS:
1743 ;     NONE
1744 ;
1745 ;--
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 39

X-39 FREWSLE - FREE A WORKING SET LIST ENTRY 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```

1747 ; FOR G^MMG$FREWSLX ENTRY POINT
1748 ;
1749 ; INPUTS:
1750 ;
1751 ;     R1 = WORKING SET LIST INDEX
1752 ;     R2 = VIRTUAL ADDRESS (LOW BITS = PAGE TYPE)
1753 ;     R3 = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE ENTRY
1754 ;     R4 = PROCESS CONTROL BLOCK ADDRESS
1755 ;     R5 = PROCESS HEADER ADDRESS
1756 ;     IPL = SYNCH
1757 ;
1758 ; OUTPUTS:
1759 ;
1760 ;     RO = STATUS
1761 ;     R1 = RESOURCE TO WAIT FOR IF NOT SUCCESSFUL
1762 ;
1763 ;
1764 ;
1765 ; FOUND AN EMPTY WORKING SET LIST ENTRY, CHECK WHETHER THERE IS A
1766 ; NEW PEAK WORKING SET SIZE AND WHETHER SWAP AREA NEEDS TO GROW.
1767 ;
1768 ; RO = GPGCNT+PPGCNT
1769 ;
1770 ;     .ENABLE LSB
1771 ;
1772 10$:   BBC      #PHD$V_WSPEAKCHK,PHD$W_FLAGS(R5),15$ ;BRANCH IF CANNOT BE
1773 ;                               ;ABOVE PREVIOUS PEAK WORKING SET SIZE
1774       CMLP    R0,G^CTL$GL_WSPEAK      ;ABOVE PREVIOUS RECORDED PEAK?
1775       BLSSU   15$                      ;BRANCH IF NOT
1776       ADDL3   #1,R0,G^CTL$GL_WSPEAK   ;YES, NEW PEAK INCLUDES THE PAGE
1777 ;                               ;ABOUT TO BE ADDED TO THE WORKING SET
1778 15$:   BBC      #PHD$V_IWSPEAKCK,PHD$W_FLAGS(R5),20$ ;BRANCH IF CANNOT BE
1779 ;                               ;ABOVE PREVIOUS PEAK WORKING SET SIZE
1780       CMLP    R0,G^CTL$GL_IWSPEAK     ;ABOVE PREVIOUS RECORDED PEAK?
1781       BLSSU   20$                      ;BRANCH IF NOT
1782       ADDL3   #1,R0,G^CTL$GL_IWSPEAK  ;YES, NEW PEAK INCLUDES THE PAGE
1783 ;                               ;ABOUT TO BE ADDED TO THE WORKING SET
1784 20$:   MOVL    R1,PHD$L_WSNEXT(R5)    ;UPDATE NEXT POINTER
1785       MOVZWL  #SS$_NORMAL,R0          ;SUCCESSFUL RETURN INDICATION
1786       RSB
1787 ;
1788 35$:   BRB      10$                    ;GET BRANCH DESTINATION TO REACH
1789 ;
1790 LCKWSLE_NOTPGTB:
1791       BUG_CHECK BADLCKWSLE,FATAL      ;LOCKED WORKING SET LIST ENTRY NOT PAGE TABL
1792 WSSIZEERR:
1793       BUG_CHECK WSSIZEERR,FATAL      ;VALUE OF WSSIZE IS SMALLER THAN IN USE
1794 ;
1795 ; THIS IS A LOCKED PAGE IN THE DYNAMIC PORTION OF THE WORKING SET LIST
1796 ; IT MUST BE A PAGE TABLE PAGE, NOTE THE CONDITION CODES ARE STILL SET
1797 ; FROM THE FETCH OF THE WORKING SET LIST ENTRY.
1798 ;
1799 50$:   BGEQ    LCKWSLE_NOTPGTB        ;BRANCH IF THIS IS NOT A PAGE TABLE
1800       BBS     #WSL$V_PAGTYP,R2,60$    ;BRANCH TO SKIP GLOBAL PAGE TABLE PAGES
1801       BSBW    SCANDEADPT              ;SCAN THE PAGE TABLE TO SEE IF DEAD
1802 ;                               ;IF SO RID IT OF TRANSITION PAGES.
1803       BLBS   R0,60$                  ;AND NOW GET A FREE WS LIST ENTRY

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 40  
X-39 FREWSLE - FREE A WORKING SET LIST ENTRY 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```

1804
1805          UNIVERSAL_SYMBOL          MMG$FREWSLE
1806 ;MMG$FREWSLE::
1807          MOVZWL  G^SGN$GW WSLMxSKP,R3          ;MAX NUMBER OF TB VALID ENTRIES TO SKIP
1808          MOVL   PHD$L_WSNEXT(R5),R1          ;INDEX TO NEXT CANDIDATE TO DISCARD
1809          TSTL   (R5)[R1]                    ;IS THIS ENTRY FREE?
1810          BEQL   80$                          ;BRANCH IF SO, CHECK FOR TRULY FREE
1811 60$:      INCL   R1                          ;STEP TO NEXT ENTRY
1812          CMPL   R1,PHD$L_WSLAST(R5)          ;AT THE END YET?
1813          BLEQU  70$                          ;CONTINUE
1814          MOVL   PHD$L_WSDYN(R5),R1          ;BACK TO THE TOP
1815 70$:      MOVL   (R5)[R1],R2                ;R2 = VA FROM WSLE
1816          BNEQ  120$                          ;BRANCH IF ENTRY IN USE
1817 80$:      ADDL3  PCB$L_PPGCNT(R4),PCB$L_GPGCNT(R4),R0 ;CURRENT PAGE COUNT IN USE
1818          CMPL   R0,PHD$L_WSSIZE(R5)          ;ARE PAGES IN USE = WORKING SET SIZE?
1819          BEQL   90$                          ;BRANCH IF SO, NEED TO REPLACE A PAGE
1820          BGTRU  WSSIZEERR                    ;BRANCH IF MORE PAGES IN WS THAN IN WSSIZE
1821          SUBL3  PHD$L_WSLIST(R5),PHD$L_WSQUOTA(R5),R2 ;QUOTA NUMBER OF PAGES-
1
1822          CMPL   R0,R2                        ;ARE WE WITHIN QUOTA NUMBER OF PAGES?
1823          BLEQU  35$                          ;BRANCH IF SO, ALLOWED ANOTHER PAGE
1824          CMPL   G^SCH$GL_GROWLIM,G^SCH$GL_FREECNT ;ENOUGH FREE PAGES TO EXTEND?
1825          BLSS  35$                          ;BR IF SO
1826 90$:      SUBL3  #1,R1,R0                    ;SAVE INDEX OF LAST NON-ZERO WSLE
1827 100$:     INCL   R1                          ;STEP TO NEXT ENTRY
1828          CMPL   R1,PHD$L_WSLAST(R5)          ;AT THE END YET?
1829          BLEQU  110$                         ;CONTINUE
1830          MOVL   PHD$L_WSDYN(R5),R1          ;BACK TO THE TOP
1831          BLBS  G^MMG$GB_FREWFLGS,110$        ;IF SWAPPER REQUESTED, DON'T MOVE LAST
1832          MOVL   R0,PHD$L_WSLAST(R5)          ;SHRINK WSLAST BACK OVER 0 WSLE'S
1833          ;THIS IS SAFE BECAUSE WS IS FULL
1834 110$:     MOVL   (R5)[R1],R2                ;R2 = VA FROM WSLE
1835          BEQL   100$                         ;BRANCH IF UNUSABLE FREE ENTRY
1836 120$:     BLBC  R2,WSLENOVAL                ;BRANCH IF ENTRY NOT VALID
1837          BBS   #WSL$V_WSLOCK,R2,50$        ;SKIP ENTRY IF IT IS LOCKED
1838          ;CONDITION CODES STILL SET FROM LOAD OF R2
1839          BBC    S^#EXE$V_TBCHK,G^EXE$GL_FLAGS,130$ ;BRANCH IF TBCHK NOT ENABLED
1840          MTPR  R2,#PR$_TBCHK                ;TEST FOR VALID IN TB
1841          BVC   130$                          ;BRANCH IF NO VALID TRANSLATION
1842          SOBGEQ R3,60$                       ;SKIP PAGE UNLESS COUNT EXHAUSTED
1843 130$:     MOVL   R1,PHD$L_WSNEXT(R5)          ;UPDATE NEXT POINTER
1844          BSBW  MMG$SVAPTECHK                ;RETURN R3 = SYS VA OF PAGE TABLE ENTRY
1845          ;OK FOR PROCESS PAGE TABLES AND
1846          ;PROCESS HEADER PAGES WITH PROCESS PCB ADR
1847          .DISABLE LSB
1848 ;
1849 ; R1 = WSLX, R2 = VA FROM WSLE, R3 = SVAPTE, R4 = PCB, R5 = PHD
1850 ;
1851          BBC    #PTE$V_MODIFY,(R3),MMG$FREWSLX ;BRANCH IF PAGE NOT MODIFIED
1852          ;IF ENTRY NOT VALID MODIFY=0 SO BRANCH
1853          CMPL   G^MPW$GL_WAITLIM,-
1854          G^SCH$GL_MFYCNT                    ; Above upper wait process threshold ?
1855          BLEQ  140$                          ; BR if yes
1856          BBC    S^#SCH$V_MPW,G^SCH$GB_SIP,-
1857          MMG$FREWSLX                        ; BR if modified page writer not active
1858          CMPL   G^MPW$GL_LOWAITLIM,-
1859          G^SCH$GL_MFYCNT                    ; Above lower wait process threshold ?
1860          BGTR  MMG$FREWSLX                  ; BR if no

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

```

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-
8 Page 41
X-39 FREWSLE - FREE A WORKING SET LIST ENTRY 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (
1861 140$:   MOVL    G^CTL$GL_PCB,R0           ; Get process PCB address (R4 may be SYSTEM'
1862         TSTW    PCB$W_MTXCNT(R0)         ; Does process hold any mutexes ?
1863         BNEQ    MMG$FREWSLX              ; BR if yes (avoids possible deadlocks)
1864         BBS     S^#MMG$V_NOWAIT,G^MMG$GB_FREWFLGS,MMG$FREWSLX
1865         ;BRANCH IF THIS IS SWAPPER
1866         TSTL    G^MMG$GL_MAXPFIDX         ; Any page files installed ?
1867         BEQL    MMG$FREWSLX              ; None - don't wait the process
1868         MOVZWL   #RSN$ _MPWBUSY,R1        ;R1 = RESOURCE TO WAIT FOR
1869         CLRL    R0                        ;RETURN FAILURE INDICATION
1870         RSB     R0                        ;RETURN RESOURCE TO WAIT FOR IN R1
1871
1872 WSLENOVAL:
1873         BUG_CHECK WSLENOVAL,FATAL         ;WSL ENTRY NOT VALID
1874
1875         .ENABLE LSB
1876
1877         UNIVERSAL_SYMBOL      MMG$FREWSLX
1878 ;MMG$FREWSLX::
1879         PUSHL   R1                ;SAVE WSLX FOR DELETE BY WSLX
1880
1881         ASSUME PTE$V_MODIFY EQ PTE$V_TYP1
1882         BICL3   #^C<PTE$M_VALID ! -       ;FETCH VALID BIT
1883         PTE$M_TYP1 ! PTE$M_TYP0 ! - ;PTE TYPE BITS
1884         PTE$M_GPTX>, (R3),R0           ;AND PFN/GPTX FROM PAGE TABLE ENTRY
1885         BBSC    #PTE$V_VALID,R0,10$      ;BRANCH IF PTE VALID
1886         ;CLEAR VALID BIT IN R0
1887 ;
1888 ; PAGE TABLE ENTRY NOT VALID,
1889 ; PAGE IN TRANSITION AND READINPROG OR
1890 ; GLOBAL PTE POINTING TO TRANSITION PTE
1891 ;
1892         ASHL    #-PTE$V_TYP0,R0,R1      ;IF NEITHER TYP1 OR TYP0 IS SET
1893         BEQL    30$                    ;BRANCH TO RELEASE PAGE
1894 ;
1895 ; THIS WORKING SET LIST ENTRY POINTED TO A PAGE WITH A PAGE TABLE ENTRY
1896 ; WHICH IS NEITHER VALID NOR IN TRANSITION. THIS PAGE MUST BE A GLOBAL
1897 ; PAGE ON THE WAY IN TO MEMORY. THE GLOBAL PTE MUST BE IN TRANSITION.
1898 ;
1899         BSBW    FRE_GBLTRANS             ;GET PFN IF PAGE IS GLOBAL TRANSITION
1900         BRB     30$                    ;RELEASE ACTIVE PAGE
1901
1902 10$:      INVALIDATE_TB   R2,-           ;INVALIDATE TRANSLATION BUFFER
1903         INST1=<MOVL (R3),R1>,-
1904         INST2=<BICB #<PTE$M_VALID!PTE$M_MODIFY>@-24,3(R3)>
1905
1906 ;++
1907 ; NB: Co-routine address + 2 LWs are on top of stack during execution
1908 ;--
1909
1910
1911         UNIVERSAL_SYMBOL      MMG$FRE_TRYSKIP
1912 ;MMG$FRE_TRYSKIP::
1913         ;HOOK RETURN LOCATION FOR MP SUPPORT

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 42  
X-39 FREWSLE - FREE A WORKING SET LIST ENTRY 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```

1914 ; RELEASE THIS WORKING SET LIST ENTRY
1915 ;
1916 30$:   MOVL    @W^PFN$AL_BAK[R0],R1    ;GET BACKING STORE ADDRESS (VBN)
1917       BBS     #PFN$V_MODIFY,@W^PFN$AB_STATE[R0],50$ ;BRANCH IF PAGE MODIFIED
1918       BITL    #PFN$M_PGFLVBN,R1      ;DOES PAGE HAVE BACKING STORE?
1919       BNEQ    70$                     ;BRANCH IF YES
1920 ;
1921 ; NULL PAGE FILE BACKING STORE ADDRESS, AND PAGE IS NOT MODIFIED
1922 ;
1923 NULLPGFL_NOMFY:
1924       BUG_CHECK MFYNULPGFL,FATAL
1925 ;
1926 ; NOW THAT WE HAVE A MODIFIED COPY OF THE PAGE IN MEMORY, NO NEED FOR OBSOLETE
1927 ; PAGE FILE COPY.
1928 ;
1929 50$:   PUSHL   R3                       ;SAVE SVAPTE
1930       BSBW    MMG$DALCBAKSTORE        ;RELEASE OLD PAGE FILE BACKING STORE
1931       POPL    R3                       ;RESTORE SVAPTE
1932 ;
1933 ; G^PFN$AL_BAK[R0] IS ALL SET UP, R0 = PFN, R2 = VA, R3 = SVAPTE (SLAVE IF GBL)
1934 ;
1935 70$:   ASSUME   PFN$C_PROCESS EQ 0
1936       ASSUME   PFN$C_SYSTEM EQ 1
1937       ASSUME   PFN$C_GLOBAL EQ 2
1938       ASSUME   PFN$C_GBLWRT EQ 3
1939       ASSUME   PFN$C_PPGTBL EQ 4
1940       ASSUME   PFN$C_GPGTBL EQ 5
1941
1942       CMPV     #WSL$V_PAGTYP,#WSL$S_PAGTYP,R2,#PFN$C_GLOBAL ;GLOBAL PAGE?
1943       BLSSU   130$                          ;BRANCH IF PROCESS OR SYSTEM
1944       BLSS    120$                          ;BRANCH IF PROCESS OR GLOBAL PAGE TABLE
1945 ;
1946 ; GLOBAL PAGE - MAKE SLAVE PTE INTO GLOBAL FORMAT
1947 ;
1948       SUBL3    G^MMG$GL_GPTBASE,@W^PFN$AL_PTE[R0],R1 ;BYTE INDEX TO GPTE
1949       ROTL     #32-2,R1,R1                  ;GLOBAL PAGE TABLE INDEX
1950       ASSUME   PTE$V_TYPO EQ PTE$S_GPTX ;TYPO ADJACENT TO GPTX FIELD
1951       BISL     #PTE$M_TYPO,R1              ;SET TYPO BIT FOR GLOBAL FORMAT
1952       BICL3    #<PTE$M_TYPO!PTE$M_GPTX>,- ;STORE GPTX + TYPO IN PTE
1953             (R3),-(SP)                    ;CHANGING FROM TRANSITION TO GLOBAL
1954       BISL3    R1,(SP)+,(R3)
1955       BSBW    MMG$DECPTREF                 ;SLAVE PTE NO LONGER LOCKED
1956       DECshr   GTR=150$,-                 ;ONE LESS SHARER, BRANCH IF STILL IN USE
1957             IMAGE_FLAG=SYS_NONPAGED
1958       MOVL     @W^PFN$AL_PTE[R0],R1        ;GET MASTER PTE ADR
1959       BICB     #<PTE$M_VALID ! PTE$M_MODIFY>@-24,3(R1) ;FORM TRANSITION PTE
1960       BRB     130$                         ;GO COUNT ONE LESS WSL REF
1961
1962 ;
1963 ; PROCESS OR GLOBAL PAGE TABLE
1964 ;
1965 120$:   CMPZV   #WSL$V_PAGTYP,#WSL$S_PAGTYP,R2,#PFN$C_PPGTBL ;PROCESS PAGE TABLE?
1966       BNEQ    130$                         ;BRANCH IF NO
1967       MOVZWL   PHD$W_PHVINDEXT(R5),R1     ;PROCESS HEADER VECTOR INDEX
1968       INCW     @PHV$GL_REFBAS[R1]        ;ADD A PROCESS HEADER REFERENCE
1969             ;WHEN PROCESS PAGE TABLE IS PUT
1970             INTO TRANSITION STATE

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 43

X-39 FREWSLE - FREE A WORKING SET LIST ENTRY 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```
1971 130$:   DECF  EQL=140$                ;COUNT ONE LESS WSL REF
1972 ;
1973 ; OTHER REFERENCES OUTSTANDING, COULD BE DIRECT I/O, PAGING I/O
1974 ; IF CURRENT PAGE STATE IS ACTIVE CHANGE IT TO RELEASE PENDING WHICH
1975 ; IF REFAULTED WILL BE TRANSFORMED BACK TO ACTIVE.
1976 ; LEAVE READ IN PROGRESS STATE AS IS.
1977 ;
1978         CMPZV  #PFN$V_LOC,#PFN$$_LOC,@W^PFN$AB_STATE[R0],#PFN$C_ACTIVE
1979                                     ;UNLESS STATE IS ACTIVE
1980         BNEQ   150$
1981         INSV   #PFN$C_RELPEND,#PFN$V_LOC,#PFN$$_LOC,@W^PFN$AB_STATE[R0]
1982                                     ;OTHERWISE SET RELEASE PENDING STATE
1983         BRB    150$
1984 ;
1985 ; R0 = PFN, REFCNT = 0, R2 = VA, R3 = SVAPTE
1986 ;
1987
1988 140$:   BSBW   MMG$RELPFN                ;REFCNT = 0, RELEASE PFN
1989 150$:   MOVL   (SP)+,R1                 ;RECOVER SAVED WSLX
1990         BSBB   MMG$DELWSLEX            ;DELETE WORKING SET LIST ENTRY (BY INDEX)
1991         MOVZWL #SS$_NORMAL,R0          ;SUCCESSFUL RETURN INDICATION
1992         RSB
1993
1994         .DISABLE LSB
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 44  
X-39 DELWSLEX - DELETE WORKING SET LIST ENTRY 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```
1996      .SBTTL DELWSLEX - DELETE WORKING SET LIST ENTRY BY INDEX
1997 ;++
1998 ; FUNCTIONAL DESCRIPTION:
1999 ;
2000 ;     THIS ROUTINE DELETES THE WORKING SET LIST ENTRY INDEXED BY
2001 ; R1, AND PLACES THE WORKING SET LIST ENTRY ON THE FREE LIST.
2002 ;
2003 ; CALLING SEQUENCE:
2004 ;
2005 ;     JSB      G^MMG$DELWSLEX
2006 ;     JSB      G^MMG$DELWSLEPPG
2007 ;
2008 ; INPUT PARAMETERS:
2009 ;
2010 ;     R1 = WORKING SET LIST INDEX
2011 ;     R2 = VIRTUAL ADDRESS IF ENTERING AT DELWSLEPPG
2012 ;     R4 = PROCESS CONTROL BLOCK ADDRESS
2013 ;     R5 = PROCESS HEADER ADDRESS
2014 ;
2015 ; IMPLICIT INPUTS:
2016 ;
2017 ;     NONE
2018 ;
2019 ; OUTPUT PARAMETERS:
2020 ;
2021 ;     R0 PRESERVED
2022 ;
2023 ; IMPLICIT OUTPUTS:
2024 ;
2025 ;     NONE
2026 ;
2027 ; COMPLETION CODES:
2028 ;
2029 ;     NONE
2030 ;
2031 ; SIDE EFFECTS:
2032 ;
2033 ;     NONE
2034 ;
2035 ;--
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 45

X-39 DELWSLEX - DELETE WORKING SET LIST ENTRY 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```
2037      .ENABL  LSB
2038
2039      UNIVERSAL_SYMBOL      MMG$DELWSLEX
2040 ;MMG$DELWSLEX::
2041      MOVL      (R5) [R1], R2      ;FETCH WORKING SET LIST ENTRY
2042      EXTZV     #WSL$V_PAGTYP, #WSL$$_PAGTYP, R2, R3 ;GET THE PAGE TYPE
2043      BEQL      30$      ;BRANCH IF PROCESS PAGE
2044      CASE      R3, <-      ;AND DISPATCH ON IT
2045      30$, -      ;PROCESS PAGE
2046      45$, -      ;SYSTEM PAGE
2047      10$, -      ;GLOBAL READ ONLY
2048      10$, -      ;GLOBAL WRITABLE
2049      45$, -      ;PROCESS PAGE TABLE
2050      45$ >      ;GLOBAL PAGE TABLE
2051      BUG_CHECK DELWSLEX, FATAL ;BAD PAGE TYPE
2052 ;
2053 ; GLOBAL PAGE, READ ONLY OR WRITABLE
2054 ;
2055 10$:      BSBB      DECVALSLECNT      ;DEC VALID WORKING SET LIST ENTRY COUNT
2056      DECL      PCB$$_GPCNT (R4)      ;ONE LESS GLOBAL PAGE IN WORKING SET
2057      BRB      50$
2058 ;
2059 ; PROCESS PAGE, R2 = VIRTUAL ADDRESS
2060 ;
2061      UNIVERSAL_SYMBOL      MMG$DELWSLEPPG
2062 ;MMG$DELWSLEPPG::
2063 30$:      BSBB      DECVALSLECNT      ;DECREMENT VALID WORKING SET LIST ENTRY CNT
2064 45$:      DECL      PCB$$_PPCNT (R4)      ;ONE LESS PROCESS PAGE IN WORKING SET
2065 50$:      CLRL      (R5) [R1]      ;FREE THE WORKING SET LIST ENTRY
2066      RSB
2067
2068      .DSABL  LSB
2069
2070 DECVALSLECNT:
2071      ADDL3     PHD$$_PTWSLEVAL (R5), R5, R3 ;BASE ADR OF BYTE ARRAY OF COUNTS OF
2072      ;VALID WSLE'S IN EACH PAGE TABLE
2073      EXTV      #VA$V_VPN+7, #VA$$_VPN+1-7, R2, R2 ;BITS 16:30 OF VA SIGN EXTENDED
2074      BGEQ      10$      ;BRANCH IF P0 SPACE
2075      ADDL      G^SGN$GL_PTPAGCNT, R3      ;END ADDRESS OF BYTE ARRAY
2076 10$:      DECB      (R3) [R2]      ;ONE LESS VALID WSLE IN THIS PAGE TABLE
2077      BGEQ      20$      ;BRANCH IF PT STILL HAS OTHER VALID WSLE'S
2078      DECW      PHD$$_PTCNTVAL (R5)      ;ONE LESS PT WITH VALID WSLE'S
2079 20$:      RSB
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 46  
X-39 ININWPFN - ALLOCATE AND INIT A NEW PFN 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```
2081          .SBTTL  ININWPFN - ALLOCATE AND INIT A NEW PFN
2082 ;++
2083 ; FUNCTIONAL DESCRIPTION:
2084 ;
2085 ;     ALLOCATE A NEW PFN AND INITIALIZE THE PFN DATA BASE FOR IT
2086 ; AND MAKE A WORKING SET LIST ENTRY.
2087 ;
2088 ; CALLING SEQUENCE:
2089 ;
2090 ;     JSB      G^MMG$ININWPFN          ;ALLOCATE AND INIT NEW PFN
2091 ;
2092 ; INPUT PARAMETERS:
2093 ;
2094 ;     R2 = FAULT VA (LOW BITS = PAGTYP)
2095 ;     R3 = SVAPTE (SLAVE IF GLOBAL)
2096 ;     R4 = PCB ADDRESS (PROCESS IF GLOBAL)
2097 ;     R5 = PROCESS HEADER ADDRESS (PROCESS IF GLOBAL)
2098 ;
2099 ; IMPLICIT INPUTS:
2100 ;
2101 ;     NONE
2102 ;
2103 ; OUTPUT PARAMETERS:
2104 ;
2105 ;     R0 = PFN, OR NEGATIVE IF NONE AVAILABLE
2106 ;
2107 ; IMPLICIT OUTPUTS:
2108 ;
2109 ;     NONE
2110 ;
2111 ; COMPLETION CODES:
2112 ;
2113 ;     NONE
2114 ;
2115 ; SIDE EFFECTS:
2116 ;
2117 ;     NONE
2118 ;
2119 ;--
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 47

X-39 ININEWPFN - ALLOCATE AND INIT A NEW PFN 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (

```
2121 ININEWPFNWAIT:
2122     POPR     #^M<R2,R3,R5>           ;RESTORE SAVED REGISTERS
2123     RSB      ;AND RETURN NO FREE PAGES INDICATION
2124
2125     UNIVERSAL_SYMBOL      MMG$ININEWPFN
2126 ;MMG$ININEWPFN::
2127     PUSHL    R5             ;SAVE PHD
2128     PUSHL    R3             ;SAVE SVAPTE
2129     PUSHL    R2             ;SAVE VA
2130     BSBW     MMG$ALLOC PFN    ;ALLOCATE A NEW PFN
2131     BBS      #31,R0, ININEWPFNWAIT ;BRANCH IF NONE AVAILABLE
2132     MOVQ     (SP),R2        ;R2=VA, R3=SVAPTE
2133     CMPV     #WLS$V_PAGTYP,#WLS$S_PAGTYP,R2,#PFN$C_GLOBAL ;GLOBAL PAGE?
2134     BLSS     40$           ;BRANCH IF NOT
2135     ASSUME   PTE$V_GPTX EQ 0
2136     BICL3    #^C<PTE$M_GPTX>,(R3),R3 ;GLOBAL PAGE TABLE INDEX
2137     MOVAL    @W^MMG$GL_GPTBASE[R3],R3 ;SVAPTE OF MASTER
2138     MOVL     G^MMG$GL_SYSPHD,R5    ;SYSTEM PROCESS HEADER ADR FOR GLOBAL
2139 40$:     BBS      #VA$V_SYSTEM,R2,50$ ;DON'T COUNT PT REF FOR SYSTEM PAGE
2140     BSBW     MMG$INCPTREF    ;LOCK PAGE TABLE ENTRY (NOT FOR SYSTEM)
2141 50$:     MOVL     R3,@W^PFN$AL_PTE[R0] ;STORE PTE ADDRESS
2142     ASSUME   PFN$V_PAGTYP EQ 0
2143     ROTL     #<32-WLS$V_PAGTYP>,R2,R2 ;POSITION PAGE TYPE FIELD
2144     BICB3    #^C<PFN$M_PAGTYP>,R2,@W^PFN$AB_TYPE[R0] ;SET PAGE TYPE
2145     MOVQ     (SP)+,R2        ;R2=VA, R3=SVAPTE
2146     POPL     R5             ;R5=PHD
2147 ;
2148 ; FALL THROUGH TO G^MMG$MAKEWSLE
2149 ;
```



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 48

X-39 MAKEWSLE - MAKE A WORKING SET LIST ENTRY 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```
2151      .SBTTL  MAKEWSLE - MAKE A WORKING SET LIST ENTRY
2152 ;++
2153 ; FUNCTIONAL DESCRIPTION:
2154 ;
2155 ;     THIS ROUTINE ENTERS SPECIFIED VIRTUAL ADDRESS INTO THE WORKING
2156 ; SET LIST.  IT ASSUMES THAT THERE IS A FREE WORKING SET LIST ENTRY ON THE
2157 ; FREE LIST.  IF THE PAGE IS A GLOBAL PAGE THE SLAVE PAGE TABLE ENTRY IS
2158 ; LOCKED AT THIS TIME AND THE SHRCNT AND/OR REFCNT IS INCREMENTED.
2159 ; THIS ROUTINE ALSO KEEPS THE ACTIVE PAGE COUNTERS IN THE PCB (PPGCNT, GPGCNT).
2160 ;
2161 ;
2162 ; CALLING SEQUENCE:
2163 ;
2164 ;     JSB      G^MMG$MAKEWSLE
2165 ;
2166 ; INPUT PARAMETERS:
2167 ;
2168 ;     R0 = PAGE FRAME NUMBER
2169 ;     R2 = VA (LOW BITS = PAGTYP)
2170 ;     R3 = SVAPTE (SLAVE IF GLOBAL)
2171 ;     R4 = PCB ADDRESS (PROCESS IF GLOBAL)
2172 ;     R5 = PHD ADDRESS (PROCESS IF GLOBAL)
2173 ;
2174 ; IMPLICIT INPUTS:
2175 ;
2176 ;     FREE WORKING SET LIST CONTAINS AT LEAST ONE ENTRY
2177 ;
2178 ; OUTPUT PARAMETERS:
2179 ;
2180 ;     R0 = PFN PRESERVED
2181 ;
2182 ; IMPLICIT OUTPUTS:
2183 ;
2184 ;     NONE
2185 ;
2186 ; COMPLETION CODES:
2187 ;
2188 ;     NONE
2189 ;
2190 ; SIDE EFFECTS:
2191 ;
2192 ;     NONE
2193 ;
2194 ;--
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 49

X-39 MAKEWSLE - MAKE A WORKING SET LIST ENTRY 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```

2196          UNIVERSAL_SYMBOL          MMG$MAKEWSLE
2197 ;MMG$MAKEWSLE::
2198          MOVL          PHD$L_WSNEXT(R5),R1          ;WSLX FOR FREE ENTRY
2199
2200          ASSUME        WSL$V_VALID EQ 0
2201          BLBS          (R5)[R1],20$          ;BRANCH IF ENTRY BUSY, ERROR
2202          BISL3         #WSL$M_VALID,R2,(R5)[R1] ;STORE NEW WSLE
2203
2204          EXTZV         #WSL$V_PAGTYP,#WSL$$_PAGTYP,R2,R1 ;EXTRACT THE PAGE TYPE
2205          BEQL          50$          ;BRANCH IF PROCESS PAGE
2206          CASE          R1,<-          ;AND DISPATCH ON IT
2207          50$,-          ;PROCESS PAGE
2208          70$,-          ;SYSTEM PAGE
2209          30$,-          ;GLOBAL READ ONLY
2210          30$,-          ;GLOBAL WRITABLE
2211          70$,-          ;PROCESS PAGE TABLE
2212          70$ >          ;GLOBAL PAGE TABLE
2213 20$:          BUG_CHECK MAKEWSLE,FATAL          ;BAD PAGE TYPE OR
2214          ;WSNEXT POINTS TO VALID WSLE
2215 ;
2216 ; GLOBAL PAGE, READ ONLY OR WRITABLE
2217 ;
2218 30$:          BSBB          MMG$INCPTRF          ;LOCK THE SLAVE PAGE TABLE ENTRY
2219          BSBB          INCVALWSLECNT          ;INC VALID WORKING SET LIST ENTRY COUNT
2220          INCL          PCB$L_GPGCNT(R4)          ;ANOTHER GLOBAL PAGE IN WORKING SET LIST
2221          PFN_REFERENCE -
2222          ACBW          <#1,#1,@W^PFN$AX_SHRCNT[R0],80$>,-          ;COUNT SHARER, BRANCH IF FIR
2223          LONG_OPCODE=ACBL,-
2224          IMAGE=SYS_NONPAGED
2225          BRB          90$
2226 ;
2227 ; PROCESS PAGE
2228 ;
2229 50$:          BSBB          INCVALWSLECNT          ;INC VALID WORKING SET LIST ENTRY COUNT
2230 70$:          INCL          PCB$L_PPGCNT(R4)          ;ONE MORE ACTIVE PROCESS PAGE IN WSL
2231          PFN_REFERENCE -
2232          MOVW          <PHD$L_WSNEXT(R5),@W^PFN$AX_WSLX[R0]>,- ;SET INDEX TO WSLE
2233          LONG_OPCODE=MOVL,-
2234          IMAGE=SYS_NONPAGED
2235          ;ONLY FOR PRIVATE PAGES
2236
2237 80$:          INCW          @W^PFN$AW_REFcnt[R0]          ;ANOTHER REFERENCE FOR THE PAGE
2238 90$:          RSB
2239
2240 INCVALWSLECNT:
2241          ADDL3         PHD$L_PTWSLEVAL(R5),R5,R3 ;BASE ADR OF BYTE ARRAY OF COUNTS OF
2242          ;VALID WSLE'S IN EACH PAGE TABLE
2243          EXTV          #VA$V_VPN+7,#VA$$_VPN+1-7,R2,R2 ;BITS 16:30 OF VA SIGN EXTENDED
2244          BGEQ          10$          ;BRANCH IF P0 SPACE
2245          ADDL          G^SGN$GL_PTPAGCNT,R3          ;BASE ADR TO NEGATIVE INDEX FROM
2246 10$:          INCB          (R3)[R2]          ;ANOTHER VALID WSLE IN THIS PAGE TABLE
2247          BGTR          20$          ;BRANCH IF NOT THE FIRST
2248          INCW          PHD$W_PTCNTVAL(R5)          ;ANOTHER PAGE TABLE WITH VALID WSLE'S
2249 20$:          RSB

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 50

X-39 LOCKPGTB - LOCK PAGE TABLE 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (22)

```
2251      .SBTTL  LOCKPGTB - LOCK PAGE TABLE
2252 ;++
2253 ; FUNCTIONAL DESCRIPTION:
2254 ;
2255 ;     LOCKPGTB TAKES A VIRTUAL ADDRESS, REFERENCES AND
2256 ;     LOCKS THE ASSOCIATED PAGE TABLE, AND RETURNS THE SYSTEM
2257 ;     VIRTUAL ADDRESS OF THE PAGE TABLE ENTRY.  IT IS CALLED
2258 ;     WITH IPL = ASTDEL OR LOWER AND RETURNS WITH IPL = SYNCH.
2259 ;
2260 ; CALLING SEQUENCE:
2261 ;
2262 ;     JSB      G^MMG$LOCKPGTB
2263 ;
2264 ; INPUT PARAMETERS:
2265 ;
2266 ;     R2 = VIRTUAL ADDRESS
2267 ;     R4 = PROCESS CONTROL BLOCK ADDRESS
2268 ;     R5 = PROCESS HEADER ADDRESS (P1 SPACE IF PROCESS PCB,
2269 ;         SYSTEM SPACE OF SYSTEM PCB)
2270 ;     IPL = ASTDEL OR LOWER
2271 ;
2272 ; IMPLICIT INPUTS:
2273 ;
2274 ;     NONE
2275 ;
2276 ; OUTPUT PARAMETERS:
2277 ;
2278 ;     R2 PRESERVED
2279 ;     R3 = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE ENTRY
2280 ;     IPL = SYNCH
2281 ;
2282 ; IMPLICIT OUTPUTS:
2283 ;
2284 ;     PAGE TABLE LOCKED VIA INCPTRF
2285 ;
2286 ; COMPLETION CODES:
2287 ;
2288 ;     NONE
2289 ;
2290 ; SIDE EFFECTS:
2291 ;
2292 ;     NONE
2293 ;
2294 ;--
2295
2296      UNIVERSAL_SYMBOL      MMG$LOCKPGTB
2297 ;MMG$LOCKPGTB::
2298      BSBW      MMG$PTREF      ;REFERENCE PTE, GET SVAPTE
2299              ;RETURNS AT IPL=SYNCH
2300      BLBC      R0, INCPTRFBUG ;BRANCH IF LENGTH VIOLATION
2301 ;
2302 ; FALL THROUGH TO G^MMG$INCPTRF
2303 ;
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 51

X-39 INCPTRF - INCREMENT PAGE TABLE REFERENC 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```
2305      .SBTTL  INCPTRF - INCREMENT PAGE TABLE REFERENCE COUNT
2306 ;++
2307 ; FUNCTIONAL DESCRIPTION:
2308 ;
2309 ;     THIS ROUTINE ACCEPTS THE ADDRESS OF A PAGE TABLE ENTRY AND
2310 ; LOCKS THE ASSOCIATED PAGE TABLE INTO MEMORY.  IT ALSO MAINTAINS THE
2311 ; COUNT OF SUCH LOCKED PAGE TABLES IN THE PROCESS HEADER VECTOR.
2312 ;
2313 ; CALLING SEQUENCE:
2314 ;
2315 ;     JSB      G^MMG$INCPTRF
2316 ;
2317 ; INPUT PARAMETERS:
2318 ;
2319 ;     R3 = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE ENTRY (MASTER IF GLOBAL)
2320 ;     R5 = PROCESS HEADER ADDRESS (SYSTEM IF GLOBAL)
2321 ;
2322 ; IMPLICIT INPUTS:
2323 ;
2324 ;     NONE
2325 ;
2326 ; OUTPUT PARAMETERS:
2327 ;
2328 ;     R0,R2,R3 PRESERVED
2329 ;
2330 ; IMPLICIT OUTPUTS:
2331 ;
2332 ;     NONE
2333 ;
2334 ; COMPLETION CODES:
2335 ;
2336 ;     NONE
2337 ;
2338 ; SIDE EFFECTS:
2339 ;
2340 ;     NONE
2341 ;
2342 ;--
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 52

X-39 INCPTRF - INCREMENT PAGE TABLE REFERENC 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```

2344          UNIVERSAL_SYMBOL          MMG$INCPTRF
2345 ;MMG$INCPTRF::
2346          EXTZV    #VA$V_VPN,#VA$$_VPN,R3,R1 ;GET PAGE NUMBER OF PT CONTAINING THIS PTE
2347 ;
2348 ; ***** WARNING ***** THE FOLLOWING DEPENDS ON GPTBASE = SPTBASE
2349 ;
2350          MOVL     @W^MMG$GL_SPTBASE[R1],R1          ;PTE FOR PAGE TABLE
2351          BGEQ     20$                               ;DISASTER IF NOT VALID
2352          ASSUME   PTE$V_PFN EQ 0
2353          BICL     #^C<PTE$M_PFN>,R1                ;GET PAGE FRAME NUMBER
2354          CMPL    R1,G^MMG$GL_MAXPFN                ; IS THERE PFN DATABASE?
2355          BGTR     5$                               ; NO, SKIP INCREMENT
2356          PFN_REFERENCE -
2357          ACBW    <#1,#1,@W^PFN$AX_SHRCNT[R1],10$>,- ;INC SHRCNT, BRANCH IF FIRST
2358          LONG_OPCODE=ACBL,-
2359          IMAGE=SYS_NONPAGED
2360 5$:      RSB
2361 ;
2362 ; SHARE COUNT JUST WENT FROM 0 TO 1 INDICATING THAT THE FIRST ACTIVE
2363 ; PAGE TABLE ENTRY WAS JUST PLACED IN THE PAGE TABLE
2364 ;
2365 ; ASSUMPTION HERE IS THAT THIS ROUTINE IS NOT CALLED FOR SYSTEM PAGE TABLES
2366 ; THIS IS EITHER A PROCESS OR GLOBAL PAGE TABLE.
2367 ;
2368 10$:     PFN_REFERENCE -
2369          MOVZWL   <@W^PFN$AX_WSLX[R1],R1>,-          ;WORKING SET LIST INDEX
2370          LONG_OPCODE=MOVL,-
2371          IMAGE=SYS_NONPAGED
2372          BISL    #WSL$M_WSLock,(R5)[R1] ;SET WORKING SET LOCKDOWN BIT
2373          INCW    PHD$W_PTCNTACT(R5) ;ANOTHER ACTIVE PAGE TABLE
2374          MOVZWL   PHD$W_PHVINDEX(R5),R1 ;PROCESS HEADER VECTOR INDEX
2375          INCW    @PHV$GL_REFCBAS[R1] ;COUNT ANOTHER PAGE TABLE LOCKED
2376          RSB
2377 ;
2378 ; PAGE TABLE PAGE WAS NOT VALID
2379 ;
2380 20$:
2381 INCPTRFBUG:
2382          BUG_CHECK INCPTRF,FATAL ;PAGE TABLE NOT VALID
2383          ;LENGTH VIOLATION FROM LOCKPGTB

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 53  
X-39 DECPTRF - DECREMENT PAGE TABLE REFERENC 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```
2385      .SBTTL  DECPTRF - DECREMENT PAGE TABLE REFERENCE COUNT
2386 ;++
2387 ; FUNCTIONAL DESCRIPTION:
2388 ;
2389 ;      THIS ROUTINE DECREASES THE REFERENCE COUNT FOR THE PAGE TABLE
2390 ; CONTAINING THE PAGE TABLE ENTRY ADDRESSED BY R3.  IF THE RESULTING REFERENCE
2391 ; COUNT INDICATES THAT NO MORE PAGE TABLE ENTRIES ARE IN USE, THE PROCESS
2392 ; HEADER VECTOR REFERENCE COUNT IS DECREMENTED AS WELL INDICATING A
2393 ; FREE PAGE TABLE
2394 ;
2395 ; CALLING SEQUENCE:
2396 ;
2397 ;      JSB      G^MMG$DECPTRF
2398 ;
2399 ; INPUT PARAMETERS:
2400 ;
2401 ;      R3 = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE ENTRY
2402 ;
2403 ; IMPLICIT INPUTS:
2404 ;
2405 ;      NONE
2406 ;
2407 ; OUTPUT PARAMETERS:
2408 ;
2409 ;      R0,R2,R3 PRESERVED
2410 ;
2411 ; IMPLICIT OUTPUTS:
2412 ;
2413 ;      NONE
2414 ;
2415 ; COMPLETION CODES:
2416 ;
2417 ;      NONE
2418 ;
2419 ; SIDE EFFECTS:
2420 ;
2421 ;      NONE
2422 ;
2423 ;--
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 54

X-39 DECPTRF - DECREMENT PAGE TABLE REFERENC 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```

2425          UNIVERSAL_SYMBOL          MMG$DECPTRF
2426 ;MMG$DECPTRF::
2427          EXTZV  #VA$V_VPN,#VA$$_VPN,R3,R1 ;INDEX TO SPT ENTRY FOR PAGE TABLE
2428          MOVL   @W^MMG$GL_SPTBASE[R1],R1      ;PTE FOR PAGE TABLE
2429          BGEQ   40$                          ;BRANCH IF NOT VALID, ERROR
2430          ASSUME PTE$V_PFN EQ 0
2431          BICL   #^C<PTE$M_PFN>,R1            ;GET PAGE FRAME NUMBER FOR PAGE TABLE
2432          CMPL   R1,G^MMG$GL_MAXPFN          ;IS THERE PFN DATA BASE FOR THIS PAGE?
2433          BGTR   20$                          ;NO, SKIP DECREMENT
2434          DECshr PFN=R1,GTR=20$,-            ;ONE LESS ACTIVE PTE IN THIS PT
2435          IMAGE_FLAG=SYS_NONPAGED
2436 ;
2437 ; SHARE COUNT JUST WENT TO 0, THIS PAGE TABLE IS NO LONGER REQUIRED
2438 ; TO REMAIN RESIDENT
2439 ; R1 = PFN FOR PAGE TABLE PAGE
2440 ;
2441          PUSHL  R0                          ;SAVE THIS REGISTER
2442          PFN_REFERENCE -
2443          MOVZWL <@W^PFN$AX_WSLX[R1],R0>,-    ;USE IT TO HOLD THE WORKING SET LIST
2444          LONG_OPCODE=MOVL,-
2445          IMAGE=SYS_NONPAGED
2446          MOVL   G^MMG$GL_SYSPHD,R1          ;ADDRESS OF SYSTEM HEADER
2447          CMPL   R3,R1                        ;PTE ADR IN SYSTEM HEADER?
2448          BGEQU  10$                          ;BRANCH IF YES, GLOBAL PAGE TABLE
2449          SUBL3  G^SWP$GL_BALBASE,R3,R1      ;ADR RELATIVE TO BEGIN OF BAL SET
2450          DIVL  G^SWP$GL_BSLOTSZ,R1          ;PROCESS HEADER INDEX
2451          ASHL  #-9,R1,R1                     ;DIVIDE BY PAGE SIZE
2452          BSBB   MMG$DECPHDREF1              ;DECREMENT PROCESS HEADER REFERENCE COUNT
2453          MULL  G^SWP$GL_BSLOTSZ,R1          ;CONVERT PROCESS HEADER INDEX
2454          ROTL  #9,R1,R1                      ;MULL BY PAGE SIZE
2455          ADDL  G^SWP$GL_BALBASE,R1          ;TO PROCESS HEADER ADDRESS
2456 10$:      BICL  #WSL$M_WSLOCK,(R1)[R0]    ;SHUT OFF WORKING SET LOCK
2457          DECW  PHD$W_PTCNTACT(R1)          ;ONE LESS ACTIVE PAGE TABLE
2458          BLSS  40$                          ;BRANCH IF ERROR
2459          POPL  R0                          ;RESTORE SAVED REGISTER
2460 20$:      RSB                               ;AND RETURN
2461 ;
2462 ; PAGE TABLE PTE NOT VALID, OR PAGE TABLE REFERENCE COUNT IS BAD
2463 ; OR PTCNTACT WENT NEGATIVE
2464 ;
2465 40$:      BUG_CHECK DECPTRF,FATAL          ;ERROR IN DECPTRF

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 55

X-39 DECPHDREF - DECREMENT PROCESS HEADER REF 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```
2467      .SBTTL  DECPHDREF - DECREMENT PROCESS HEADER REFERENCE COUNT
2468 ;++
2469 ; FUNCTIONAL DESCRIPTION:
2470 ;
2471 ;     DECPHDREF REDUCES THE PROCESS HEADER REFERENCE COUNT AND INFORMS
2472 ;     THE SWAPPER IF THE COUNT GOES TO ZERO.  THIS COUNT IS RAISED ONCE
2473 ;     FOR EACH REASON THAT A GIVEN SPT ENTRY IS BUSY NOT COUNTING THE
2474 ;     WORKING SET LIST ENTRY REFERENCE.  THE FOLLOWING ARE REASONS WHY
2475 ;     THE REFERENCE COUNT IS INCREASED FOR A GIVEN PAGE TABLE PAGE.
2476 ;         1. PLACED ON THE FREE OR MODIFIED LIST
2477 ;         2. READ OR WRITE IN PROGRESS
2478 ;         3. SHARE COUNT IS ABOVE 0, I.E. IT CONTAINS ACTIVE PTE'S
2479 ;     THE REFERENCE COUNT IS DECREASED UNDER THE FOLLOWING CONDITIONS:
2480 ;         1. SHARE COUNT DECREASED FROM 1 TO 0, I.E. LAST PTE GONE
2481 ;         2. READ OR WRITE COMPLETE
2482 ;         3. PAGE CONTENTS DELETED (DELCONPFN)
2483 ;         4. FAULTED OUT OF TRANSITION STATE
2484 ;
2485 ; CALLING SEQUENCE:
2486 ;
2487 ;     JSB      MMG$DECPHDREF      ;R5 = PROCESS HEADER ADDRESS
2488 ;     JSB      MMG$DECPHDREF1    ;R1 = PROCESS HEADER VECTOR INDEX
2489 ;
2490 ; INPUT PARAMETERS:
2491 ;
2492 ;     DECPHDREF
2493 ;         R5 = PROCESS HEADER ADDRESS FOR PAGE TABLE PAGE
2494 ;
2495 ;     DECPHDREF1
2496 ;         R1 = PROCESS HEADER VECTOR INDEX
2497 ;
2498 ; IMPLICIT INPUTS:
2499 ;
2500 ;     NONE
2501 ;
2502 ; OUTPUT PARAMETERS:
2503 ;
2504 ;     DECPHDREF
2505 ;         ONLY R1 ALTERED
2506 ;
2507 ;     DECPHDREF1
2508 ;         ALL REGISTERS PRESERVED
2509 ;
2510 ; IMPLICIT OUTPUTS:
2511 ;
2512 ;     NONE
2513 ;
2514 ; COMPLETION CODES:
2515 ;
2516 ;     NONE
2517 ;
2518 ; SIDE EFFECTS:
2519 ;
2520 ;     NONE
2521 ;
2522 ;--
2523
```



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 56  
X-39 DECPHDREF - DECREMENT PROCESS HEADER REF 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```
2524     UNIVERSAL_SYMBOL      MMG$DECPHDREF
2525 ;MMG$DECPHDREF::
2526     MOVZWL  PHD$W_PHVINDEX(R5),R1  ;PROCESS HEADER VECTOR INDEX
2527     UNIVERSAL_SYMBOL      MMG$DECPHDREF1
2528 ;MMG$DECPHDREF1::
2529     DECW    @PHV$GL_REFCBAS[R1]    ;COUNT ONE LESS REFERENCE
2530     BEQL    10$                    ;BRANCH IF THAT WAS THE LAST REFERENCE
2531     RSB
2532 10$:   LOCK    LOCKNAME=SCHED      ;LOCK SCHED DATABASE
2533     JSB     G^SCH$SWPWAKE          ;INFORM THE SWAPPER, HEADER MAY GO
2534     UNLOCK  LOCKNAME=SCHED,-      ;UNLOCK SCHED DATABASE
2535     CONDITION=RESTORE            ;JUST RESTORE OWNERSHIP COUNT
2536     ; NO IPL CHANGE
2537     RSB
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 57

X-39 INIBLDPKT - INIT FOR CALLING BUILDPKT 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (25

```
2539      .SBTTL  INIBLDPKT - INIT FOR CALLING BUILDPKT
2540 ;++
2541 ; FUNCTIONAL DESCRIPTION:
2542 ;
2543 ;     THIS ROUTINE SETS UP R0-R2 FOR A SINGLE PAGE READ/WRITE
2544 ; TO THE ADDRESS SPECIFIED BY THE BACKING STORE ADDRESS.
2545 ;
2546 ; CALLING SEQUENCE:
2547 ;
2548 ;     JSB      G^MMG$INIBLDPKT
2549 ;
2550 ; INPUT PARAMETERS:
2551 ;
2552 ;     R2 = BACKING STORE ADDRESS
2553 ;     R3 = PAGE TABLE ENTRY ADDRESS (MASTER IF GLOBAL)
2554 ;     R5 = PROCESS HEADER ADDRESS (SYSTEM HEADER IF GLOBAL PAGE)
2555 ;     THIS IS ONLY USED FOR SECTION TYPE BACKING STORE ADDRESSES
2556 ; IF THE BACKING STORE ADDRESS IN R2 IS KNOWN TO BE A
2557 ; PAGING FILE ADDRESS, THEN IT SELF DESCRIBES AND THIS
2558 ; PARAMETER IS IGNORED.
2559 ;
2560 ; IMPLICIT INPUTS:
2561 ;
2562 ;     NONE
2563 ;
2564 ; OUTPUT PARAMETERS:
2565 ;
2566 ;     R0 = VIRTUAL BLOCK NUMBER
2567 ;     R1 = SECTION OR PAGE FILE CONTROL BLOCK ADDRESS
2568 ;     R2 = WINDOW ADDRESS
2569 ;     R3 = PAGE TABLE ENTRY ADDRESS (PRESERVED)
2570 ;
2571 ; IMPLICIT OUTPUTS:
2572 ;
2573 ;     NONE
2574 ;
2575 ; COMPLETION CODES:
2576 ;
2577 ;     NONE
2578 ;
2579 ; SIDE EFFECTS:
2580 ;
2581 ;     NONE
2582 ;
2583 ;--
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-  
8 Page 58  
X-39 INIBLDPKT - INIT FOR CALLING BUILDPKT 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1 (29

```

2585          UNIVERSAL_SYMBOL          MMG$INIBLDPKT
2586 ;MMG$INIBLDPKT::
2587          BBS      #PFN$V_GBLBAK,R2,10$      ;NOT AN I/O ADDRESS IF GBL BAK
2588          BBS      #PTE$V_TYPO,R2,20$        ;BRANCH IF SECTION ADDRESS
2589          EXTZV    #PFN$V_PGFLX,#PFN$S_PGFLX,R2,R1 ;PAGE FILE INDEX
2590          MOVL     @W^MMG$GL_PAGSWPVC[R1],R1 ;PAGE FILE CONTROL BLOCK ADDRESS
2591          ASSUME   PTE$V_PGFLVBN EQ 0
2592          BICL3    #^C<PTE$M_PGFLVBN>,R2,R0 ;PAGE FILE VBN
2593          BNEQ     40$                          ;BRANCH IF GOOD VBN
2594 ;
2595 ; INVALID BACKING STORE ADDRESS FOR I/O
2596 ;
2597 10$:      BUG_CHECK IVBAKADIO,FATAL          ;INVALID BACKING STORE ADR FOR I/O
2598 ;
2599 ; SECTION TABLE BACKING STORE ADDRESS
2600 ;
2601 20$:      CVTWL   R2,R2                      ;SECTION TABLE INDEX
2602          ADDL3    PHD$L_PSTBASOFF(R5),R5,R1 ;SECTION TABLE BASE ADDRESS
2603          MOVAL    (R1)[R2],R1                ;SECTION TABLE ENTRY ADDRESS
2604          SUBL3    PHD$L_POBR(R5),R3,R0       ;BYTE OFFSET FROM BASE OF PAGE TABLE
2605          ASHL     #-2,R0,R0                  ;LONG WORD INDEX FROM PAGTBL BASE
2606          ASSUME   SEC$V_VPX EQ 0
2607          BICL3    #^C<SEC$M_VPX>,SEC$S_VPXFC(R1),R2 ;VIRTUAL PAGE NUMBER
2608          SUBL     R2,R0                      ;RELATIVE PAGE IN SECTION
2609 40$:
2610          ASSUME   SEC$L_VBN EQ PFL$S_VBN
2611          ASSUME   SEC$L_WINDOW EQ PFL$S_WINDOW
2612          ADDL     SEC$L_VBN(R1),R0           ;FORM FILE VBN
2613          MOVL     SEC$L_WINDOW(R1),R2       ;FILE WINDOW
2614          RSB
2615

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 59

X-39 MMG\$SWITCH\_PRCPGFL - Switch process page 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```

2617      .SBTTL MMG$SWITCH_PRCPGFL - Switch process pagefile
2618 ;++;
2619 ; FUNCTIONAL DESCRIPTION:
2620 ;
2621 ; This routine is called after a failure to allocate page file
2622 ; backing store for the current PROCESS page file. It reserves
2623 ; an additional number of pages in the BEST, available PROCESS
2624 ; page file.
2625 ;
2626 ; CALLING SEQUENCE:
2627 ;
2628 ;     BSBx/JSB MMG$SWITCH_PRCPGFL
2629 ;
2630 ; INPUT PARAMETERS:
2631 ;
2632 ;     R5 = Process header address (system header if global page)
2633 ;
2634 ; IMPLICIT INPUTS:
2635 ;
2636 ;     PHD$W_PRCPGFLPAGES assumed to be -1
2637 ;     (i.e., called just after failure to allocate backing store)
2638 ;
2639 ; OUTPUT PARAMETERS:
2640 ;
2641 ;     NONE
2642 ;
2643 ; IMPLICIT OUTPUTS:
2644 ;
2645 ;     Switch to BEST PROCESS pagefile, which can be any one of the
2646 ;     existing assignments (including the current one on entry), or
2647 ;     a new assignment resulting from calling MMG$ASNPRCPGFL.
2648 ;
2649 ; NB: All process header fields are correctly set to allow the caller to
2650 ;     immediately allocate page file backing store. It is assumed that
2651 ;     PHD$W_PRCPGFLPAGES is not decremented again in the caller's
2652 ;     assignment retry logic.
2653 ;
2654 ; COMPLETION CODES:
2655 ;
2656 ;     NONE
2657 ;
2658 ; SIDE EFFECTS:
2659 ;
2660 ;     NONE
2661 ;
2662 ;--
2663
2664      UNIVERSAL_SYMBOL      MMG$SWITCH_PRCPGFL
2665 ;MMG$SWITCH_PRCPGFL::
2666      MOVQ      R0, -(SP)          ; Save working registers
2667      MCOMW    PHD$W_PRCPGFLPAGES(R5), R0      ; Reservable page count = -1 ?
2668      BNEQ     30$                ; Bugcheck if not...
2669      MOVZBL   PHD$B_PRCPAGFIL(R5), -(SP)      ; Save current assignment
2670      BSBW     MMG$ASNPRCPGFL      ; Make BEST pagefile current
2671      POPL     R0                  ; Retrieve old assignment and
2672      CVTWL    PHD$W_PRCPGFLOPAGES(R5), R1     ; # pages with backing store
2673      BLSS     40$                ; Bugcheck if page count < 0

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PAGEFAULT - TRANSLATION NOT VALID EXCEPTION HANDLE 10-MAY-1989 16:31:05 VAX MACRO V5.0-

8 Page 60

X-39 MMG\$SWITCH\_PRCPGFL - Switch process page 29-SEP-1988 11:26:39 [SYS.SRC]PAGEFAULT.MAR;1

```
2674      SUBL2  R1,PHD$$_PRCPGFLREFS(R5)[R0]    ; Adj. page cnt. for prev. file
2675      BLSS   40$                               ; Bugcheck if ref count < 0
2676      CLRW   PHD$$_PRCPGFLOPAGES(R5)         ; Show no current allocation
2677      MOVL   G^MMG$$_GL_RSRVPAGCNT,R1        ; # pages to reserve
2678      BSBW   MMG$$_RSRVPRCPGFL2              ; Reserve the pages
2679 20$:  MOVQ   (SP)+,R0                       ; Restore working registers
2680      RSB                               ; and return
2681
2682 30$:   BUG_CHECK NOPRCPGFL,FATAL
2683
2684 40$:   BUG_CHECK BADPRCPGFLC,FATAL
2685
2686      .END
```

## 2 WRTMFYPAG.LIS

WRTMFYPAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 0  
Table of contents

|      |      |   |
|------|------|---|
| (2)  | 166  | DECLARATIONS  |
| (3)  | 312  | Modified page writer initialization                   |
| (4)  | 388  | MODIFIED PAGE WRITE COMPLETION AST                    |
| (5)  | 575  | WRTMFYPAG - WRITE MODIFIED PAGES                      |
| (6)  | 1117 | GETPFNCTX   |
| (7)  | 1255 | PTESCAN - SCAN ADJACENT PTE'S                         |
| (8)  | 1403 | PURGEMPL - Setup to selectively flush pages from MPL  |
| (9)  | 1589 | MMG\$MPW END - Terminate current MPW thread           |
| (10) | 1712 | PURGEMPL_CLEANUP - Remove any DEAD MPL purge requests |

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

WRTMFYPAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 1  
X-26 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYPAG.MAR;1 (2)

```
2 .TITLE WRTMFYPAG - WRITE MODIFIED PAGES
3 .IDENT 'X-26'
4 ; PERF_STATS=1
5 ; DEBUG=^X00000002
6 ;
7 ;*****
8 ;*
9 ;* COPYRIGHT (c) 1978, 1980, 1982, 1984, 1987 BY *
10 ;* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
11 ;* ALL RIGHTS RESERVED. *
12 ;*
13 ;* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
14 ;* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
15 ;* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
16 ;* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
17 ;* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
18 ;* TRANSFERRED. *
19 ;*
20 ;* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
21 ;* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
22 ;* CORPORATION. *
23 ;*
24 ;* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
25 ;* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
26 ;*
27 ;*
28 ;*****
29
30 ;++
31 ; FACILITY:
32 ;
33 ; ABSTRACT:
34 ;
35 ; ENVIRONMENT:
36 ;
37 ; AUTHOR: PETER H. LIPMAN , CREATION DATE: 3-JAN-77
38 ;
39 ; MODIFIED BY:
40 ;
41 ; X-26 SSA0017 Stan Amway 5-Apr-1989
42 ; Rework error handling in I/O completion code to
43 ; accomodate failures due to VBN mapping failures.
44 ; Normally, these should not occur. However, for pages
45 ; with section backing store, some file space management
46 ; operations (notably file extension) can cause the
47 ; the window control blocks to be in a state where
48 ; mapping failures are possible.
49 ;
50 ; Change table in comments that preface MMG$MPW_END
51 ; to reflect algorithmic changes made in X-25.
52 ;
53 ; X-25 SSA0016 Stan Amway 15-Mar-1989
54 ; Several miscellaneous fixes in MMG$PURGEMPL and
55 ; MMG$MPW_END. Change 'HALT's to 'BUG_CHECK's.
56 ;
57 ; X-24 SSA0015 Stan Amway 10-Aug-1988
58 ; In MMG$PURGEMPL, accept optional parameter
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 2  
X-26 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (2)

59 ; for MAINTAIN request.  
60 ;  
61 ; X-23 SSA0014 Stan Amway 1-Jul-1988  
62 ; Rework MPL threshold checking logic.  
63 ;  
64 ; X-22 SSA0013 Stan Amway 24-Jun-1988  
65 ; -21 Cleanup MMG locking: remove CONDITION=RESTORE in  
66 ; UNLOCK macro invocations; avoid a benign, but superfluous  
67 ; LOCK request at label GET\_NXT\_CLUSTER\_1.  
68 ;  
69 ; X-20 SSA0012 Stan Amway 14-Jun-1988  
70 ; Bound latency (IPL SYNCH, MMG lock) of list processing.  
71 ;  
72 ; X-19 SSA0011 Stan Amway 19-Apr-1988  
73 ; Add routine MMG\$PURGEMPL to selectively purge pages  
74 ; from the modified list. Add logic to MMG\$WRTMFYFAG  
75 ; to process list based upon any outstanding criteria  
76 ; established by MMG\$PURGEMPL.  
77 ;  
78 ; X-18 JDC0387 Jon Callas 14-APR-1988  
79 ; Chase the GSD-delete queue properly.  
80 ;  
81 ; X-17 WMC0017 Wayne Cardoza 8-Mar-1988  
82 ; Properly synchronize with swapper when pages with SWPVBN are  
83 ; being written.  
84 ;  
85 ; X-16 SSA0010 Stan Amway 16-Nov-1987  
86 ; Reduce the probability (to practically 0) of the SWAPPER  
87 ; being placed into the FPG or other PFN-depletion related  
88 ; wait states during GSD cleanup. This change complements  
89 ; the one in X-15.  
90 ;  
91 ; X-15 SSA0009 Stan Amway 6-Nov-1987  
92 ; Prevent SWAPPER from being placed in RWMPB wait state  
93 ; due to page faulting during GSD cleanup.  
94 ;  
95 ; X-14 SSA0008 Stan Amway 6-Oct-1987  
96 ; Reinstate code that reenables threshold checking  
97 ; (removed in X-13). Remove setting/clearing of  
98 ; PCB\$V\_SWPVBN in PCB\$L\_STS. It was never used,  
99 ; and with multi-threading, a single bit is not adequate.  
100 ; Correctly handle SWPVBN pages when PFLMAP has  
101 ; been allocated.  
102 ;  
103 ; X-13 JDC0357 Jon Callas 16-APR-1987  
104 ; Add support for cleaning the delete-pending GSD queue.  
105 ; Use ADAWI to modify PCB\$W\_DIOCNT.  
106 ;  
107 ; X-12 SSA0007 Stan Amway 5-Jun-1987  
108 ; Handle page file depletion case correctly to avoid  
109 ; infinite loop.  
110 ;  
111 ; X-11 SSA0006 Stan Amway 18-Mar-1987  
112 ; Properly initialize page file allocation failure array.  
113 ;  
114 ; X-10 SSA0005 Stan Amway 27-Feb-1987  
115 ; Call MPW-specific routines for page file allocation



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 3  
X-26 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (2)

```

116 ;           and deallocation.
117 ;
118 ;           X-9      SSA0004      Stan Amway      4-Nov-1986
119 ;           Force MPW_LOWAITLIMIT >= MPW_LOLIMIT.
120 ;           Add support for multiple pagefiles per process.
121 ;
122 ;           X-8      SSA0003      Stan Amway      13-Oct-1986
123 ;           Call EXE$BLDPKTMFW, instead of EXE$BLDPKTSWPW.
124 ;
125 ;           X-7      RNG0007      Rod Gamache      8-Oct-1986
126 ;           Acquire MMG spinlock at appropriate times.
127 ;
128 ;           X-6      RNG0006      Rod Gamache      24-Sep-1986
129 ;           Unlock MMG before initiating MP write operation.
130 ;
131 ;           V04-005  SSA0002      Stan Amway      11-Sep-1986
132 ;           Reinstate the optimization to initially test the
133 ;           modified page count without raising IPL to SYNCH
134 ;           and acquiring the MMG spinlock.
135 ;
136 ;           V04-004  SSA0001      Stan Amway      22-Aug-1986
137 ;           Multi-thread modified page writing.
138 ;
139 ;           Allow modified list processing to continue after page
140 ;           file space allocation failures.
141 ;
142 ;           When completing an I/O operation, resume processes in
143 ;           MPW_BUSY wait state when modified page count reaches
144 ;           MPW_LOWAITLIMIT, rather than MPW_LOLIMIT.
145 ;
146 ;           Keep track of pages with I/O in process. This value
147 ;           is used by code that checks the free list size to get
148 ;           a better estimate of the true free list size. This finesse
149 ;           is now important because of the potential number of such
150 ;           pages due to multi-threading.
151 ;
152 ;           Add module initialization routine. Logic was formerly
153 ;           in INIT.MAR.
154 ;
155 ;           X-1D2    WMC0001      Wayne Cardoza      12-Feb-1986
156 ;           Make some references G^.
157 ;
158 ;           V04-002  TCM0002      Trudy C. Matthews      3-Oct-1985
159 ;           Change reference to MPW$GL_BADPAGTOTAL to G^.
160 ;
161 ;           V04-001  TCM0001      Trudy C. Matthews      29-Mar-1985
162 ;           Move global data cells in psect $$$210 to SYSDATA.MAR.
163 ;
164 ;--

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 4  
X-26 DECLARATIONS 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (2)

```

166      .SBTTL  DECLARATIONS
167 ;
168 ; INCLUDE FILES:
169 ;
170      $BOOSTATEDEF      ; Bootstrap state definitions
171      $CADEF            ; CONDITIONAL ASSEMBLY DEFINITIONS
172      $CPUDEF          ; PER-CPU DATA BLOCK OFFSETS
173      $DYNDEF          ; Dynamically allocated structure codes
174      $GSDDEF          ; Global section descriptor definitions
175      $INIRTNDEF       ; System loader init routine definitions
176      $IRPDEF          ; I/O REQUEST PACKET DEFINITIONS
177      $IPLDEF          ; PROCESSOR PRIORITY LEVELS
178      $MPWDEF          ; Define MPW structures and constants
179      $OPDEF           ; DEFINE OPCODE EQUIVALENT VALUES
180      $PCBDEF          ; PROCESS CONTROL BLOCK DEFINITIONS
181      $PFLDEF          ; PAGE FILE CONTROL BLOCK DEFINITIONS
182      $PFLMAPDEF       ; Page file mapping window definitions
183      $PFNDEF          ; PAGE FRAME NUMBER DATA BASE DEFINITIONS
184      $PHDDEF          ; PROCESS HEADER DEFINITIONS
185      $PRDEF           ; PROCESSOR REGISTER DEFINITIONS
186      $PRIDEF          ; PRIORITY INCREMENT CLASS DEFINITIONS
187      $PTEDEF          ; PAGE TABLE ENTRY DEFINITIONS
188      $RSNDEF          ; RESOURCE NUMBER DEFINITIONS
189      $SSECDEF         ; SECTION TABLE ENTRY DEFINITIONS
190      $SSDEF           ; System service return codes
191      $$$SYSTEM_PRIM_DATADEF ; DATA PRIVATE TO SYSTEM PRIMITIVES IMAGE
192      $VADEF           ; VIRTUAL ADDRESS FIELD DEFINITIONS
193 ;
194 ; MACROS:
195 ;
196
197      .IF      DF,DEBUG
198      .IF      EQ,DEBUG-1
199 MPW_EVT$C_ALCIRP=0
200 MPW_EVT$C_WRTFAG=1
201 MPW_EVT$C_WRTSEC=2
202 MPW_EVT$C_WRTSWP=3
203 MPW_EVT$C_WRTDON=4
204 MPW_EVT$C_DALIRP=5
205      .MACRO  LOG_EVENT EVT, SRC=(SP), IREG=0
206      .LIST   MEB
207      MOVZBL W^MPW$GB_TRCPTR, R' IREG
208      BISL3  #MPW_EVT$C 'EVT, SRC, W^MPW$GL_TRCBUF[R' IREG]
209      INCB   W^MPW$GB_TRCPTR
210      .NLIST MEB
211      .ENDM
212      .ENDC
213      .ENDC
214 ;
215 ; EQUATED SYMBOLS:
216 ;
217
218      $VIELD  MPW, 0, <-
219      <SUCCESS,,M>,-      ; SUCCESSFUL COMPLETION BIT
220      <BADPAG,,M>,-      ; THIS PAGE HAD A WRITE ERROR
221      <NOTDONE,,M>,-    ; THESE PAGES WERE NOT WRITTEN
222      <INCSEGRA,,M> -   ; I/O incomplete due to VBN map failure

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 5  
X-26 DECLARATIONS 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (2)

```
223 >
224
225 MPW$C_MAXREQCNT = 32 ; Max. no. of pending MPL purge requests
226 ; (NB: must be <= 32 because of ties to
227 ; no. of bits in REQFND/REQFAIL)
228
229 MPW$C_MAXPAGCNT = 128 ; Maximum number of pages to examine
230 ; without releasing MMG spinlock and
231 ; lowering IPL to ASTDEL
232
233 ; *****
234 ;
235 ; ***** THIS ENTIRE MODULE MUST BE RESIDENT *****
236 ;
237 ; *****
238
239 ;
240 ; OWN STORAGE:
241 ;
242 DECLARE_PSECT EXEC$NONPAGED_DATA
243
244 MPW$GL_SVAPTELOW:: ; Aggregate SVAPTE extent range for
245 .LONG 0,0 ; outstanding MPL purge requests
246 MPW$GL_SVAPTEHIGH==MPW$GL_SVAPTELOW+4
247
248 MPW$GQ_SVAPTE:: ; Array of SVAPTE ranges for
249 .LONG 0[2*MPW$C_MAXREQCNT] ; outstanding MPL purge requests

250
251 MPW$AL_PTE::
252 .LONG 0 ; Address of page table entry array
253
254 MPW$AW_PHVINDEXT::
255 .LONG 0 ; Address of process header vector index arr
256
257 MPW$GL_IRPFL::
258 .LONG 0 ; MPW IRP lookaside list FLINK
259
260 MPW$GL_IRPBL::
261 .LONG 0 ; MPW IRP lookaside list BLINK
262
263 MPW$GQ_PGFLX_FAIL::
264 .BLKQ 2 ; Page file allocation failure array
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYPAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 6  
X-26 DECLARATIONS 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYPAG.MAR;1 (2)

```
265
266 ;++
267 ; The following two longwords must be adjacent and in fixed order
268 MPW$GL_REQFND::
269     .LONG    0                ; Bit mask of SVAPTE range indices
270                                     ; found during MPL processing
271 MPW$GL_REQFAIL::
272     .LONG    0                ; Bit mask of SVAPTE range indices
273                                     ; found during MPL processing, but
274                                     ; that couldn't be written
275 ;--
276
277 MPW$GL_REQIDX::
278     .LONG    0                ; Current SVAPTE range request index
279
280 ;++
281 ; The following two longwords must be adjacent and in fixed order
282 MPW$GL_PAGECOUNT::
283     .LONG    0                ; Pages examined with MMG lock held
284 MPW$GL_SAVED_PFN::
285     .LONG    0                ; Current PFN at processing suspension
286 ;--
287
288 ;++
289 ; The following two bytes must be adjacent and in fixed order
290 MPW$GB_STATE::
291     .BYTE    0                ; MPL purge control state
292
293 MPW$GB_REQCNT::
294     .BYTE    0                ; No. of outstanding MPL purge requests
295 ;--
296
297 MPW$GB_IOCNT::
298     .BYTE    0                ; MPW I/O thread count semaphore
299
300     .IF      DF,DEBUG
301     .IF      EQ,DEBUG-1
302
303 MPW$GB_TRCPTR::
304     .BYTE    0
305     .=.+1
306 MPW$GL_TRCBUF::
307     .LONG    0[256]
308
309     .ENDC
310     .ENDC
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 7  
X-26 Modified page writer initialization 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (3)

```

312      .SBTTL Modified page writer initialization
313 ;++
314 ; FUNCTIONAL DESCRIPTION:
315 ;
316 ;      This routine is called by the exec loader to initialize all
317 ;      data structures and cells required for modified page writing.
318 ;
319 ;      For each modified page writing thread, an MPW I/O request packet
320 ;      (MPW IRP) is allocated and linked into a lookaside list. Each MPW
321 ;      IRP contains:
322 ;
323 ;          Standard I/O request packet
324 ;          Page table entry array
325 ;          Process header vector index array
326 ;
327 ;--
328      INITIALIZATION ROUTINE -
329      MPW$INIT
330
331      DECLARE_PSECT EXEC$INIT_CODE
332
333 DELAY_INIT:
334      BBCC      #INIRTN$V_NO_RECALL, -
335              (R5), 10$
336 10$:      MOVL      #SS$ _NORMAL, R0
337      RSB
338
339 MPW$INIT::
340      BBC      #BOOSTATE$V_POOL_INIT, - ; Do nothing if
341              G^EXE$GL_STATE, DELAY_INIT; non-paged pool not initialized
342      PUSHR      #^M<R1, R2, R3>
343      MOVZWL      G^MPW$GW_MPWPFC, R1      ; Modified page writer page fault cluster
344      BICL      #7, R1                      ; Truncate to multiple of 8
345      BNEQ      10$, R1                    ; Cannot allow zero
346      MOVL      #16, R1                    ; Use minimum instead
347 10$:      MOVW      R1, G^MPW$GW_MPWPFC      ; Reset parameter
348      MOVW      R1, G^SWP$GW_SWPINC      ; Make swapfile allocation match MPWPFC
349      MOVL      G^SCH$GL_MFYLIM, R0
350      CMPL      G^MPW$GL_WAITLIM, R0      ; If MPW_WAITLIMIT not >= MPW_HILIMIT,
351      BGEQ      12$, R0                    ; force parameter to
352      ADDL3      R1, R0, G^MPW$GL_WAITLIM ; MPW_HILIMIT + MPW_WRTCLUSTER
353 12$:
354      MOVL      G^MPW$GL_LOWAITLIM, R2
355      CMPL      R2, R0                      ; If MPW_LOWAITLIMIT not <= MPW_HILIMIT,
356      BLEQ      14$, R0                    ; force parameter to
357      SUBL3      R1, R0, R2                ; MPW_HILIMIT - MPW_WRTCLUSTER
358 14$:      CMPL      R2, G^SCH$GL_MFYLOLIM ; If MPW_LOWAITLIMIT not >= MPW_LOLIMIT,
359      BGEQ      16$, R0                    ; force parameter to
360      MOVL      G^SCH$GL_MFYLOLIM, R2      ; MPW_LOLIMIT
361 16$:      MOVL      R2, G^MPW$GL_LOWAITLIM
362
363      CVTBL      G^MPW$GB_IOLIM, R3        ; Maximum MPW concurrent I/O threads
364      CMPL      R3, #1                      ; Insure that value is in range 1-127
365      BGEQ      20$, R3
366      MOVL      #1, R3
367 20$:      MOVB      R3, G^MPW$GB_IOLIM      ; Reset parameter
368      ADDL2      #7, R1                    ; Allow bit-level page file allocation

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

WRTMFYPAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 8  
X-26 Modified page writer initialization 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYPAG.MAR;1 (3)

```
369      ROTL      #2,R1,R4          ; Size of pte array
370      MOVAW     IRP$C_LENGTH(R4) [R1],R1 ; 6 bytes per page to allocate + IRP
371      MOVAL     W^MPW$GL_IRPFL,W^MPW$GL_IRPFL
372      MOVAL     W^MPW$GL_IRPFL,W^MPW$GL_IRPBL
373 30$:  JSB      G^EXE$ALONONPAGED      ; Allocate the storage
374      BLBC     R0,33$
375      MOVW     R1,IRP$W_SIZE(R2)      ; Set structure size
376      MOVW     #<DYN$C_MPWMAP@8!DYN$C_INIT>,-
377      IRP$B_TYPE(R2)      ; Set structure type
378      INSQUE   (R2),@W^MPW$GL_IRPBL   ; Insert into MPW IRP lookaside list
379      INCB     W^MPW$GB_IOCNT      ; Show 1 more thread available
380      SOBGTR   R3,30$              ; Create a MPW IRP for each I/O thread
381 33$:  TSTB     W^MPW$GB_IOCNT      ; At least 1 thread available ?
382      BEQL     40$                ; BR if no
383      CLRB     W^MPW$GB_IOCNT      ; Set semaphore to initial state
384 35$:  MOVL     #SS$_NORMAL,R0
385 40$:  POPR     #^M<R1,R2,R3>
386      RSB
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 9  
X-26 MODIFIED PAGE WRITE COMPLETION AST 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (4)

```

388      .SBTTL  MODIFIED PAGE WRITE COMPLETION AST
389 ;++
390 ; FUNCTIONAL DESCRIPTION:
391 ;
392 ;     THIS ROUTINE IS A KERNEL MODE AST WHICH DOES THE CLEANUP OPERATIONS
393 ; TO COMPLETE THE WRITING OF THE MODIFIED PAGES.  IT ALSO STARTS
394 ; THE NEXT MODIFIED PAGE WRITE IF THERE IS ANY TO DO.
395 ;
396 ; CALLING SEQUENCE:
397 ;
398 ;     BSBW      WRITEDONE
399 ;
400 ; INPUT PARAMETERS:
401 ;
402 ;     IPL = ASTDEL
403 ;     R5 = AST CONTROL BLOCK = I/O REQUEST PACKET
404 ;
405 ; IMPLICIT INPUTS:
406 ;
407 ;     NONE
408 ;
409 ; OUTPUT PARAMETERS:
410 ;
411 ;     R4, R5 ALTERED
412 ;
413 ; IMPLICIT OUTPUTS:
414 ;
415 ;     NONE
416 ;
417 ; COMPLETION CODES:
418 ;
419 ;     NONE
420 ;
421 ; SIDE EFFECTS:
422 ;
423 ;     NONE
424 ;
425 ;--
426
427      DECLARE_PSECT  EXEC$NONPAGED_CODE
428
429      .ENABLE LSB
430 WRITEDONE:
431      PUSHR  #^M<R6,R7,R8,R9,R10,R11,AP> ;PRESERVE NON-VOLATILE REGISTERS
432      LOCK   LOCKNAME=MMG,-             ;LOCK MMG DATABASE FOR PFN CHANGES
433      LOCKIPL=#IPL$_SYNCH,-           ;RAISE IPL
434      SAVIPL=-(SP),-                 ;SAVE CURRENT IPL
435      PRESERVE=NO                     ; OK to destroy R0
436      EXTZV  #VA$V_VPN,#<16-VA$V_VPN>,- ;GET ORIGINAL PAGE COUNT
437      IRP$W_OBCNT(R5),R10
438      SUBL2  R10,G^MPW$GL_IOPAGCNT     ; Show pages no longer in transit
439      MOVL  IRP$L_IOST1(R5),R11        ;GET ERROR STATUS AND TRANSFERRED BYTE COUNT
440      BICL  #<MPW$M_BADPAG!MPW$M_NOTDONE!MPW$M_INCSEGTRA>,R11 ;CLEAR FLAGS
441      CMPW  IRP$L_IOST1(R5),#SS$_INCSEGTRA ; VBN mapping failure ?
442      BNEQ  10$
443      BISL  #MPW$M_INCSEGTRA,R11       ; Yes, remember this case
444 10$:     PUSHL  R5                    ; Deallocate MPW IRP

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 10  
X-26 MODIFIED PAGE WRITE COMPLETION AST 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (4)

```

445      .IF      DF,DEBUG
446      .IF      EQ,DEBUG-1
447      LOG_EVENT EVT=WRIDON
448      .ENDC
449      .ENDC
450      BSBW      DEALLOCATE MPW_IRP
451      MOVAB     IRP$C_LENGTH(R5),-      ; Set address of MPW PTE array
452      W^MPW$AL_PTE
453      MOVZWL    G^MPW$GW_MPWFPC,R0      ; Set address of MPW PHV index array
454      BLSL2     #7,R0
455      MOVAL     @W^MPW$AL_PTE[R0],-
456      W^MPW$AW_PHVINDE
457      ; NB: The addresses just established
458      ;       point into the MPW IRP just
459      ;       returned to the lookaside list.
460
461      CLRL      R6      ;INIT PTE INDEX
462      EXTZV     #<16+VA$V_VPN>,#<16-VA$V_VPN>,R11,R7 ;TRANSFERRED PAGE COUNT
463      BNEQ      20$      ;BRANCH IF NO PAGES SUCCESSFULLY TRANSFERRED
464      BRW       90$
465 20$:  MOVL     @W^MPW$AL_PTE[R6],R0      ;GET PAGE FRAME NUMBER OF NEXT PAGE
466      MOVB      @W^PFN$AB_TYPE[R0],R8    ;PAGE TYPE AND RPTEVT BIT
467      CVTWL     @W^MPW$AW_PHVINDE[R6],R4 ;AND THE CORRESPONDING PROCESS
468      ;HEADER VECTOR INDEX
469      BGEQ      22$      ;BRANCH IF NOT SWAPVBN WRITE
470      BRW       200$
471 22$:  BITL     #<MPW$M_BADPAG ! MPW$M_NOTDONE>,R11 ;NOT SUCCESSFULLY TRANSFERRED?
472      BEQL      25$      ;BRANCH IF THIS PAGE IS OK
473      BISB      #PFN$M_MODIFY,@W^PFN$AB_STATE[R0] ;NOTE PAGE STILL MODIFIED
474 25$:  CMPZV     #PFN$V_PAGTYP,#PFN$S_PAGTYP,- ;PROCESS PAGE TABLE?
475      R8,#PFN$C_PPGTBL
476      BNEQ      40$      ;BRANCH IF NOT
477      MOVL      R4,R1      ;PROCESS HEADER VECTOR INDEX
478      JSB       G^MMG$DECPHDREF1      ;ONE LESS PROCESS HEADER REF
479      ;AT PAGE WRITE COMPLETION
480 40$:  DECREF     GTR=60$      ;ONE LESS REFERENCE
481      BBC       #MPW$V_BADPAG,R11,50$    ;BRANCH IF NOT PAGE WRITE ERROR PAGE
482      BBS       #MPW$V_INCSEGTRA,R11,45$ ; If VBN map failure, don't place on bad
483      ; page list; return to modify list
484      MOVZBL     #PFN$C_BADPAGLST,R2      ;PLACE THIS PAGE
485      JSB       G^MMG$INSPFNT      ;ON THE BAD PAGE LIST
486      INCL      G^MPW$GL_BADPAGTOTAL    ;COUNT IT
487      BRB       60$
488 45$:  MOVL     @W^PFN$AL_BAK[R0],R9      ;Get backing store address
489      BBS       #PTE$V_TYPO,R9,50$      ; Continue if section backing store,
490      BUG_CHECK VBNMAPFAIL,FATAL      ; VBN mapping failure should never
491      ; happen for pages destined for a page
492      ; file
493 50$:  JSB       G^MMG$RELPFN      ;RELEASE THE PAGE
494 60$:  CVTWL     @W^PHV$GL_PIXBAS[R4],R4  ;CALCULATE PCB ADDRESS FROM PHV INDEX
495      BLSS      80$      ;BRANCH IF PCB IS GONE
496      MOVL     @W^SCH$GL_PCBVEC[R4],R4  ;FETCH PCB ADR
497      ;
498      ; IF DELPAG IS WAITING FOR THIS WRITE COMPLETION, THE RPTEVT BIT IS SET
499      ;
500 70$:  BBC       #PFN$V_RPTEVT,R8,80$    ;BRANCH IF NO REPORT EVENT REQUESTED
501      BSBW      RPT_EVT      ;ELSE, REPORT THE PFCOM EVENT

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 11  
X-26 MODIFIED PAGE WRITE COMPLETION AST 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (4)

```

502 80$:      AOBLSS  R7,R6,92$      ;LOOP THROUGH ALL PAGES
503 90$:      BLBS    R11,100$      ;BRANCH IF PAGE WRITE HAD NO ERROR
504          INCL    R7              ;ONE MORE PAGE FOR THE ERROR PAGE
505          BBS     #MPW$V_BADPAG,R11,95$ ;BRANCH IF ALREADY MARKED BAD
506 92$:      BRW     20$           ;BRANCH IF HAVEN'T PROCESSED THE ERROR PAGE
507
508 ;
509 ; NOW PROCESS THE UNWRITTEN PAGES IF ANY
510 ;
511 95$:      MOVL    R10,R7          ;RESET LIMIT TO ORIGINAL PAGE COUNT
512          DECL    R6              ;IN ORDER TO EXECUTE THE LOOP 0 OR MORE TIME
513          BICB    #^C<MPW$M_INCSEGTRA>,R11; Save INCSEGTRA flag
514          BISB    #<MPW$M_SUCCESS ! MPW$M_NOTDONE>,R11 ;COMPLETE THE PAGES NOT TRANSFE
515          BRB     80$
516 100$:     BBS     #MPW$V_INCSEGTRA,R11,150$ ; BR if VBN map failure
517          UNLOCK  LOCKNAME=MMG,-   ;UNLOCK MMG DATABASE
518          NEWIPL=(SP),-            ;RESTORE PREVIOUS IPL
519          PRESERVE=NO              ; OK to destroy RO
520          BRW     GET_NXT_CLUSTER  ; Try to write another I/O cluster
521
522 ;
523 ; We have incurred a VBN mapping failure on an I/O segment (a very rare event).
524 ; Quiesce the current modified page writing thread to minimize the probability
525 ; that we'll have to deal with this case again in the near future.
526 ;
527 150$:
528          ASSUME  MPW$GL_REQFAIL EQ MPW$GL_REQFND+4
529          CLRQ    W^MPW$GL_REQFND   ; Show no pages found/failed
530          ASSUME  MPW$GL_SAVED_PFN EQ MPW$GL_PAGECOUNT+4
531          CLRQ    W^MPW$GL_PAGECOUNT ; Show no pages examined
532                                     ; and no saved PFN
533          CLRL    -(SP)              ; No MPW IRP allocated yet
534          BRW     NOMOREPAGES        ; Terminate MPW in orderly fashion
535
536 ;
537 ; PROCESS SWPVBN CASE
538 ;
539 200$:     EXTZV  #0,#15,R4,R4      ;FIX UP THE PHV INDEX
540          BBC     S^#SCH$V_SIP,G^SCH$GB_SIP,220$ ;CONTINUE IF SWAPPER NOT ACTIVE
541          CVTWL  @W^PHV$GL_PIXBAS[R4],R2 ;CALCULATE PCB ADDRESS FROM PHV INDEX
542          BLSS   220$               ;BRANCH IF PCB IS GONE
543          MOVL  @W^SCH$GL_PCBVEC[R2],R2 ;FETCH PCB ADR
544          CML   R2,G^SWP$GL_INPCB    ;COULD IT BE THIS PROCESS INSWAPPING
545          BNEQ  220$                 ;NO - JUST PROCEED NORMALLY
546          DECF  EQL=210$             ;MAKE SURE SWAPPER HASN'T HANDLED IT
547          INCW  @W^PFN$AW_REFCNT[R0] ;PUT REF COUNT BACK THE WAY IT WAS
548          BRB   220$                 ;HANDLE PAGE NORMALLY
549 210$:     MOVZBL #PFN$C_MFYFAGLST,R2 ;PUT BACK ON MODIFY LIST
550          JSB   G^MMG$INSPFNT        ;AT TAIL SO WE WON'T SEE IT FOR A WHILE
551          BRW   70$
552
553 220$:     CLRW  @W^PFN$AW_SWPVBN[R0] ;ALL DONE WITH THE SWPVBN FIELD
554          BITL  #MPW$M_INCSEGTRA,R11 ; VBN mapping failure ?
555          BEQL  230$                 ; Continue if not
556          BUG_CHECK VBNMAPFAIL,FATAL ; VBN mapping failure should never
557                                     ; happen for pages destined for a swap
558          ; image

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYPAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 12  
X-26 MODIFIED PAGE WRITE COMPLETION AST 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYPAG.MAR;1 (4)

```
559 230$: BRW      25$                ;AND IN THE CASE OF A WRITE ERROR
560                                           ;DO NOT JAM ON THE MODIFY BIT
561
562 RPT_EVT:
563     MOVZBL  #PRI$_IOCOM,R2          ;I/O COMPLETE PRIORITY CLASS
564     LOCK    LOCKNAME=SCHED,-        ;LOCK SCHED DATABASE FOR RPTEVT
565     LOCKIPL=#IPL$_SYNCH,-          ;RAISE IPL (already at that IPL)
566     PRESERVE=NO                    ; OK to destroy R0
567     RPTEVT  PFCOM                  ;REPORT PAGE FAULT COMPLETE
568     UNLOCK  LOCKNAME=SCHED,-        ;UNLOCK SCHED DATABASE
569     PRESERVE=NO                    ; OK to destroy R0
570                                           ;LEAVE IPL AT SYNCH
571     RSB                               ; Return
572
573     .DISABLE LSB
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 13  
X-26 WRTMFYFAG - WRITE MODIFIED PAGES 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (5)

```
575      .SBTTL  WRTMFYFAG - WRITE MODIFIED PAGES
576 ;++
577 ; FUNCTIONAL DESCRIPTION:
578 ;
579 ;     THIS ROUTINE GATHERS A CLUSTER OF PAGES OF LIKE KIND FROM THE
580 ; MODIFIED PAGE LIST, AND WRITES THEM BACK TO THEIR BACKING STORE ADDRESSES.
581 ; FOR PAGING FILE PAGES, THE ADDRESSES ARE REALLOCATED AS A CONTIGUOUS BLOCK
582 ; AND THE PAGES ARE WRITTEN BACK IN ONE OPERATION.
583 ;
584 ; CALLING SEQUENCE:
585 ;
586 ;     BSBW      MMG$WRTMFYFAG
587 ;
588 ; INPUT PARAMETERS:
589 ;
590 ;     IPL = 0
591 ;
592 ; IMPLICIT INPUTS:
593 ;
594 ;     NONE
595 ;
596 ; OUTPUT PARAMETERS:
597 ;
598 ;     R4, R5 ALTERED
599 ;
600 ; IMPLICIT OUTPUTS:
601 ;
602 ;     NONE
603 ;
604 ; COMPLETION CODES:
605 ;
606 ;     NONE
607 ;
608 ; SIDE EFFECTS:
609 ;
610 ;     NONE
611 ;
612 ;--
613
614      .ENABLE      LOCAL_BLOCK
615
616 ;-
617 ; Local subroutine to conditionally deallocate a MPW IRP
618 ;
619 ; Inputs:
620 ;     (SP)      Return address
621 ;     4(SP)    0 or MPW IRP to deallocate
622 ;
623 ; Outputs:
624 ;     NONE
625 ;
626 ; Side Effects:
627 ;     R0, R1 Destroyed
628 ;
629 ;     If MPW IRP address was <> 0, then
630 ;         IRP returned to lookaside list
631 ;         MPW$GB_IOCNT updated
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 14  
X-26 WRTMFYFAG - WRITE MODIFIED PAGES 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (5)

```

632 ;
633 ;      MPW IRP address removed from stack
634 ;--
635
636 DEALLOCATE_MPW_IRP:
637     MOVL     G^CTL$GGL_PCB,R0      ; Get PCB address
638     MOVL     4(SP),R1              ; Unused MPW IRP to release ?
639     BEQL     1$                    ; Br if none
640     INSQUE   (R1),@W^MPW$GGL_IRPBL ; Return MPW IRP to lookaside list
641     .IF     DF,DEBUG
642     .IF     EQ,DEBUG-1
643     LOG_EVENT EVT=DALIRP, SRC=4(SP), IREG=1
644     .ENDC
645     .ENDC
646     ADAWI    #-1,PCB$W_DIOCNT(R0)   ; Restore DIO quota
647     INCB     W^MPW$GB_IOCNT         ; Show 1 less MPW I/O thread
648     BGTR     BR_BAD_MPW_CNT
649 1$:     MOVL     (SP)+, (SP)        ; Remove MPW IRP address from stack
650     RSB
651     ; Return
652 BR_BAD_MPW_CNT:
653     BRW      BAD_MPW_CNT
654
655 ;
656 ; Inputs:
657 ;      MMG spinlock must be held - IPL = SYNCH
658 ;
659 NOMOREPAGES:
660     BSBB     DEALLOCATE_MPW_IRP     ; Deallocate MPW IRP
661     CMLPL    #MPW$C_MAXPAGCNT, -    ; Are we here due to examined
662     W^MPW$GGL_PAGECOUNT           ; page count overflow ?
663     BLSS     10$                    ; BR if yes
664     PUSHAB   B^21$                  ; (cheap branch)
665     BRW      MMG$MPW_END             ; Cleanup and set MPW state
666
667 ;+++
668 ; Inputs:
669 ;      MMG spinlock must be held - IPL = SYNCH
670 ;      (SP) = Saved IPL at entry
671
672 10$:     UNLOCK LOCKNAME=MMG, -      ; Unlock MMG database
673     NEWIPL=#IPL$_ASTDEL, -         ; Lower IPL, but block AST threads
674     PRESERVE=NO                     ; OK to destroy R0
675     LOCK     LOCKNAME=MMG, -        ; Lock MMG database
676     PRESERVE=NO                     ; OK to destroy R0
677     MOVL     W^MPW$GGL_SAVED_PFN,R0 ; Retrieve saved PFN
678     CMPZV    #PFN$V_LOC,#PFN$$S_LOC,- ; Page still on modified list ?
679     @W^PFN$AB_STATE[R0],#PFN$C_MFYFAGLST
680     BEQL     20$                    ; BR if yes
681 15$:     CLRL     W^MPW$GGL_SAVED_PFN ; List has changed - restart scan from begin
682 20$:     CLRL     W^MPW$GGL_PAGECOUNT ; Show no pages examined
683     BRW      GET_NXT_CLUSTER_1      ; Join mainline processing at restart point
684
685 21$:     UNLOCK LOCKNAME=MMG, -      ; UNLOCK MMG DATABASE
686     NEWIPL=(SP)+, -                 ; RESTORE PREVIOUS IPL
687     PRESERVE=NO                     ; OK to destroy R0
688 ;+++

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 15  
X-26 WRTMFYFAG - WRITE MODIFIED PAGES 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (5)

```

689 ;
690 ;Check the queue of deleted GSDs for clean-up
691 ;
692     Cmpl      G^EXE$GL_GSDDELFL,-      ;Quick check on the deleted GSD queue.
693     #EXE$GL_GSDDELFL      ;If we miss this because of SMP, we
694     BNEQ      25$                ;don't care for two reasons: (1) we'll
695                                     ;do it again in another second. (2) it
696                                     ;wasn't the reason we're running now
697                                     ;anyway. We'll be woken again if need
698                                     ;be.
699
700 22$:     POPR      #^M<R6,R7,R8,R9,R10,R11,AP> ;RESTORE NON-VOLATILE REGISTERS
701     RSB                                     ;AND RETURN
702
703 25$:     MOVAL     G^EXE$GL_GSDMTX,R0      ;Get the GSD mutex, but don't wait.
704     MOVL      G^CTL$GL_PCB,R4
705     JSB       G^SCH$LOCKWNOWAIT
706     BLBC     R0,22$
707
708     BISB     S^#MMG$M_NOWAIT,G^MMG$GB_FREWFLGS ; Prevent FREWLSE MWAIT
709     MOVAL     G^EXE$GL_GSDDELFL,R2      ;Get the head of the queue
710     MOVL      R2,R3
711
712 30$:
713 ;
714 ; The following check is done to avoid placing the SWAPPER in a memory
715 ; management related wait state due to unavailable physical page frames.
716 ; The worst case requirement, assuming that a GSD is <= 1 page in size,
717 ; is 2 pages (allowing for a page crossing). While it is possible that
718 ; either a) concurrent activity by a higher priority, real time process, or
719 ; b) concurrent activity on other CPUs in a SMP configuration could use
720 ; all available page frames after the check is made, the probability is
721 ; EXTREMELY small, approaching zero for practical purposes.
722 ;
723     Cmpl      G^SCH$GL_FREECNT,#16      ; Do we have adequate free pages ?
724     BLEQU     60$                ; BR if no; avoid FPG wait, etc.
725     MOVL      (R3),R3            ;Get next entry.
726     Cmpl      R2,R3              ;No more?
727     BEQL     60$                ;Then exit.
728     MOVL      GSD$L_IPID(R3),R0      ;Get the IPID of the nominal owner
729     BEQL     30$                ;Skip it if null
730     JSB       G^MMG$QUEUE_GSD_CLEAN ;Queue the AST
731     CLRL     GSD$L_IPID(R3)        ;Don't queue it twice
732     BRB      30$                ;Go back and get another
733
734 60$:     BICB     S^#MMG$M_NOWAIT,G^MMG$GB_FREWFLGS ; Allow FREWLSE MWAIT
735     PUSHAB   B^22$              ;(return through common exit)
736     JMP      G^MMG$GSDMTXULK      ;Unlock the mutex
737
738     .IF      DF,PERF_STATS
739 ENTRY_STATS:
740     MOVZBL   W^MPW$GB_STATE,R0      ; Get current MPW state for indexing
741     ADDL2    G^SCH$GL_MFYCNT,-      ; Accumulate MPL count at entry
742     G^PMS$GL_RESERVED1+ -
743     <4*<MPW$C_MAXSTATE*2>>[R0]
744     INCL     G^PMS$GL_RESERVED1+ - ; Count the number of entries
745     <4*<MPW$C_MAXSTATE*3>>[R0]

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 16  
X-26 WRTMFYFAG - WRITE MODIFIED PAGES 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (5)

```

746      BRB      GET_NXT_CLUSTER_2      ; Join main processing loop
747      .ENDC
748
749      .DISABLE      LOCAL_BLOCK
750
751      UNIVERSAL_SYMBOL      MMG$WRTMFYFAG
752 ;MMG$WRTMFYFAG::
753      CMLP      G^SCH$GL_MFY LIM, -      ; Enough pages on modified
754      G^SCH$GL_MFYCNT      ; page list to consider writing?
755      BLEQ      20$      ; BR if yes
756 10$:      RSB      ; Nothing to do for now
757 20$:      BBSSI      S^#SCH$V_MPW,G^SCH$GB_SIP,10$ ; Do nothing if already active
758 ;
759 ; FIRST ENTRY TO MODIFIED PAGE WRITER
760 ;
761      PUSHR      #^M<R6,R7,R8,R9,R10,R11,AP> ; Save non-volatile registers
762      CLRQ      -(SP)      ; No MPW IRP allocated, saved IPL = 0
763      LOCK      LOCKNAME=MMG, -      ; Lock MMG database
764      LOCKIPL=#IPL$_SYNCH,-      ; Raise IPL
765      PRESERVE=NO      ; OK to destroy R0
766
767      ASSUME      MPW$GL_REQFAIL EQ MPW$GL_REQFND+4
768      CLRQ      W^MPW$GL_REQFND      ; Show no pages found/failed
769      ASSUME      MPW$GL_SAVED_PFN EQ MPW$GL_PAGECOUNT+4
770      CLRQ      W^MPW$GL_PAGECOUNT      ; Show no pages examined
771      ; and no saved PFN
772
773      ; Set maintenance state (conditionally)
774      MOVZWL      #<MPW$C_MAINTAIN!MPW$M_LOLIMIT>,R0
775      MOVL      G^MPW$GL_LOWAITLIM,R1      ; New low MPL limit is MPW_LOWAITLIMIT
776      .IF      NDF,PERF_STATS
777      PUSHAB      B^GET_NXT_CLUSTER_2      ; Join main processing loop
778      .IFF
779      PUSHAB      B^ENTRY_STATS      ; Maintain entry statistics
780      .ENDC
781      BRW      MMG$PURGEMPL
782
783 BAD_MPW_CNT:
784      BUG_CHECK MPWALCIRP,FATAL      ; Failure to correctly allocate MPW IRP
785
786 ;
787 ; Continuation after either I/O initiation or completion
788 ;
789 GET_NXT_CLUSTER:
790      LOCK      LOCKNAME=MMG, -      ; Lock MMG database
791      LOCKIPL=#IPL$_SYNCH,-      ; Raise IPL
792      PRESERVE=NO      ; OK to destroy R0
793      ASSUME      MPW$GL_REQFAIL EQ MPW$GL_REQFND+4
794      CLRQ      W^MPW$GL_REQFND      ; Show no pages found/failed
795      ASSUME      MPW$GL_SAVED_PFN EQ MPW$GL_PAGECOUNT+4
796      CLRQ      W^MPW$GL_PAGECOUNT      ; Show no pages examined
797      ; and no saved PFN
798 ;
799 ; Continuation after processing suspension
800 ;
801 GET_NXT_CLUSTER_1:
802      CLRL      -(SP)      ; No MPW IRP allocated yet

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYPAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 17  
X-26 WRTMFYPAG - WRITE MODIFIED PAGES 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYPAG.MAR;1 (5)

```

803      CMPL      G^SCH$GL_MFYCNT,-      ; Enough pages on modify list
804      G^SCH$GL_MFYLOLIM      ; to try for another cluster?
805      BLEQU     BR_NOMOREPAGES      ; No, all done for now
806 ;
807 ; Continuation after initial entry
808 ;
809 GET_NXT_CLUSTER_2:
810      REMQUE    @W^MPW$GL_IRPFL,R2      ; Get a MPW IRP from lookaside list
811      BVS      BR_NOMOREPAGES      ; BR if none available
812      MOVL     R2,(SP)      ; Save MPW IRP address
813      .IF      DF,DEBUG
814      .IF      EQ,DEBUG-1
815      LOG EVENT EVT=ALCIRP
816      .ENDC
817      .ENDC
818      MOVL     G^CTL$GL_PCB,R4      ; Get PCB address of this process
819      ADAWI     #1,PCB$W_DIOCNT(R4)      ; Insure enough DIO quota
820      DECB     W^MPW$GB_IOCNT      ; Show another MPW I/O thread active
821      BGEQ     BAD_MPW_CNT      ; Bugcheck if count >= 0
822      MOVAB    IRP$C_LENGTH(R2),-      ; Set address of MPW PTE array
823      W^MPW$AL_PTE
824      MOVZWL   G^MPW$GW_MPWFC,R0      ; Set address of MPW PHV index array
825      BISL2    #7,R0
826      MOVAL    @W^MPW$AL_PTE[R0],-
827      W^MPW$AW_PHVINDEX
828      CLRQ     W^MPW$GO_PGFLX_FAIL      ; Clear page file allocation
829      CLRQ     W^MPW$GO_PGFLX_FAIL+8      ; failure array
830      CLRL     W^MPW$GL_REQIDX      ; Assume current index will be zero
831      MOVL     W^MPW$GL_SAVED_PFN,R0      ; Retrieve saved PFN (if any)
832      BNEQ     GET_PAGE_TYPE_1
833      MOVL     G^PFN$AL_MFYLSTHD,R0      ;FIRST PFN IN MODIFIED PAGE LIST
834 GET_PAGE_TYPE:
835      BEQL     BR_NOMOREPAGES      ;BRANCH IF LIST IS EMPTY
836 GET_PAGE_TYPE_1:
837      AOBLEQ    #MPW$C_MAXPAGCNT,-      ; Count another page examined
838      W^MPW$GL_PAGECOUNT,-      ; and suspend processing if maximum
839      PROCESS_PAGE      ; page count exceeded
840      MOVL     RO,W^MPW$GL_SAVED_PFN      ; Save restart PFN
841 BR_NOMOREPAGES:
842      BRW      NOMOREPAGES
843 PROCESS_PAGE:
844      BSBW     GETPFNCTX      ;SET UP TO PROCESS THIS PFN
845      CLRL     R6      ;INIT INDEX TO PTE ARRAY
846      CASE     R1,<-      ;DISPATCH ON BACKING STORE TYPE
847      PAGEFILE,-      ;PAGING FILE PAGE
848      SECTION,-      ;SECTION PAGE (PROCESS OR GLOBAL)
849      BADBAKADR,-      ;GLOBAL BACKING STORE ADDRESS
850      SWPVBN,-      ;SWPVBN, WRITE BACK TO SWAP FILE
851      NEXT_MFYYPAG >      ; Skip page
852 BADBAKADR:
853      BUG_CHECK IVBAKADIO,FATAL      ;INVALID BACKING STORE ADDRESS FOR I/O
854 ;
855 ; NO PAGE FILE VBN'S AVAILABLE IN THIS PAGE FILE
856 ;
857 NO_PAGEFILE:
858      BBSS     W^MPW$GL_REQIDX,-      ; Show failure to write for current
859      W^MPW$GL_REQFAIL,5$      ; request index

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 18  
X-26 WRTMFYFAG - WRITE MODIFIED PAGES 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (5)

```

860 5$:      POPR      #^M<R0,R3,R4,R5,R6,R7> ;RECOVER REGISTERS FROM STACK
861          BSS      R7,W^MPW$GQ_PGFLX_FAIL,-; Show an allocation failure for this
862          NEXT MFYPAG ; page file index
863 10$:     BBCCI    S^#SCH$V_TCD,G^SCH$GB_SIP,- ; Enable threshold checking again
864          NEXT MFYPAG
865 ; (Fall through to try next page...)
866 NEXT_MFYFAG:
867          PFN_REFERENCE -
868          MOVZWL    <@W^PFN$Ax FLINK[R0],RO>,- ;CHAIN TO NEXT PAGE
869          LONG_OPCODE=MOVL,-
870          IMAGE=SYS_NONPAGED
871          BRB      GET_PAGE_TYPE ;AND PROCESS IT
872 ;
873 ; PAGE FILE PAGE, GATHER A CLUSTER FROM THE SAME PAGE FILE
874 ;
875 PAGEFILE:
876          EXTZV    #PFN$V_PGFLX,#PFN$S_PGFLX,R2,R7 ;PAGE FILE INDEX
877          BBS      R7,W^MPW$GQ_PGFLX_FAIL,-; Skip page if previous space allocation
878          NEXT MFYPAG ; failure for this page file index
879          PUSHR    #^M<R0,R3,R4,R5,R6,R7> ;SAVE PFN, SVAPTE,PCB,PHD,PTE INDEX,FILE
880          CLRQ     R0 ;NOTHING TO FREE
881          MOVL     @MMG$GL_PAGSWPVC[R7],R3 ;GET PAGE FILE CONTROL BLOCK ADDRESS
882          MOVZBL   PFL$B_ALLOCSIZ(R3),R2 ;DESIRED CLUSTER SIZE
883 5$:       JSB      G^MPW$ALLOCPAGFIL1 ;ALLOCATE A CLUSTER
884          BNEQ     20$ ;BRANCH IF WE HAVE ALLOCATION
885          SUBL     #16,R2 ;ALLOCATION FAILED, TRY SMALLER CLUSTER
886          BLEQ     10$ ;BRANCH IF ALREADY AT MINIMUM
887          MOVB     R2,PFL$B_ALLOCSIZ(R3) ;SET NEW SIZE TO ATTEMPT FROM NOW ON
888          CLRL     PFL$L_STARTBYTE(R3) ;START AT BEGINNING OF MAP
889          BRB      5$ ;TRY AGAIN
890
891 10$:      JSB      G^MPW$ALLOCPAGFIL2 ;ALLOCATE SPACE, ANY AMOUNT OK
892          BEQL     NO_PAGEFILE ;BRANCH IF NONE, NO WRITING CAN BE DONE
893 20$:     MOVL     R2,R11 ;SAVE NUMBER OF PAGES ALLOCATED
894          MOVL     R0,R8 ;AND THE STARTING PAGE FILE VBN
895          POPR     #^M<R0,R3,R4,R5,R6,R7> ;RECOVER PFN, SVAPTE
896          MOVL     R11,R9 ;NUMBER OF PAGEFILE PAGES ALLOCATED
897 ;
898 ; AT THIS POINT THE REGISTERS CONTAIN THE FOLLOWING VALUES:
899 ; R0 = PFN
900 ; R3 = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE ENTRY
901 ; R4 = PCB ADDRESS FOR THE PROCESS IN WHICH THIS CODE IS RUNNING
902 ; R5 = PHD ADDRESS OF THE PROCESS WHICH OWNS THE MODIFIED PAGE
903 ; R6 = INDEX TO NEXT ENTRY TO USE IN PTE AND PHVINDEXT ARRAYS
904 ; R7 = PAGE FILE INDEX
905 ; R8 = NEXT PAGE FILE VBN TO USE
906 ; R9 = NUMBER OF PAGE FILE VBN'S NOT YET USED IN THE CLUSTER ALLOCATED
907 ; R11 = NUMBER OF PAGE FILE VBN'S ALLOCATED IN THE CLUSTER
908 ;
909 PAGFILCLUSTER:
910          PUSHL    R11 ;SAVE COUNT OF ALLOCATED PAGE FILE
911          PFN_REFERENCE -
912          MOVZWL    <@W^PFN$Ax BLINK[R0],-(SP) >,- ;REMEMBER WHERE TO RESTART SCAN OF L
913          LONG_OPCODE=MOVL,-
914          IMAGE=SYS_NONPAGED
915 ;          CLRL     MPW$L_COUNT ;INIT COUNT OF CLUSTER TO 0
916          BSBW     PTESCAN ;TRY TO GET ADJACENT PAGES TO THIS ONE

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYPAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 19  
X-26 WRTMFYPAG - WRITE MODIFIED PAGES 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYPAG.MAR;1 (5)

```

917 ;      MOVL      MPW$L_COUNT,R0          ;GET THE COUNT
918 ;      INCL      MPW$A_PGFLCLUSTERS[R0]    ;BUMP THE COUNT
919
920 ;
921 ; DONE WITH THIS CLUSTER OF PAGE TABLE ENTRIES
922 ; IF MORE PAGE FILE VBN'S ARE LEFT, SCAN MODIFIED LIST FOR MORE PAGES
923 ; IN SAME PAGE FILE.
924 ; 0(SP) = SAVED PFN (OR 0) TO LINK FORWARD FROM
925 ; 4(SP) = NUMBER OF PAGE FILE VBN'S ALLOCATED
926 ;
927      POPR      #^M<R0,R11>                  ;R0 = SAVED PFN TO LINK FORWARD
928                                          ;R11 = NUMBER OF PAGE FILE VBN'S ALLOCATED
929      SUBL3     R6,R11,R9                    ;NO. OF PAGE FILE VBN'S NOT USED
930 ; DISABLE THE FOLLOWING BECAUSE THIS MIGHT RETURN A SMALL ALLOCATION WITHOUT
931 ; WRITING ANY PAGES AT ALL.
932 ;      CMPW      R9,W^MPW$GW_MINLIM        ;ARE WE AT THRESHOLD TO SHUT OFF CLUSTER
933      BLEQ     100$                          ;BRANCH IF USED ALL THAT WE SHOULD
934      TSTL     RO                             ;PFN 0 IS LIST HEAD
935      BNEQ     70$                            ;GET FLINK AS NEXT CANDIDATE
936      MOVL     G^PFN$AL_MFYLSTHD,RO        ;NEXT CANDIDATE FROM FRONT OF LIST
937      BRB      75$
938 70$:      PFN_REFERENCE -
939      MOVZWL   <@W^PFN$A<x FLINK[R0],RO>,-    ;CHAIN TO NEXT PFN IN LIST
940      LONG_OPCODE=MOVL,-
941      IMAGE=SYS_NONPAGED
942 75$:      BEQL     100$                      ;BRANCH IF END OF LIST
943 80$:      BSBW     GETPFNCTX                 ;SET UP TO PROCESS THIS PFN
944      TSTL     R1                             ;PAGE FILE VBN?
945      BNEQ     70$                            ;BRANCH IF NOT
946      CMPZV    #PFN$V_PGFLX,#PFN$S_PGFLX,R2,R7 ;SAME PAGE FILE INDEX?
947      BNEQ     70$                            ;BRANCH IF NOT
948      BRB      PAGFILCLUSTER                 ;FIND ANOTHER PTE CLUSTER
949 ;
950 ; SET UP TO WRITE THIS CLUSTER OF PAGES
951 ;
952 100$:     SUBL3     R6,R8,R11                ;FORM AND SAVE FIRST PAGE FILE VBN
953      MOVL     @MMG$GL_PAGSWPVC[R7],R3      ;ADDRESS OF PAGE FILE CONTROL BLOCK
954      MOVL     R9,R1                          ;ANY PAGE FILE VBN'S TO DEALLOCATE?
955      BEQL     140$                          ;BRANCH IF THEY WERE ALL USED
956      MOVL     R8,R0                          ;SET VBN NUMBER OR AREA TO DEALLOCATE
957 ;
958 ; THERE ARE R1 PAGES OF PAGE FILE ALLOCATED BUT NOT USED STARTING WITH VBN R0.
959 ; MUST RETURN THEM TO THE PAGE FILE
960 ;
961      JSB      G^MPW$DEALLOCPAGFIL          ;FREE THE PAGES IN THE FILE
962 ;
963 ; NOW SET UP TO DO THE CALL TO BUILDPKT
964 ;      R3 = PAGE FILE CONTROL BLOCK ADDRESS
965 ;      R6 = NO. OF PAGES TO TRANSFER
966 ;      R7 = PAGE FILE INDEX
967 ;      R11 = STARTING PAGEFILE VBN
968 ;
969      ASSUME    SEC$L_VBN EQ PFL$L_VBN
970      ASSUME    SEC$L_WINDOW EQ PFL$L_WINDOW
971 140$:
972      .IF      DF,DEBUG
973      .IF      EQ,DEBUG-1

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 20  
X-26 WRTMFYFAG - WRITE MODIFIED PAGES 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (5)

```

974      LOG_EVENT EVT=WRTFAG
975      .ENDC
976      .ENDC
977      ADDL3   R11,PFL$L_VBN(R3),R0      ;FORM VBN IN PAGE FILE
978      MOVL    PFL$L_WINDOW(R3),R2      ;WINDOW ADDRESS
979 ;
980 ; 0(SP) = ADDRESS OF I/O REQUEST PACKET
981 ; 4(SP) = CALLER'S IPL
982 ; R0 = VBN IN FILE TO WRITE
983 ; R2 = ADDRESS OF WINDOW FOR FILE
984 ; R4 = PROCESS CONTROL BLOCK ADDRESS FOR THE PROCESS IN WHICH THIS CODE IS RUNNING
985 ; R6 = NUMBER OF PAGES TO WRITE
986 ;
987 MPW_BLDPKT:
988      ADDL2   R6,G^MPW$G_L_IOPAGCNT      ; Show more pages being written
989      ASHL    #9,R6,R1                    ;NUMBER OF BYTES TO WRITE
990      MOVAL   @W^MPW$SAL_PTE,R3          ;SVAPTE FOR TRANSFER
991
992
993      .IF     GT,CA$ MEASURE
994      ADDL    R6,G^PMS$G_L_PWRITES      ;COUNT THE PAGES WRITTEN
995      INCL   G^PMS$G_L_PWRITIO          ;AND THE NUMBER OF WRITE REQUESTS
996      .ENDC
997
998      .IF     DF,PERF_STATS
999      MOVZBL  W^MPW$GB_STATE,R5          ; Get current MPW state for indexing
1000     ADDL    R6,G^PMS$G_L_RESERVED1[R5] ;COUNT THE PAGES WRITTEN
1001     INCL   G^PMS$G_L_RESERVED1+<4*MPW$C_MAXSTATE>[R5] ;AND THE NUMBER OF WRITE R
1002     .ENDC
1003
1004     UNLOCK  LOCKNAME=MMG,-                ;UNLOCK MMG DATABASE
1005     NEWIPL=#IPL$_ASTDEL                ; Lower IPL, but block AST threads
1006
1007     MOVL    (SP)+,R5                      ;GET I/O PACKET ADDRESS
1008     MOVAL   W^WRITEDONE,IRP$L_ASTPRM(R5) ;ADDRESS OF KERNEL MODE AST
1009     ;FOR WRITE COMPLETION PROCESSING
1010     SUBB3   G^MPW$GB_PRIO,#31,IRP$B_PRI(R5) ;SET PRIORITY FOR TRANSFER
1011     JSB     G^EXE$BLDPKTMPW            ;BUILD AND QUEUE THE I/O REQUEST PACKET
1012 ;
1013 ; R4 NO LONGER HAS PCB ADDRESS IN IT
1014 ;
1015
1016     BRW     GET_NXT_CLUSTER                ; Try to write another I/O cluster
1017
1018 ;
1019 ; SECTION PAGE - TRY TO FORM A CLUSTER OF THESE
1020 ;
1021
1022     .ENABL  LSB
1023
1024 SECTION:
1025     MOVL    R2,R7                          ;BACKING STORE ADDRESS
1026     MOVZWL  G^MPW$GW_MPWPFC,R9          ;MAXIMUM NUMBER OF PAGES TO CLUSTER
1027 ;     CLRL   MPW$L_COUNT                  ;INIT COUNT OF CLUSTER TO 0
1028     BSBW    PTESCAN                        ;LOOK AT ADJACENT PTE'S FOR A CLUSTER TO WRI
1029 ;     MOVL   MPW$L_COUNT,R0                ;GET THE COUNT
1030 ;     INCL   MPW$A_SECTCLUSTERS[R0]      ;BUMP THE COUNT

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 21  
X-26 WRTMFYFAG - WRITE MODIFIED PAGES 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (5)

```

1031      .IF      DF,DEBUG
1032      .IF      EQ,DEBUG-1
1033      LOG_EVENT EVT=WRTSEC
1034      .ENDC
1035      .ENDC
1036      MOVL     @W^MPW$AL_PTE,R0      ;GET STARTING PAGE NUMBER
1037      MOVL     @W^PFN$AL_BAK[R0],R2  ;GET ITS BACKING STORE ADDRESS
1038      MOVL     @W^PFN$AL_PTE[R0],R3  ;AND ITS PAGE TABLE ENTRY ADDRESS
1039      ;
1040      ; R5 = PROCESS HEADER ADR ASSOCIATED WITH THIS SVAPTE FROM ORIGINAL GETPFNCTX CALL
1041      ;
1042      10$:     BSBW     MMG$INIBLDPKT      ;TRANSLATE BACKING STORE TO VBN AND WINDOW
1043      BRW      MPW_BLDPKT                ;GO QUEUE THE REQUEST
1044      ;
1045      ; THIS PAGE IS A SWPVBN PAGE AND IS TO BE WRITTEN BACK TO THE SWAP FILE
1046      ; RATHER THAN ITS NORMAL BACKING STORE ADDRESS
1047      ;
1048      SWPVBN:
1049      BBS      S^$SCH$V_SIP,G^$SCH$GB_SIP,40$ ;DON'T PROCESS SWAP VBN'S
1050      ; IF SWAPPER IS ACTIVE
1051      MOVZWL   PHD$W_PHVINDEX(R5),R1    ;GET PROCESS HEADER VECTOR INDEX
1052      CVTWL    @W^PHV$GL_PIXBAS[R1],R1  ;GET PROCESS INDEX
1053      BLSS     PROCESS_GONE              ;BRANCH IF PROCESS IS NO LONGER AROUND
1054      BSBW     MAP_SWPVBN                ; Convert rel. page # to PF#/VBN (WSSWP form
1055      MOVL     R3,AP                      ;STARTING PTE RANGE
1056      MOVL     R3,R10                    ;ENDING PTE RANGE
1057      MNEGL    #1,R7                     ;SET PTE$V TYPO
1058      BSBW     SCAN_DONE                 ;PUT JUST THIS PAGE IN MPW PTE ARRAY
1059      ;
1060      ; R9, R10 ASSUMED PRESERVED
1061      ;
1062      MOVQ     R9,R2                      ;R2 = BACKING STORE ADR, R3 = SVAPTE
1063      .IF      DF,DEBUG
1064      .IF      EQ,DEBUG-1
1065      LOG_EVENT EVT=WRTSWP
1066      .ENDC
1067      .ENDC
1068      BBCS     #15,@W^MPW$AW_PHVINDEX,10$ ;INDICATE SWAP VBN PAGE WRITE
1069      BRB      10$                       ;SET UP FOR BUILDPKT
1070      ;
1071      ; DO NOT PROCESS THIS MODIFIED PAGE WITH SWPVBN SET SINCE SWAPPER IS ACTIVE
1072      ;
1073      40$:     BBS      W^MPW$GL_REQIDX,-  ; Show failure to write for current
1074      W^MPW$GL_REQFAIL,45$              ; request index
1075      45$:     BRW      NEXT_MFYFAG       ;SKIP THIS MODIFIED PAGE
1076      ;
1077      .DSABL   LSB
1078      ;
1079      ; Local subroutine to convert relative page number
1080      ; in swap image to PF#/VBN format.
1081      ;
1082      MAP_SWPVBN:
1083      MOVL     @W^SCH$GL_PCBVEC[R1],R7  ; Get PCB address
1084      MOVZWL   PCB$W_APTCNT(R7),R9      ; R9 = active page table pages to skip
1085      ADDL     R2,R9                      ; R9 = relative page number in swap slot
1086      MOVAL    PCB$L_WSSWP(R7),R8       ; R8 = address of mapping pointer
1087      TSTL     (R8)                       ; Is there a PFLMAP ?

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 22  
X-26 WRTMFYFAG - WRITE MODIFIED PAGES 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (5)

```
1088      BGTR      10$           ; BR if no
1089      BEQL      NOSWAPSPACE   ; BR if swap space not allocated
1090      ADDL3     #PFLMAP$Q_PTR, (R8), R8 ; Address 1st pointer in mapping window
1091
1092 10$:    BICL3     #<1@31>, 4 (R8), R7 ; Get pointer page count
1093      SUBL2     R7, R9         ; Subtract from pages remaining to skip
1094      BLSS      13$           ; Quit if enough pages skipped
1095      BBS       #31, 4 (R8), 12$ ; Bugcheck if last pointer
1096      MOVAQ     8 (R8), R8     ; Move to next pointer
1097      BRB       10$
1098
1099 12$:    BUG_CHECK INVPFLMAP, FATAL
1100
1101                                     ; R9 contains -(excess page count)
1102 13$:    ADDL3     R7, (R8), R8 ; R8 = WSSWP of page beyond current pointer
1103      ADDL3     R9, R8, R9     ; R9 = WSSWP of SWPVBN page
1104      RSB
1105
1106 ;
1107 ; PROCESS WAS DELETED AND SWAP VBN WAS SET IN PFN DATA BASE
1108 ;
1109 PROCESS_GONE:
1110      BUG_CHECK PROCGONE, FATAL ; PROCESS NOT IN SYSTEM
1111 ;
1112 ; No swap space allocated for process
1113 ;
1114 NOSWAPSPACE:
1115      BUG_CHECK INSSWPFIL, FATAL ; INSUFFICIENT SWAP FILE SPACE
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYPAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 23  
X-26 GETPFNCTX 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYPAG.MAR;1 (6)

```

1117      .SBTTL  GETPFNCTX
1118 ;
1119 ; CALLING SEQUENCE:
1120 ;
1121 ;      BSBW   GETPFNCTX
1122 ;
1123 ; INPUTS:
1124 ;
1125 ;      RO = PFN
1126 ;
1127 ; OUTPUTS:
1128 ;
1129 ;      RO = PFN (PRESERVED)
1130 ;      R1 = TYPE OF BACKING STORE ADDRESS
1131 ;          = 0 IF PAGING FILE
1132 ;          = 1 IF SECTION ADDRESS
1133 ;          = 2 IF ILLEGAL
1134 ;          = 3 IF SWPVBN
1135 ;          = 4 if page is to be skipped based on selective MPL flush criteria
1136 ;      R2 = BACKING STORE ADDRESS OR SWPVBN
1137 ;      R3 = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE ENTRY
1138 ;      R4 = PRESERVED
1139 ;      R5 = PROCESS HEADER ADDRESS
1140 ;
1141 GETPFNCTX:
1142      MOVL   @W^PFNSAL_PTE[R0],R3      ;SYSTEM VIRTUAL ADDRESS OF PAGE TABLE ENTRY
1143 ;
1144      ASSUME MPWSC_IDLE      EQ 0
1145      ASSUME MPWSC_MAINTAIN EQ 1
1146      ASSUME MPWSC_SVAPTE   EQ 2
1147      ASSUME MPWSC_OPCCRASH EQ 3
1148      EXTZV #1,#1,W^MPW$GB_STATE,R1
1149      BNEQ  CHECK_PURGE_RANGE      ; BR if selective MPL writing in effect ?
1150 GETPFNCTX_CONT:
1151      BESS  R1,W^MPW$GL_REQFND,-    ; Show a writable page found
1152      GETPFNCTX_CONT2
1153 GETPFNCTX_CONT2:
1154      EXTZV #PFNSV_PAGTYP,#PFNSS_PAGTYP,@W^PFNSAB_TYPE[R0],R1 ;PAGE TYPE
1155 ;
1156      ASSUME PFNSC_PROCESS EQ 0
1157      ASSUME PFNSC_SYSTEM EQ 1
1158      ASSUME PFNSC_GLOBAL EQ 2
1159      ASSUME PFNSC_GBLWRT EQ 3
1160      ASSUME PFNSC_PPGTBL EQ 4
1161      ASSUME PFNSC_GPGTBL EQ 5
1162 ;
1163      CASE  R1,<-
1164          PROCESS,-                ;PROCESS PAGE
1165          SYSPHD,-                 ;SYSTEM PAGE
1166          BADTYP,-                 ;GLOBAL READ ONLY
1167          SYSPHD,-                 ;GLOBAL WRITABLE
1168          PHDR,-                   ;PROCESS PAGE TABLE
1169          SYSPHD-                   ;GLOBAL PAGE TABLE
1170      >
1171 BADTYP: BUG_CHECK BADPAGTYPE,FATAL ;BAD PAGE TYPE
1172 ;
1173 CHECK_PURGE_RANGE:

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYPAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 24  
X-26 GETPFNCTX 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYPAG.MAR;1 (6)

```

1174      .IF      DF,DEBUG
1175      .IF      EQ,DEBUG-2
1176      BLBC     G^SGN$GL_VMSD1,2$
1177      BBC      #1,G^SGN$GL_VMSD1,1$
1178      BRB      GETPFNCTX_CONT
1179 1$:     JSB      G^INI$BRK
1180 2$:
1181      .ENDC
1182      .ENDC
1183      Cmpl     R3,W^MPW$GL_SVAPTELOW ; Quick check - is SVAPTE within
1184      BLSSU    10$ ; the aggregate extent range ?
1185      Cmpl     R3,W^MPW$GL_SVAPTEHIGH
1186      BLEQU    20$
1187 10$:     MOVL   #4,R1 ; SVAPTE definitely out of range
1188      RSB ; Return with SKIP PAGE status
1189 20$:     MOVZBL W^MPW$GB_REQCNT,R1 ; SVAPTE within the aggregate extent range
1190      DECL    R1 ; If only one request, specific range is
1191      BGTR    40$ ; identical to aggregate range, so we're don
1192 30$:     MOVL   R1,W^MPW$GL_REQIDX ; Set current request index
1193      CMPB    #MPW$C_OPCCRASH,W^MPW$GB_STATE
1194      BNEQ    GETPFNCTX_CONT ; BR if not OPCCRASH request
1195
1196 35$:     PUSHAB B^CHECK_PURGE_PAGETYPE ; Force return through pagetype checking
1197      BRB      GETPFNCTX_CONT2
1198
1199 ;
1200 ; We have to check multiple SVAPTE ranges.
1201 ; Loop through the requests to see if we have a match.
1202 ;
1203 40$:     MOVAQ  W^MPW$GQ_SVAPTE[R1],R2 ; R2 = address of SVAPTE range
1204      Cmpl     R3,(R2) ; SVAPTE high enough ?
1205      BLSSU    50$ ; BR if no
1206      Cmpl     R3,4(R2) ; SVAPTE low enough ?
1207      BLEQU    30$ ; BR if yes
1208 50$:     SOBGEQ R1,40$ ; Loop until all requests examined or match
1209      MOVL   #4,R1 ; Return SKIP PAGE status
1210      RSB
1211
1212 CHECK_PURGE_PAGETYPE:
1213      TSTL    R1 ; Is this a pagefile page ?
1214      BNEQ    10$ ; BR if no
1215      MOVL   #4,R1 ; Return SKIP PAGE status
1216      RSB
1217
1218 10$:     BBSS    W^MPW$GL_REQIDX,- ; Show at least one page found
1219      W^MPW$GL_REQFND,20$ ; for this range
1220 20$:     RSB
1221
1222 SYSPHD:
1223      MOVL   G^MMG$GL_SYSPHD,R5 ;ADDRESS OF SYSTEM HEADER
1224      BRB      GOTPHDR ;JOIN THE COMMON CODE
1225
1226 PHDR:
1227      SUBL3   G^SWP$GL_BALSPT,R3,R5 ;NO. OF BYTES INTO SPT BEYOND BALSET BASE
1228      ASHL   #7,R5,R5 ;NO. OF SPT ENTRIES * 512
1229      BRB      GETPHDR ;GET PROCESS HEADER ADDRESS
1230

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

WRTMFYPAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 25  
X-26 GETPFNCTX 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYPAG.MAR;1 (6)

```
1231 PROCESS:
1232     SUBL3     G^SWP$GL_BALBASE,R3,R5 ;NO. OF BYTES BEYOND FIRST BAL SET PAGE
1233 GETPHDR:
1234     DIVL     G^SWP$GL_BSLOTSZ,R5 ;FORM PROCESS HEADER INDEX
1235     ASHL     #-9,R5,R5 ;DIVIDE BY PAGE SIZE
1236     MULL     G^SWP$GL_BSLOTSZ,R5 ;CONVERT PROCESS INDEX
1237     ROTL     #9,R5,R5 ;MULL BY BYTES PER PAGE
1238     ADDL     G^SWP$GL_BALBASE,R5 ;TO PROCESS HEADER ADDRESS
1239 GOTPHDR:
1240     MOVZWL   @W^PFN$AW_SWPVEN[R0],R2 ;IS SWPVEN SET?
1241     BNEQ     40$ ;BRANCH IF YES
1242     MOVL     @W^PFN$AL_BAK[R0],R2 ;GET BACKING STORE ADDRESS
1243
1244     ASSUME   PFN$V_GBLBAK EQ PTE$V_TYPO+1
1245     EXTZV    #PTE$V_TYPO,#2,R2,R1 ;GET BACKING STORE ADDRESS TYPE
1246     RSB
1247 40$:
1248     ASSUME   PFN$C_PROCESS EQ 0
1249     BITB     #PFN$M_PAGTYP,@W^PFN$AB_TYPE[R0] ;REQUIRE SWPVEN PAGE TO BE PROCESS
1250     BNEQ     60$ ;BRANCH IF NOT, ERROR
1251     MOVL     #3,R1 ;CODE FOR SWPVEN BACKING DESTINATION
1252     RSB
1253 60$:     BUG_CHECK BADSWPVEN,FATAL ;SWAP VBN ONLY FOR PROCESS PAGES
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 26  
X-26 PTESCAN - SCAN ADJACENT PTE'S 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (7)

```

1255      .SBTTL  PTESCAN - SCAN ADJACENT PTE'S
1256 ;
1257 ; PTESCAN, ROUTINE TO SCAN ADJACENT PAGE TABLE ENTRIES FOR TRANSITION
1258 ; PAGES THAT ARE ON THE MODIFIED PAGE LIST AND HAVE THE APPROPRIATE
1259 ; BACKING STORE ADDRESS FOR CLUSTERING.
1260 ; A SCAN IS MADE BACKWARDS FROM THE SVAPTE AND THEN FORWARDS.
1261 ;
1262 ; CALLING SEQUENCE:
1263 ;
1264 ;      BSBW      PTESCAN
1265 ;
1266 ; INPUTS:
1267 ;
1268 ;      R3 = SVAPTE TO START SCANNING FROM
1269 ;      R5 = PROCESS HEADER ADDRESS ASSOCIATED WITH THAT SVAPTE
1270 ;      R6 = INDEX TO NEXT AVAILABLE SLOT IN MODIFIED PAGE WRITER
1271 ;           PTE AND PHVINDEXT ARRAYS
1272 ;      R7 = PAGE FILE INDEX IF PAGE FILE PAGE, TYPO BIT CLEAR
1273 ;           = SECTION BACKING STORE ADDRESS IF TYPO BIT SET
1274 ;      R8 = PAGE FILE VBN TO USE IF SCANNING FOR PAGE FILE PAGES
1275 ;      R9 = MAXIMUM NUMBER OF PAGES TO FIND
1276 ;
1277 ; OUTPUTS:
1278 ;
1279 ;      R6 UPDATED TO POINT TO NEXT SLOT NOT USED
1280 ;      R8 = FIRST PAGE FILE VBN NOT USED
1281 ;      R4,R5,R7 PRESERVED
1282 ;      R0,R1,R2,R3 SCRATCHED
1283 ;      R9,R10,R11,AP SCRATCHED
1284 ;
1285 CHK_ACCESS:
1286      EXTZV   #VA$V_VPN,#VA$$_VPN,R3,R1 ;INDEX INTO SYSTEM PAGE TABLE
1287      TSTL    @W^MMG$GL_SPTBASE[R1]   ;IS THIS PAGE OF PAGE TABLE ENTRIES VALID
1288      BGEQ    SCAN_FORWARD             ;BRANCH IF NOT
1289      BRB     SCAN_NEXT1
1290 PTESCAN:
1291      MOVL    R3,AP                     ;SAVE SVAPTE
1292      EXTZV   #VA$V_VPN,#VA$$_VPN,R3,R1 ;KEEP PAGE NUMBER OF SVAPTE
1293      MNEGL   #4,R11                    ;GOING BACKWARDS LOOKING AT PTE'S
1294      BRB     SCAN_AGAIN                 ;LOOK AT NEXT PAGE TABLE ENTRY
1295 ;
1296 ; NOTE THAT R1 = PAGE NUMBER OF AN ACCESSABLE PTE
1297 ;
1298 SCAN_NEXT:
1299      ADDL3   R10,R11,R3                 ;FORM NEXT SVAPTE TO CHECK
1300      CMPZV   #VA$V_VPN,#VA$$_VPN,R3,R1 ;PTE IN SAME PAGE?
1301      BNEQ    CHK_ACCESS                 ;BRANCH IF NOT, CHECK ACCESSABILITY
1302 SCAN_NEXT1:
1303      BICL3   #^C<PTE$M_VALID !-
1304            PTE$M_TYP1 ! PTE$M TYPO !-
1305            PTE$M_BAKX>, (R3),R0        ;GET PFN IF TRANSITION PAGE
1306      BEQL    SCAN_FORWARD               ;BRANCH IF DEMAND ZERO PAGE
1307      ASHL    #-PTE$V TYPO,R0,R2        ;VALID, TYP1, TYPO ALL ZERO?
1308      BNEQ    SCAN_FORWARD               ;BRANCH IF NOT TRANSITION PAGE
1309      CMPL    R0,G^MMG$GL_MAXPFN        ;IS THIS A LEGAL PFN?
1310      BGTR    SCAN_FORWARD              ;NO (ALSO REJECTS WINDOW PAGES)
1311      CMPZV   #PFN$V_LOC,#PFN$$_LOC,-   ;SEE IF ON MODIFIED PAGE LIST

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 27  
X-26 PTESCAN - SCAN ADJACENT PTE'S 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (7)

```

1312          @W^PFNSAB_STATE[R0],#PFNSC_MFYFAGLST
1313          BNEQ   SCAN_FORWARD          ;BRANCH IF NOT ON MODIFIED PAGE LIST
1314          TSTW   @W^PFNSAW_SWPVBN[R0]  ;IF SWAP VBN PAGE,
1315          BNEQ   SCAN_FORWARD          ;DON'T USE IT
1316          CMPL   R3,@W^PFNSAL_PTE[R0]  ;CHECK FOR SPURIOUS MATCH
1317          BNEQ   SCAN_FORWARD          ;BRANCH IF SPURIOUS
1318          MOVL   @W^PFNSAL_BAK[R0],R2   ;GET BACKING STORE ADDRESS
1319          BBS    #PTE$V_TYPO,R7,60$    ;BRANCH IF SECTION ADDRESS
1320 ;
1321 ; PAGE FILE PAGE, REQUIRE ANOTHER PAGE FILE PAGE WITH SAME PAGE FILE INDEX
1322 ;
1323          BBS    #PTE$V_TYPO,R2,SCAN_FORWARD ;BRANCH IF SECTION PAGE
1324          CMPZV  #PFNS$V_PGFLX,#PFNS$S_PGFLX,R2,R7 ;SAME PAGE FILE INDEX?
1325          BEQL   SCAN_Again            ;BRANCH IF YES
1326          BRB    SCAN_FORWARD          ;NOT SAME PAGE FILE
1327 ;
1328 ; SECTION PAGE, MUST CHECK FOR SECTION BACKING STORE TYPE AND SAME BACKING STORE
1329 ;
1330 60$:      CMPL   R2,R7                  ;SAME SECTION?
1331          BNEQ   SCAN_FORWARD          ;BRANCH IF NOT
1332 SCAN_Again:
1333          MOVL   R3,R10                  ;ADDRESS OF LAST PTE CHECKED
1334          SOBGR  R9,SCAN_NEXT           ;BRANCH IF MORE PAGE FILE VBN'S TO USE
1335 ;      INCL   L^MPW$L_BACKUPFAIL(R11) ;COUNT FAILURE FOR EITHER DIRECTION
1336          BRB    SCAN_DONE              ;ALLOCATED PAGE FILE EXHAUSTED,
1337 ;      ;WRITE THE CLUSTER
1338 ;
1339 ; PTE SCAN CANNOT PROCEED IN CURRENT DIRECTION, SWITCH TO SCAN FORWARD
1340 ; IF NOT ALREADY SCANNING FORWARD
1341 ;
1342 SCAN_FORWARD:
1343          MNEGL  R11,R11                  ;SWITCH DIRECTION OF PTE SCAN
1344          BLSS   SCAN_DONE              ;PTE SCAN COMPLETE IF ALREADY SCANNED FORWARD
1345          MOVL   AP,R3                    ;GET STARTING PTE ADDRESS
1346          MOVL   R10,AP                  ;RECORD THIS AS STARTING PTE ADDRESS
1347 ;      ;SINCE IT IS LEQU THAN STARTING PTE
1348          MOVL   R3,R10                  ;START FORWARD FROM ORIGINAL START PTE
1349          BRB    SCAN_NEXT              ;CONTINUE THE PTE SCAN
1350 ;
1351 ; AP = SVAPTE, R10 IS THE OTHER SVAPTE, NOT NECESSARILY IN ORDER
1352 ; R7 = PAGE FILE INDEX OR SECTION BACKING STORE ADDRESS
1353 ; IF PTE$V_TYPO IS SET, NO BACKING STORE MANIPULATION
1354 ; R9 PRESERVED FROM HERE ON
1355 ; R10 PRESERVED FROM HERE ON IF AP LEQU R10
1356 ;
1357 SCAN_DONE:
1358          CMPL   AP,R10                  ;GET PTE ADDRESSES IN ORDER
1359          BLEQU  40$                      ;BRANCH IF R10 IS TOP OF RANGE
1360          MOVL   R10,R0                  ;SAVE BOTTOM OF RANGE
1361          MOVL   AP,R10                  ;HIGH END OF RANGE
1362          MOVL   R0,AP                  ;LOW END OF RANGE
1363 ;
1364 ; AP = FIRST PTE ADDRESS, R10 = LAST PTE ADDRESS INCLUSIVE
1365 ; ALL PFN'S IN THESE PTE'S ARE ON THE MODIFIED PAGE LIST AND
1366 ; ARE FROM THE SAME PAGE FILE OR SECTION
1367 ;
1368 40$:      BICL3  #^C<PTE$M_PFN>,(AP)+,R11 ;GET PAGE FRAME NUMBER

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

WRTMFYPAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 28  
X-26 PTESCAN - SCAN ADJACENT PTE'S 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYPAG.MAR;1 (7)

```

1369      MOVL      R11,@W^MPW$AL_PTE[R6]      ;STORE PFN IN PTE ARRAY
1370      MOVW      PHD$W_PHVINDEX(R5),@W^MPW$AW_PHVINDEX[R6] ;STORE THE ASSOCIATED
1371                                     ;PROCESS HEADER VECTOR INDEX
1372      INCL      R6                          ;NEXT PTE INDEX
1373      INCL      MPW$L_COUNT                 ;COUNT PAGES IN THIS CLUSTER
1374      BBS       #PTE$V_TYPO,R7,65$        ;BRANCH IF SECTION PAGE
1375      BICL3     #^C<PFN$M_PGFLVBN!PFN$M_PRCPGFLX>,- ; Fetch page file VBN and
1376      @W^PFN$AL_BAK[R11],R2              ; PROCESS page file index
1377      BITL      #PFN$M_PGFLVBN,R2         ; Backing store already assigned ?
1378      BNEQ      100$                       ; BR if yes; BUGCHECK
1379      INSV      R7,#PFN$V_PGFLX,-        ; Insert SYSTEM page file index
1380      #PFN$S_PGFLX,R2                    ; into its backing store field
1381      BISL3     R2,R8,-                    ; Merge allocated VBN into the
1382      @W^PFN$AL_BAK[R11]                 ; backing store address
1383      INCL      R8                          ;NEXT PAGE FILE VBN
1384 65$:      MOVL      R11,R0                ;PFN TO CONVENTIONAL REGISTER
1385      MOVZBL    #PFN$C_MFYPAGLST,R2       ;INDEX TO MODIFIED PAGE LIST
1386      JSB       G^MMG$REMPFN             ;REMOVE PAGE FROM MODIFIED PAGE LIST
1387      BICB3     #<PFN$M_MODIFY ! PFN$M_LOC>,- ; SHUT OFF MODIFY BIT
1388      @W^PFN$AB_STATE[R0],R1
1389      BISB3     #PFN$C_WRTINPROG,R1,@W^PFN$AB_STATE[R0] ;SET WRITE IN PROGRESS
1390      INCW      @W^PFN$AW_REFcnt[R0]      ;AND COUNT AN I/O REFERENCE
1391      CMPZV     #PFN$V_PAGTYP,#PFN$S_PAGTYP,- ; IF PROCESS PAGE TABLE PAGE
1392      @W^PFN$AB_TYPE[R0],#PFN$C_PPGTBL
1393      BNEQ      80$
1394      MOVZWL    PHD$W_PHVINDEX(R5),R1     ; THEN MUST COUNT A PROCESS HEADER REF
1395      INCW      @W^PHV$GL_REFcBAS[R1]
1396 80$:      CML      AP,R10                ;DONE LAST PTE IN RANGE?
1397      BLEQU     40$                       ;BRANCH IF MORE TO DO
1398      RSB
1399
1400 100$:     BUG_CHECK MODRELNBAK,FATAL    ;BACKING STORE VBN FOR MODIFIED PAGE
1401

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 29  
X-26 PURGEMPL - Setup to selectively flush pa 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (

```
1403      .SBTTL PURGEMPL - Setup to selectively flush pages from MPL
1404 ;++
1405 ; Functional description:
1406 ;
1407 ;     This routine accepts requests for selective flushing, or purging,
1408 ;     of the modified page list and does the necessary setup for
1409 ;     the mainline modified page writing code.
1410 ;
1411 ; CALLING SEQUENCE:
1412 ;
1413 ;     JSB      MMG$PURGEMPL
1414 ;
1415 ; INPUT PARAMETERS:
1416 ;
1417 ;     R0<7:0> = Request type
1418 ;
1419 ;           0      Reserved
1420 ;
1421 ;           1      MAINTAINence purge request
1422 ;
1423 ;           2      SVAPTE range request
1424 ;                 Low/high SVAPTE in R1/R2
1425 ;
1426 ;           3      OPCCRASH purge request
1427 ;
1428 ;     R0<31:8> = Request-type specific modifiers
1429 ;
1430 ;     R1      = Request-type specific parameter
1431 ;     R2      = Request-type specific parameter
1432 ;
1433 ; IMPLICIT INPUTS:
1434 ;
1435 ;     IPL = SYNCH, MMG spinlock held
1436 ;
1437 ; OUTPUT PARAMETERS:
1438 ;
1439 ;     R0, R1, R2 destroyed
1440 ;
1441 ; IMPLICIT OUTPUTS:
1442 ;
1443 ;     Selective modified page list flushing has been initiated
1444 ;
1445 ; COMPLETION CODES:
1446 ;
1447 ;     NONE
1448 ;
1449 ; SIDE EFFECTS:
1450 ;
1451 ;     NONE
1452 ;
1453 ;--
1454
1455      UNIVERSAL_SYMBOL      MMG$PURGEMPL
1456 ;MMG$PURGEMPL::
1457      .IF      DF,DEBUG
1458      .IF      EQ,DEBUG-2
1459      BLBC     G^SGN$GL_VMSD1,10$
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 30  
X-26 PURGEMPL - Setup to selectively flush pa 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (

```

1460      BBC      #1,G^SGN$GL_VMSD1,5$
1461      BRW      WAKE MPW
1462 5$:    JSB      G^INI$BRK
1463 10$:
1464      .ENDC
1465      .ENDC
1466
1467      ASSUME   MPW$C_IDLE      EQ 0
1468      ASSUME   MPW$C_MAINTAIN EQ 1
1469      ASSUME   MPW$C_SVAPTE   EQ 2
1470      ASSUME   MPW$C_OPCCRASH EQ 3
1471      CASE    R0,<-
1472      BADREQTYPE,-                ; Reserved/illegal request
1473      MAINTAIN,-                ; MAINTAINence purge request
1474      SVAPTE,-                    ; SVAPTE range request
1475      OPCCRASH,-                ; OPCCRASH purge request
1476      >,TYPE=B
1477 BADREQTYPE:
1478      BUG_CHECK      WRTPGSBAK,FATAL
1479
1480      .ENABLE LSB
1481 OPCCRASH:
1482      MOVB      #MPW$C_OPCCRASH,-    ; Set MPL purge state
1483      W^MPW$GB_STATE
1484      CLR      W^MPW$GB_REQCNT
1485      CLRL     R0                    ; OPCCRASH preempts all other requests,
1486      MOV      #<1@31>,R1            ; so show no outstanding requests
1487      ADDL3    #<<1@30>-4>,R1,R2    ; SVAPTE low = 80000000
1488      BBCCI    S^#SCH$V_TCD,G^SCH$GB_SIP,45$ ; SVAPTE high = BFFFFFFC
1489      BRB      45$
1490
1491 40$:     BSBW      CHECK_SVAPTE      ; Check for duplicate request
1492      ; (Returns to caller if duplicate found)
1493      Cmpl     R1,W^MPW$GL_SVAPTELOW ; Set lowest SVAPTE seen so far
1494      BGEQU   41$
1495      MOV      R1,W^MPW$GL_SVAPTELOW
1496 41$:     Cmpl     R2,W^MPW$GL_SVAPTEHIGH ; Set highest SVAPTE seen so far
1497      BLEQU   42$
1498      MOV      R2,W^MPW$GL_SVAPTEHIGH
1499 42$:     CMPB     R0,#MPW$C_MAXREQCNT ; Exceeded maximum number of pending request
1500      BLSSU   50$                    ; BR if no to join common exit code
1501 ;
1502 ; The maximum number of concurrent SVAPTE requests has been exceeded. Since all
1503 ; callers of this routine can tolerate premature reactivation (from the RWMPB wait
1504 ; state), we simply drop all current requests on the floor. Note that the MPWBUSY
1505 ; resource will be declared available during the MPW thread termination processing.
1506 ;
1507 ; This strategy assumes that the request list does not artificially fill up with
1508 ; duplicate requests (prevented by CHECK_SVAPTE), that the probability of exceeding
1509 ; the maximum request count is very small, and that the probability of having this
1510 ; situation immediately recur (due to unsatisfied, pending requests) is also very sm
1511 ;
1512      CLRL     R0                    ; Show no outstanding requests
1513      CLR      W^MPW$GB_REQCNT
1514      BRB      45$                    ; Make this request the sole request
1515
1516 SVAPTE:

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 31  
X-26 PURGEMPL - Setup to selectively flush pa 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (

```

1517      .IF      DF,PERF_STATS
1518      INCL     G^PMS$GL_RESERVED18
1519      .ENDC
1520      CMPB     #MPW$C_SVAPTE,-           ; Is higher rank request is pending ?
1521      W^MPW$GB_STATE
1522      BLSSU   100$                       ; BR if yes (ignore this request)
1523      MOVB     #MPW$C_SVAPTE,-           ; Set MPL purge state
1524      W^MPW$GB_STATE
1525      MOVZBL  W^MPW$GB_REQCNT,R0         ; R0 = current number of pending requests
1526      BNEQ    40$                         ; BR if there are any, else do things the ea
1527 45$:      MOVQ    R1,W^MPW$GL_SVAPTELOW ; Aggregate extent range is identical to req
1528 50$:      MOVQ    R1,W^MPW$GQ_SVAPTE[R0] ; Set extent range for this request
1529      INCB     W^MPW$GB_REQCNT           ; Show one more pending request
1530 ;      BRB      WAKE_MPW
1531 ;
1532 WAKE_MPW:
1533      CLRL    G^SCH$GL_MFYLOLIM         ; Indicate MPL flush desired
1534      CLRL    G^SCH$GL_MFYLIM
1535 100$:     RSB
1536
1537 MAINTAIN:
1538      .IF      DF,PERF_STATS
1539      INCL     G^PMS$GL_RESERVED17
1540      .ENDC
1541      MOVL    G^SCH$GL_MFYLOSV,R2       ; Assume flushing to MPW_LOLIMIT
1542      BBC     #MPW$V_LOLIMIT,R0,150$    ; BR if assumption is correct
1543      CML    R1,R2                       ; Parameter > MPW_LOLIMIT ?
1544      BLEQU   150$                       ; BR if no - ignore parameter
1545      MOVL    R1,R2                       ; Else use parameter
1546 150$:     CMPB     #MPW$C_MAINTAIN,-     ; Is same or higher rank request pending ?
1547      W^MPW$GB_STATE
1548      BLSSU   200$                       ; BR if higher rank pending (ignore this req
1549      BNEQ    175$                       ; BR if no previous MAINTAIN request
1550      CML    R2,G^SCH$GL_MFYLIM         ; Is requested new limit lower than old limi
1551      BGEQU   200$                       ; BR if not (ignore this request)
1552 175$:     MOVB     #MPW$C_MAINTAIN,-     ; Set MPL purge state
1553      W^MPW$GB_STATE
1554      MOVL    R2,G^SCH$GL_MFYLIM         ; Start modified page writing
1555      MOVL    R2,G^SCH$GL_MFYLOLIM      ; Set new low limit
1556      BBCCI   S^#SCH$V_TCD,G^SCH$GB_SIP,200$ ; Enable threshold checking
1557 200$:     RSB
1558
1559      .DISABLE LSB
1560
1561 ; Check for duplicate SVAPTE request. These can occur if a process makes such
1562 ; a request, is placed into RWMPB, and then reawakened when the modified page
1563 ; count reaches MPW_LOWAITLIMIT, but before all requested pages have been
1564 ; written.
1565 ;
1566 ; R0 = current request count (known to be > 0)
1567 ; R1,R2 = SVAPTE range for which to search
1568 ; (SP) = return address (in this module)
1569 ; 4(SP) = return address of caller of MPW$PURGEMPL
1570 ;
1571 ; If a duplicate request is found, control returns directly to the caller
1572 ; of MPW$PURGEMPL.
1573

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYPAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 32  
X-26 PURGEMPL - Setup to selectively flush pa 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYPAG.MAR;1 (

```
1574 CHECK_SVAPTE:
1575     PUSHL    R0                ; Save current request count
1576 10$:     SOBGEQ  R0,50$
1577 20$:     POPL     R0                ; Restore current request count
1578     RSB
1579 50$:     PUSHAQ  W^MPW$GQ_SVAPTE[R0] ; Low limit matches ?
1580     CMPL     R1,@(SP)+
1581     BNEQ     10$                ; No - examine next entry
1582     PUSHAQ  W^MPW$GQ_SVAPTE+4[R0] ; High limit matches ?
1583     CMPL     R2,@(SP)+
1584     BNEQ     10$                ; No - examine next entry
1585     ADDL2    #8,SP              ; Remove saved R0 and return address
1586     RSB
1587
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 33  
X-26 MMG\$MPW\_END - Terminate current MPW thre 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (

```

1589      .SETTL MMG$MPW_END - Terminate current MPW thread
1590 ;++
1591 ; Functional description:
1592 ;
1593 ;     This routine does all of the necessary end of thread
1594 ;     cleanup for the modified page writer.
1595 ;
1596 ; CALLING SEQUENCE:
1597 ;
1598 ;     JSB      MMG$MPW_END
1599 ;
1600 ; INPUT PARAMETERS:
1601 ;
1602 ;     NONE
1603 ;
1604 ; IMPLICIT INPUTS:
1605 ;
1606 ;     IPL = SYNCH, MMG spinlock held
1607 ;
1608 ; OUTPUT PARAMETERS:
1609 ;
1610 ;     R0, R1, R2 destroyed
1611 ;
1612 ; IMPLICIT OUTPUTS:
1613 ;
1614 ;     End of thread processing complete
1615 ;     MPW state correctly set
1616 ;
1617 ; COMPLETION CODES:
1618 ;
1619 ;     NONE
1620 ;
1621 ; SIDE EFFECTS:
1622 ;
1623 ;     NONE
1624 ;
1625 ;--
1626 ;
1627 ;-----
1628 ;
1629 ; IOCNT REQFAIL STATE      Declare      Threshold      HI/LO
1630 ; -----      -----      -----      Limits      WAKEPEN MPW SIP
1631 ; / = 0      x      (Outstanding I/O)      Unchanged Unchanged      N      (1)
1632 ;           MAINTAIN      Y(1)      N
1633 ;           SVAPTE      Y(2)      N
1634 ;           OPCCRASH      Y(2)      N
1635 ;
1636 ; == 0      == 0      (Normal termination)      (Enabled)      Restored      Y      0
1637 ;                                     New state = IDLE
1638 ;           MAINTAIN      Y(1)      N
1639 ;           SVAPTE      Y(2)      N
1640 ;           OPCCRASH      N      Y(2)
1641 ;
1642 ; == 0      /= 0      (Abnormal termination)      Enabled      Unchanged      N      0
1643 ;                                     New state = old state
1644 ;           MAINTAIN      Y(1)      N
1645 ;           SVAPTE      Y(2)      N

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYYPAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 34  
X-26 MMG\$MPW\_END - Terminate current MPW thre 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYYPAG.MAR;1 (

```

1646 ;                               Enabled   Restored   Y       0
1647 ;                               New state = IDLE
1648 ;           OPCCRASH           N           Y(2)
1649 ;
1650 ; Notes:
1651 ;           (1) Event is conditionally declared based upon comparison of modified
1652 ;           page count and MPW_LOWAITLIMIT.
1653 ;           (2) Event is unconditionally declared.
1654 ; =====
1655 MMG$MPW_END::
1656     MOVL      G^SCH$GL_MFYCNT,R6      ; Get modified page count for compares
1657     TSTB     W^MPW$GB_IOCNT
1658     BEQL     10$                       ; BR if no outstanding MPW I/O
1659     ASSUME   MPW$C_SVAPTE      EQ 2
1660     ASSUME   MPW$C_OPCCRASH   EQ 3
1661     BBS      #1,W^MPW$GB_STATE,4$     ; BR if state is SVAPTE or OPCCRASH
1662     CMLPL    R6,G^MPW$GL_LOWAITLIM   ; At or below low busy wait threshold ?
1663     BGTRU    5$                       ; BR if no
1664 4$:     MOVZBL #RSN$ MPWBUSY,R0      ; Set resource wait that is satisfied
1665     JSB      G^SCH$RAVAIL          ; Declare modified page writer done
1666 5$:     RSB
1667                               ; No outstanding I/O
1668 10$:    TSTL      W^MPW$GL_REQFAIL   ; Any failures to write pages ?
1669     BNEQ     100$                     ; BR if yes
1670     BBCCI    S^#SCH$V_TCD,G^SCH$GB_SIP,12$ ; Enable threshold checking
1671 12$:    MOVL      G^SCH$GL_MFYLIMSV,G^SCH$GL_MFYLIM ; Restore high limit
1672     MOVL      G^SCH$GL_MFYLOSV,G^SCH$GL_MFYLOLIM ; Restore low limit
1673     MOVL      G^CTL$GL_PCB,R0        ; Get PCB address
1674     BBSS     #PCB$V_WAKEPEN,PCB$S_STS(R0),15$ ; Set wake pending to force
1675                               ; swap schedule re-evaluation
1676 15$:
1677     ASSUME   MPW$C_IDLE      EQ 0
1678     ASSUME   MPW$C_MAINTAIN  EQ 1
1679     ASSUME   MPW$C_SVAPTE   EQ 2
1680     ASSUME   MPW$C_OPCCRASH EQ 3
1681
1682     MOVZBL    W^MPW$GB_STATE,R0      ; Fetch MPW state
1683     ASSUME   MPW$GB_REQCNT EQ MPW$GB_STATE+1
1684     CLRW     W^MPW$GB_STATE      ; Set state to IDLE; show zero requests
1685     CASE     R0,<17$,30$,40$,20$> ; BR on MPW state
1686 17$:    BUG_CHECK      WRTPG$BAK,FATAL
1687
1688 20$:    MOVZBL #RSN$ M$EMPTY,R0      ; Set resource wait that is satisfied
1689     BRB      45$
1690 30$:    CMLPL    R6,G^MPW$GL_LOWAITLIM ; At or below low busy wait threshold ?
1691     BGTRU    50$                       ; BR if no
1692 40$:    MOVZBL    #RSN$ MPWBUSY,R0      ; Set resource wait that is satisfied
1693 45$:    JSB      G^SCH$RAVAIL          ; Declare modified page writer done
1694 50$:    BBCCI    S^#SCH$V_MPW,G^SCH$GB_SIP,60$ ;MODIFIED PAGE WRITER INACTIVE
1695 60$:    RSB
1696
1697 100$:   BBCCI    S^#SCH$V_TCD,G^SCH$GB_SIP,105$ ; Enable threshold checking again
1698 105$:
1699     ASSUME   MPW$C_IDLE      EQ 0
1700     ASSUME   MPW$C_MAINTAIN  EQ 1
1701     ASSUME   MPW$C_SVAPTE   EQ 2
1702     ASSUME   MPW$C_OPCCRASH EQ 3

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYPAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 35  
X-26 MMG\$MPW\_END - Terminate current MPW thre 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYPAG.MAR;1 (

```

1703
1704     CASE      W^MPW$GB_STATE,-      ; BR on MPW state
1705     <107$,30$,110$,12$>,-
1706     TYPE=B
1707 107$:  BUG_CHECK      WRTPGSBAK,FATAL
1708
1709 110$:  PUSHAB  B^40$      ; Do SVAPTE request cleanup
1710 ;     BRB      PURGEMPL_CLEANUP      ; Call routine by falling through...
1711

1712     .SBTTL  PURGEMPL_CLEANUP - Remove any DEAD MPL purge requests
1713 ;++;
1714 ; Functional description:
1715 ;
1716 ;     This routine removes any DEAD MPL purge requests from
1717 ;     the current set of requests. A DEAD request is one for
1718 ;     which no pages were found in the current invocation of
1719 ;     the modified page writer.
1720 ;
1721 ; CALLING SEQUENCE:
1722 ;
1723 ;     JSB      MMG$PURGEMPL_CLEANUP
1724 ;
1725 ; INPUT PARAMETERS:
1726 ;
1727 ;     bit masks of requests with processed pages
1728 ;
1729 ; IMPLICIT INPUTS:
1730 ;
1731 ;     IPL = SYNCH, MMG spinlock held
1732 ;
1733 ; OUTPUT PARAMETERS:
1734 ;
1735 ;     R0, R1, R2 destroyed
1736 ;
1737 ; IMPLICIT OUTPUTS:
1738 ;
1739 ;     NONE
1740 ;
1741 ; COMPLETION CODES:
1742 ;
1743 ;     NONE
1744 ;
1745 ; SIDE EFFECTS:
1746 ;
1747 ;     NONE
1748 ;
1749 ;--
1750
1751 MMG$PURGEMPL_CLEANUP::
1752     .IF      DF,DEBUG
1753     .IF      EQ,DEBUG-2
1754     BLBC    G^SGN$GL_VMSD1,2$
1755     BBC     #1,G^SGN$GL_VMSD1,1$
1756     RSB
1757 1$:      JSB      G^INI$BRK
1758 2$:
1759     .ENDC

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

WRTMFYFAG - WRITE MODIFIED PAGES 10-MAY-1989 16:36:29 VAX MACRO V5.0-8 Page 36

X-26 PURGEMPL\_CLEANUP - Remove any "dead" MPL 5-APR-1989 11:28:29 [SYS.SRC]WRTMFYFAG.MAR;1 (

```

1760      .ENDC
1761      CMPB      #MPW$C_SVAPTE,-          ; If current state is not SVAPTE,
1762      W^MPW$GB_STATE
1763      BNEQ      10$                      ; just return quietly
1764      MCOML     W^MPW$GL_REQFND,R0       ; In R0, form bit mask of requests that
1765      BICL3     R0,W^MPW$GL_REQFAIL,R0   ; are to remain pending
1766      MOVZBL    W^MPW$GB_REQCNT,R2      ; Get request count
1767      DECL      R2                        ; Convert to maximum index
1768      BNEQ      20$                      ; BR if more than one pending request
1769      BLBS      R0,10$                   ; If sole entry should be kept, just return
1770 5$:
1771      ASSUME    MPW$GB_REQCNT EQ MPW$GB_STATE+1
1772      CLRW      W^MPW$GB_STATE           ; Set state to IDLE; show zero requests
1773 10$:
1774      RSB
1775 20$:
1776 30$:      CLRL      R1
1777      BBS       R1,R0,50$               ; Next entry if this one should be kept
1778      MOVQ      W^MPW$GQ_SVAPTE[R2],-   ; Move last pending request to new slot
1779      W^MPW$GQ_SVAPTE[R1]
1780      CLRQ      W^MPW$GQ_SVAPTE[R2]
1781      BBC       R2,R0,40$               ; If old bit was clear, nothing to do
1782      BBSS     R1,R0,40$               ; Else set new bit
1783 40$:      DECL      R2                 ; Show one less pending request
1784 50$:      AOBLEQ    R2,R1,30$
1785      ; R2 = # pending requests - 1
1786      ADDB3     #1,R2,W^MPW$GB_REQCNT    ; Update pending request count
1787      BEQL      5$                       ; BR if no remaining, pending requests
1788      RSB
1789      .END

```

### 3 SYSQIOREQ.LIS

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 0  
Table of contents

|      |      |   |
|------|------|---|
| (2)  | 126  | DECLARATIONS                                      |
| (3)  | 220  | QIO ERROR AND EXCEPTION HANDLING ROUTINES         |
| (4)  | 266  | QUEUE I/O REQUEST                                 |
| (5)  | 717  | BUILD I/O PACKET FOR PAGE READ/WRITE              |
| (6)  | 865  | COMPLETE I/O OPERATION                            |
| (7)  | 950  | QUEUE I/O PACKET TO DRIVER                        |
| (8)  | 974  | EXE\$ALTQUEPKT - Call driver ALTSTART entry point |
| (9)  | 1114 | QUEUE I/O PACKET TO ACP                           |
| (9)  | 1192 | EXE\$QXQPPKT - QUEUE I/O PACKET TO XQP            |
| (10) | 1234 | INSERT I/O PACKET IN UNIT QUEUE                   |
| (11) | 1322 | INSERT I/O PACKET IN QUEUE BY PRIORITY            |

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 1  
X-25 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (1)

```
1
2      .TITLE  SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE
3      .IDENT  'X-25'
4
5 ;
6 ;*****
7 ;*
8 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 ;*  ALL RIGHTS RESERVED.
11 ;*
12 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 ;*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 ;*  TRANSFERRED.
18 ;*
19 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 ;*  CORPORATION.
22 ;*
23 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 ;*
26 ;*
27 ;*****
28 ;
29 ;++
30 ;
31 ;  AUTHOR:
32 ;
33 ;      D. N. CUTLER,    14-JUN-76
34 ;
35 ;  FACILITY:
36 ;
37 ;      SYSTEM SERVICE QUEUE I/O REQUEST
38 ;
39 ;  MODIFIED BY:
40 ;
41 ;      X-25      EMB0403      Ellen M. Batbouta      24-Apr-1989
42 ;              Preserve R5 across the call to the driver's alternate
43 ;              STARTIO routine in EXE$ALTQUEPKT.
44 ;
45 ;      X-24      EMB0356      Ellen M. Batbouta      17-Aug-1988
46 ;              When checking for device affinity in EXE$ALTQUEPKT,
47 ;              a special check must be made for affinity to the primary,
48 ;              since primary affinity is specified by the value, zero..
49 ;
50 ;      X-23      EMB0337      Ellen M. Batbouta      15-Aug-1988
51 ;              Add a new busy bit, ALTBSY, and wait queue to the UCB
52 ;              for the alternate startio routine, EXE$ALTQUEPKT. This
53 ;              will maintain the correct ordering of I/Os to the driver
54 ;              when a CPU switch must be made because of device affinity.
55 ;
56 ;      X-22      RNG5022      Rod Gamache           7-Apr-1988
57 ;              Make I/O's thru FINISHIO/ABORTIO complete synchronously.
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 2  
X-25 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (1)

```

58 ;
59 ;      X-21  RNG5021      Rod Gamache      31-Jan-1988
60 ;      Don't access POST queue directly - use SMP$IOPOST_IRP.
61 ;
62 ;      X-20  ACG0544      Andrew C. Goldstein,  6-Jan-1988  14:23
63 ;      Remove taking of forklock in EXE$QIOACPPKT
64 ;
65 ;      X-19  RNG5019      Rod Gamache      24-Apr-1987
66 ;      Add support of unmodified device drivers by checking
67 ;      UCB$B_FLCK for a FIPL vs. a FLCK.
68 ;
69 ;      X-18  SJF          Stu Farnham      20-Mar-1987
70 ;      Fix broken branch.
71 ;
72 ;      X-17  SF04002      Stephen Fiorelli     07-Mar-1987
73 ;      Use vms standard name for system_primitives_data.
74 ;
75 ;      X-16  WCT0032      Ward C. Travis      19-Feb-1987
76 ;      Update remaining old lookaside listhead references
77 ;      to reflect that they are now interlocked queues.
78 ;
79 ;      X-15  RNG5015      Rod Gamache      15-Jan-1987
80 ;      Make some performance optimizations.
81 ;
82 ;      X-14  WMC0014      Wayne Cardoza      06-Jan-1987
83 ;      Change some JSBs to BSBWs.
84 ;
85 ;      X-13  ROW73864      Ralph O. Weber      04-DEC-1986 15:35
86 ;      Change IOC$VERIFYCHAN part of the SQIO system service so that
87 ;      it returns SS$_IVCHAN for unassigned channels, not SS$_NOPRIV.
88 ;
89 ;      X-12  ACG0525      Andrew C. Goldstein,  27-Aug-1986  15:47
90 ;      Decrement process I/O count before queuing packet in BLDPKT.
91 ;
92 ;      X-11  RNG0011      Rod Gamache      29-Oct-1986
93 ;      Fix branch errors.
94 ;
95 ;      X-10  RNG0010      Rod Gamache      22-Oct-1986
96 ;      Make all AQB queues interlocked queues - implies that
97 ;      entries can only be inserted on the tail of the AQB queue.
98 ;
99 ;      X-9   SSA0001      Stan Amway      13-Oct-1986
100 ;      Add entry point EXE$BLDPKTMPW.
101 ;
102 ;      X-8   RNG0008      Rod Gamache      30-Sep-1986
103 ;      Make sure mods to PCB$W_DIOCNT & BIOCNT are interlocked.
104 ;
105 ;      X-7   RNG0007      Rod Gamache      24-Sep-1986
106 ;      Interlock access to PCB$L_EFCS.
107 ;      Don't check packet type when going thru EXE$ALTQOEPKT.
108 ;      Save R5 around calls to IOCSINITIATE.
109 ;
110 ;      X-4   SJF          Stu Farnham      1-Jul-1986
111 ;      Use per-CPU IOPOST queue
112 ;
113 ;      V04-003 RNG4003      Rod Gamache      4-Jun-1986
114 ;      Fix branch destination errors.

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 3  
X-25 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (1)

```
115 ;  
116 ;      V04-001 SF04001      Stephen Fiorelli      24-Oct-1985  
117 ;      System_service macro used to declare entry point  
118 ;      and build system service descriptor block.  
119 ;      Added $SYSVECTORDEF.  
120 ;  
121 ;      V04-001 ACG0467      Andrew C. Goldstein,  12-Sep-1984  17:33  
122 ;      Fix protection holes in QIO device protection check  
123 ;  
124 ;--
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 4  
X-25 DECLARATIONS 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (2)

```
126      .SBTTL  DECLARATIONS
127
128 ;
129 ; MACRO LIBRARY CALLS
130 ;
131
132      $ACBDEF      ;DEFINE ACB OFFSETS
133      $AOBDEF      ;DEFINE AOB OFFSETS
134      $CADEF       ;DEFINE CONDITIONAL ASSEMBLY PARAMETERS
135      $CCBDEF      ;DEFINE CCB OFFSETS
136      $CDRPDEF     ;DEFINE CDRP OFFSETS
137      $CPUDEF      ;DEFINE CPU DATABASE OFFSETS
138      $DDBDEF      ;DEFINE DDB OFFSETS
139      $DDTDEF      ;DEFINE DDT OFFSETS
140      $DEVDEF      ;DEFINE DEV VALUES
141      $DYNDEF      ;DEFINE DATA STRUCTURE TYPE CODES
142      $F11BDEF     ;DEFINE F11BXQP OFFSETS
143      $IODEF       ;DEFINE IO FUNCTION CODES
144      $IPLDEF      ;DEFINE INTERRUPT PRIORITY LEVELS
145      $IRPDEF      ;DEFINE IRP OFFSETS
146      $PCBDEF      ;DEFINE PCB OFFSETS
147      $PHDDEF      ;DEFINE PHD OFFSETS
148      $PRDEF       ;DEFINE PROCESSOR REGISTERS
149      $PRIDEF      ;DEFINE PRIORITY CLASS INCREMENTS
150      $PRVDEF      ;DEFINE PRIVILEGE BITS
151      $PSLDEF      ;DEFINE PROCESSOR STATUS FIELDS
152      $RSNDEF      ;DEFINE RESOURCE WAIT NUMBERS
153      $SECDEF      ;DEFINE SEC OFFSETS
154      $SSDEF       ;DEFINE STATUS VALUES
155      $$SYSVECTORDEF ;DEFINE SYSTEM SERVICE VECTOR OFFSETS
156      $$SYSTEM_PRIM_DATADEF ;DEFINE SYSTEM_PRIMITIVES DATA AREA
157      $UCBDEF      ;DEFINE UCB OFFSETS
158      $VCBDEF      ;DEFINE VCB OFFSETS
159      $WCBDEF      ;DEFINE WINDOW CONTROL BLOCK OFFSETS
160
161 ;
162 ; LOCAL SYMBOLS
163 ;
164 ; ARGUMENT LIST OFFSET DEFINITIONS
165 ;
166
167 EFN=4      ;EVENT FLAG NUMBER
168 CHAN=8    ;I/O CHANNEL NUMBER
169 FUNC=12   ;I/O FUNCTION CODE
170 IOSE=16   ;ADDRESS OF I/O STATUS BLOCK
171 ASTADR=20 ;ADDRESS OF AST SERVICE ROUTINE
172 ASTPRM=24 ;AST SERVICE ROUTINE PARAMETER
173 P1=28    ;FIRST FUNCTION DEPENDENT PARAMETER
174 P2=32    ;SECOND FUNCTION DEPENDENT PARAMETER
175 P3=36    ;THIRD FUNCTION DEPENDENT PARAMETER
176 P4=40    ;FOURTH FUNCTION DEPENDENT PARAMETER
177 P5=44    ;FIFTH FUNCTION DEPENDENT PARAMETER
178 P6=48    ;SIXTH FUNCTION DEPENDENT PARAMETER
179
180 ;
181 ; FUNCTION DECISION TABLE OFFSET DEFINITIONS
182 ;
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 5  
X-25 DECLARATIONS 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (2)

```
183
184 LEGAL=0                ;LEGAL FUNCTION MASK
185 IOTYPE=8              ;I/O FUNCTION TYPE MASK
186 FDTACT=16            ;ACTION ROUTINE MASKS
187
188 ;
189 ; TABLES FOR DETERMINING THE ACCESS DESIRED, BASED UPON THE I/O FUNCTION
190 ; CODE. THIS IS NECESSARY FOR THE FIRST TIME THROUGH PROTECTION CHECK DONE
191 ; ON SHARABLE, NON-MOUNTABLE (NON-FILES ORIENTED) DEVICES.
192 ;
193
194 .MACRO ACCMSK CODES
195 MASKL = 0
196 MASKH = 0
197
198 .IRP X,<CODES>
199 .IF GT <IO$_'X&IO$_VIRTUAL>-31
200 MASKH = MASKH!<1@<<IO$_'X&IO$_VIRTUAL>-32>>
201 .IFF
202 MASKL = MASKL!<1@<IO$_'X&IO$_VIRTUAL>>
203 .ENDC; GT <IO$_'X&IO$_VIRTUAL>-31
204 .ENDR ; X,<CODES>
205 .LONG MASKL,MASKH
206 .ENDM ACCMSK
207
208 DECLARE_PSECT EXEC$NONPAGED_DATA
209 READ_ACCESS:
210 ACCMSK <READPBLK, READLBLK, READVBLK, -
211 READHEAD, READTRACKD, REREADN, REREADP, -
212 READPROMPT, TTYREADALL, TTYREADPALL>
213
214 WRITE_ACCESS:
215 ACCMSK <WRITEPBLK, WRITELBLK, WRITEVBLK, -
216 WRITECHECK, WRITECHECKH, -
217 WRITEHEAD, WRITETRACKD, WRITERET>
```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 6  
X-25 DECLARATIONS 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (3)

```

219     DECLARE PSECT EXEC$NONPAGED_CODE
220     .SBTTL QIO ERROR AND EXCEPTION HANDLING ROUTINES
221
222 ;
223 ; Device is not mountable. See if it is necessary to do the protection check.
224 ;
225 NOT_FILE_DEV:
226     BBC     #DEV$V_SHR,UCB$L_DEVCHAR(R5),20$           ;XFER IF NOT SHARABLE
227     BBC     R7,READ_ACCESS,10$                       ;XFER IF NOT A READ FUNCTION
228     BBS     #CCB$V_RDCHKDON,CCB$B_STS(R6),10$        ;XFER IF HERE ALREADY
229     JSB     G^EXE$CHKRDACCES                          ;DO PROTECTION CHECK
230     BLBC    RO,ERRORB                                ;XFER IF NOT SUCCESSFUL
231     BISB    #CCB$M_RDCHKDON,CCB$B_STS(R6)            ;NOTE PROT CHECK MADE
232 10$:     BBC     R7,WRITE_ACCESS,20$                 ;XFER IF NOT A WRITE FUNCTION
233     BBS     #CCB$V_WRTCHKDON,CCB$B_STS(R6),20$      ;XFER IF HERE ALREADY
234     JSB     G^EXE$CHKWRTACCES                        ;DO PROTECTION CHECK
235     BLBC    RO,ERRORB                                ;XFER IF NOT SUCCESSFUL
236     BISB    #CCB$M_WRTCHKDON,CCB$B_STS(R6)          ;NOTE PROT CHECK MADE
237 20$:     BRW     CHKDON                              ;ELSE REJOIN MAINLINE CODE
238
239 ;
240 ; An access or deaccess operation is pending for this channel. Wait for
241 ; it to complete, then retry the QIO.
242 ;
243
244 DEACCESS_PENDING:
245
246     LOCK    LOCKNAME=SCHED                          ;LOCK THE SCHED DATABASE
247     MOVZWL  #RSN$_ASTWAIT,PCB$L_EFWM(R4) ;SET AST WAIT RESOURCE NUMBER
248     MOVAQ   G^SCH$GQ_MWAIT,R2                       ;SET ADDRESS OF WAIT QUEUE
249     BSSI    #RSN$_ASTWAIT,G^SCH$GL_RESMASK,10$ ;SET WAITING FLAG
250 10$:     JMP     G^SCH$WAIT                          ;WAIT FOR AST
251
252 ;
253 ; MISCELLANEOUS ERROR HANDLING AND EXCEPTION HANDLING ROUTINES. THESE HAVE
254 ; BEEN MOVED OUT OF LINE TO MAKE THE COMMON PATH NEARLY BRANCH FREE.
255 ;
256
257 CLREF:   JSB     G^SCH$CLREF                          ;CLEAR SPECIFIED EVENT FLAG
258         BRB     VCHAN                                ;CONTINUE WITH QIO
259 IVCHAN:  MOVZWL  #SS$_IVCHAN,R0                      ;SET ERROR STATUS
260         BRB     ERRORB                              ;AND ERROR REQUEST
261 PRIVERR: MOVZWL  #SS$_NOPRIV,R0                      ;SET ERROR STATUS
262 ERRORB:  BRW     ERROR                              ;AND ERROR REQUEST
263
264 DACSPND: BRB     DEACCESS_PENDING                    ;Branch assist

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 7  
X-25 QUEUE I/O REQUEST 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (4)

```
266      .SBTTL  QUEUE I/O REQUEST
267 ;+
268 ; EXE$QIOREQ - QUEUE I/O REQUEST
269 ;
270 ; THIS SERVICE PROVIDES THE CAPABILITY TO INITIATE AN I/O OPERATION
271 ; BY QUEUEING A REQUEST TO A DEVICE'S ASSOCIATED DRIVER.  ONCE THE
272 ; OPERATION HAS BEEN INITIATED, CONTROL WILL RETURN TO THE CALLER
273 ; WHO CAN SYNCHRONIZE I/O COMPLETION IN ONE OF THREE WAYS:
274 ;
275 ;      1) SPECIFY THE ADDRESS OF AN AST ROUTINE THAT WILL BE
276 ;         EXECUTED WHEN THE I/O COMPLETES.
277 ;
278 ;      2) WAIT FOR THE SPECIFIED EVENT FLAG TO BE SET.
279 ;
280 ;      3) POLL THE SPECIFIED I/O STATUS BLOCK FOR A COMPLETION
281 ;         STATUS.
282 ;
283 ; THIS ROUTINE VERIFIES THE FUNCTION INDEPENDENT PARAMETERS, ALLOCATES
284 ; AN I/O REQUEST PACKET, COPIES THE FUNCTION INDEPENDENT PARAMETERS AND
285 ; PROCESS INFORMATION TO THE I/O PACKET, CHECKS ACCESS TO THE DEVICE,
286 ; AND CALLS THE DRIVER'S FUNCTION DECISION TABLE ROUTINE(S) THAT CORRESPOND
287 ; TO THE SPECIFIED FUNCTION.  IT IS THEN UP TO THE FDT ROUTINE TO EITHER
288 ; COMPLETE THE REQUEST IMMEDIATELY (EXE$ABORTIO OR EXE$FINISHIO) OR TO
289 ; QUEUE THE I/O REQUEST FOR FURTHER PROCESSING BY THE DRIVER'S STARTIO
290 ; ROUTINE (EXE$QIODRVPKT).
291 ;
292 ; INPUTS:
293 ;
294 ;      EFN(AP) = EVENT FLAG NUMBER.
295 ;      CHAN(AP) = I/O CHANNEL NUMBER.
296 ;      FUNC(AP) = I/O FUNCTION CODE.
297 ;      IOSB(AP) = ADDRESS OF I/O STATUS BLOCK.
298 ;      ASTADR(AP) = ADDRESS OF AST SERVICE ROUTINE.
299 ;      ASTPRM(AP) = AST SERVICE ROUTINE PARAMETER.
300 ;      P1(AP) TO P6(AP) = FUNCTION DEPENDENT PARAMETERS.
301 ;
302 ;      R4 = CURRENT PROCESS PCB ADDRESS.
303 ;
304 ; OUTPUTS:
305 ;
306 ;      R0 LOW BIT CLEAR INDICATES FAILURE TO INITIATE THE I/O REQUEST.
307 ;
308 ;      R0 = SS$_ABORT - A NETWORK LOGICAL LINK WAS BROKEN.
309 ;
310 ;      R0 = SS$_ACCVIO - THE I/O STATUS BLOCK CANNOT BE WRITTEN BY
311 ;         THE CALLER.
312 ;
313 ;      R0 = SS$_DEVOFFLINE - THE SPECIFIED DEVICE IS OFFLINE.
314 ;
315 ;      R0 = SS$_EXQUOTA - THE PROCESS HAS EXCEEDED ITS BUFFERED I/O
316 ;         QUOTA, DIRECT I/O QUOTA, OR BUFFERED I/O BYTE COUNT
317 ;         QUOTA AND HAS DISABLED RESOURCE WAIT MODE.  OR, THE
318 ;         PROCESS HAS EXCEEDED ITS AST LIMIT QUOTA.
319 ;
320 ;      R0 = SS$_ILLEFC - AN ILLEGAL EVENT FLAG NUMBER WAS SPECIFIED.
321 ;
322 ;      R0 = SS$_INSFMEM - INSUFFICIENT DYNAMIC MEMORY IS AVAILABLE
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 8  
X-25 QUEUE I/O REQUEST 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (4)

```

323 ;           TO ALLOCATE AN I/O REQUEST PACKET AND THE PROCESS HAS
324 ;           DISABLED RESOURCE WAIT MODE.
325 ;
326 ;           RO = SS$_IVCHAN - AN INVALID CHANNEL NUMBER WAS SPECIFIED.
327 ;
328 ;           RO = SS$_NOPRIV - THE SPECIFIED CHANNEL DOES NOT EXIST OR WAS
329 ;           ASSIGNED FROM A MORE PRIVILEGED ACCESS MODE. OR, THE
330 ;           PROCESS DOES NOT HAVE THE PRIVILEGE TO PERFORM THE
331 ;           SPECIFIED TYPE OF I/O FUNCTION ON THE DEVICE.
332 ;
333 ;           RO = SS$_UNASEFC - UNASSOCIATED EVENT FLAG CLUSTER SPECIFIED.
334 ;
335 ;           RO LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
336 ;
337 ;           RO = SS$_NORMAL - NORMAL COMPLETION.
338 ;--
339
340     SYSTEM_SERVICE   QIO, -
341                     <R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>,-
342                     MODE=KERNEL,-
343                     NARG=12,-
344                     TYPE=QIOW
345 ;
346 ; Clear specified event flag. For local event flags, this is done in line.
347 ;
348
349 QIO:   MOVZBL   EFN(AP),R3           ;GET EVENT FLAG NUMBER
350       CMPB    R3,#63               ;CHECK FOR LOCAL
351       BGTRU   CLREF                 ;IF NO, MUST DO FULL CLREF
352       BCCI    R3,PCB$$_EFCS(R4),VCHAN ;CLEAR SPECIFIED EVENT FLAG
353
354 ;
355 ; Validate channel number, compute CCB address and acquire UCB address.
356 ;
357
358 VCHAN: BICL3   #<^XFFFF0000!<CCB$_LENGTH-1>>,-;FETCH CHANNEL NUMBER AND
359         CHAN(AP),R0                 ;CLEAR EXTRANEIOUS BITS
360       BEQL    IVCHAN                 ;IF EQL INVALID CHANNEL
361       CMPW    R0,G^CTL$$_GW_CHINDX  ;LEGAL CHANNEL NUMBER?
362       BGTRU   IVCHAN                 ;IF GTRU NO
363       MNEGL   R0,R9                 ;CONVERT TO CHANNEL INDEX
364       MOVL    G^CTL$$_GL_CCBASE,R6  ;GET BASE ADDRESS OF CCB'S
365       MOVAB   (R6)[R9],R6           ;GET ADDRESS OF CORRESPONDING CCB
366       MOVPSL  R3                     ;READ CURRENT PSL
367       MOVZBL  CCB$_AMOD(R6),R0      ;GET CHANNEL ACCESS MODE
368       BEQL    IVCHAN                 ;BRANCH IF CHANNEL IS UNASSIGNED
369       EXTZV   #PSL$_PRVMOD,#PSL$_PRVMOD,R3,R11 ;EXTRACT PSL PREV. MODE FIELD
370       ASHL    #16,R9,R9             ;PREPARE CHANNEL INDEX FOR LATER MERGE
371       CMPB    R11,R0                 ;CALLER HAVE PRIVILEGE TO ACCESS CHANNEL?
372       BGEQ    PRIVERR               ;IF GEQ NO
373       MOVL    CCB$_UCB(R6),R5       ;GET ASSIGNED DEVICE UCB ADDRESS
374       BLBS    CCB$_WIND(R6),DACSPND ;IF LBS ACCESS/DEACCESS PENDING
375
376 ;
377 ; Isolate function code and begin decoding
378 ;
379

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 9  
X-25 QUEUE I/O REQUEST 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (4)

```

380      MOVZWL  FUNC(AP),R10          ;GET I/O FUNCTION CODE AND MODIFIERS
381      BICL3   #^C<IO$M_FCODE>,R10,R7 ;CLEAR ALL BUT I/O FUNCTION CODE
382      BBS     S^#DEV$V_SPL,UCB$L_DEVCHAR(R5),SPOOL ;IF SET, DEVICE IS SPOOLED
383
384 ;
385 ; Acquire FDT address.
386 ;
387
388 NSPOOL: BBC     #DEV$V_FOD,UCB$L_DEVCHAR(R5),NOT_FILE_DEVB      ;XFER IF NOT MOUNTAB
389 CHRDN:  MOVL   UCB$L_DDT(R5),R0          ;GET ADDRESS OF DDT
390      MOVL   DDT$L_FDT(R0),R8          ;GET ADDRESS OF FDT
391      BBC     R7,LEGAL(R8),ILLIO        ;IF CLR, ILLEGAL I/O FUNCTION
392      BBC     #UCB$V_ONLINE,UCB$W_STS(R5),OFFLINE ;IF CLR, DEVICE OFFLINE
393
394 ;
395 ; Probe and clear IOSB if it is specified.
396 ;
397
398 PRIOSB: MOVL   IOSB(AP),R1            ;GET ADDRESS OF I/O STATUS BLOCK
399      NOIOSB
400      IFNOWRT #8,(R1),ACCVIO          ;CAN I/O STATUS BLOCK BE WRITTEN?
401      CLRQ    (R1)                   ;CLEAR I/O STATUS BLOCK
402
403 ;
404 ; Charge appropriate I/O counts depending upon type. Counts will have to
405 ; be backed out if no I/O packet is available. Set IPL to block process
406 ; deletion once we are committed.
407 ;
408
409 NOIOSB: SETIPL #IPL$_ASTDEL          ;PREVENT PROCESS DELETION
410      BBS     R7,IO$TYPE(R8),10$      ;BR IF BUFFERED I/O
411      BRW     DIRECT                  ;IF CLR, DIRECT I/O FUNCTION
412 10$:  ASSUME  IRP$M_BUFIO EQ 1      ;TO ALLOW INCREMENT BELOW
413      INCW    R9                      ;SET IRP$M_BUFIO
414      ADAWI   #-1,PCB$W_BIOCNT(R4)    ;CHARGE FOR ANOTHER BUFFERED I/O
415      BLSS    20$                     ;CHECK QUOTA
416      BRW     OK                      ;OK IF NOT NEGATIVE
417 20$:  PUSHAW PCB$W_BIOCNT(R4)      ;SET ADDRESS OF QUOTA CELL
418 NOCNT: MOVL   (SP),R2               ;FETCH QUOTA ADDRESS
419      ADAWI   #1,(R2)                 ;BACKOUT CHARGE
420      SETIPL  #0                      ;LOWER IPL TO WAIT AT IPL 0
421      JSB     G^EXE$$SNGLEQUOTA      ;CHECK UNIT QUOTA OF I/O FUNCTION TYPE
422      BLBC    R0,ERROR                ;IF LBC QUOTA EXCEEDED
423      ADAWI   #-1,@(SP)+             ;CHARGE FOR I/O OF TYPE
424      BRB     OK                      ;
425
426 ILLIO: MOVZWL #SS$_ILLIOFUNC,R0     ;SET ILLEGAL I/O FUNCTION STATUS
427      BRB     ERROR                   ;
428
429 OFFLINE: CMPB  #IO$_DEACCESS,R7      ;CHECK FOR DEACCESS I/O FUNCTION
430      BEQL    PRIOSB                  ;ALLOW IT TO PROCEED
431      CMPB   #IO$_ACPCONTROL,R7      ;LIKewise FOR ACP CONTROL
432      BEQL    PRIOSB                  ;SO THAT A FILE ON AN OFFLINE DEVICE
433      ;MAY BE CLOSED
434      MOVZWL #SS$_DEVOFFLINE,R0      ;SET DEVICE OFFLINE STATUS
435      BRB     ERROR                   ;
436

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 10  
X-25 QUEUE I/O REQUEST 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (4)

```

437 ;
438 ; Device is marked spooled.  Acquire intermediate UCB address if virtual funtion.
439 ;
440 SPOOL:  Cmpl      S^#IO$ _LOGICAL,R7      ;VIRTUAL I/O FUNCTION?
441         BGEQ     NSPOOL                    ;IF GEQ NO
442         MOVL     UCB$L AMB(R5),R5          ;GET INTERMEDIATE DEVICE UCB ADDRESS
443         BRB     NSPOOL                      ;
444
445 ;
446 ; Intermediate branch to the protection checking routine.
447 ;
448 NOT_FILE DEVB:
449         BRW     NOT_FILE_DEV
450
451 ACCVIO:  MOVZWL   S^#SS$ _ACCVIO,R0        ;SET ACCESS VIOLATION STATUS
452 ERROR:  SETIPL   #0                        ;ALLOW INTERRUPTS
453         PUSHL   R0                          ;SAVE FINAL STATUS
454         MOVL     PCB$L_PID(R4),R1          ;GET PROCESS ID OF CURRENT PROCESS
455         CLRL    R2                          ;SET PRIORITY CLASS INCREMENT
456         MOVZBL  EFN(AP),R3                 ;GET SPECIFIED EVENT FLAG NUMBER
457         JSB     G^SCH$POSTEF              ;POST SPECIFIED EVENT FLAG
458         POPL    R0                          ;RESTORE FINAL STATUS
459         RET
460
461 ;
462 ; ALLOCATE REQUEST I/O PACKET - WHEN THE LOOKASIDE LIST IS EMPTY.
463 ;
464
465 ALLOC:   JSB     G^EXE$ALLOCIRP            ;ALLOCATE I/O REQUEST PACKET
466         BLBS    R0,SUCCESS                ;IF LBS SUCCESSFUL ALLOCATION
467         PUSHAW  PCB$W_BIOCNT(R4)          ;ASSUME BUFFERED I/O
468         BLBS    R9,NALLOC                 ;IF SET, BUFFERED I/O
469         PUSHAW  PCB$W_DIOCNT(R4)          ;ELSE DIRECT I/O
470 NALLOC:  ADAWI   #1,@(SP)+                ;RESTORE COUNT, SINCE NO I/O STARTED
471         BRB     ERROR                      ;
472
473 NODCNT:  PUSHAW  PCB$W_DIOCNT(R4)          ;SET FOR DIRECT I/O FUNCTION
474         BRB     NODCNT                    ;
475
476 ;
477 ; Convert section index to window address.
478 ;
479
480 SECTION:CVTWL  CCB$L_WIND(R6),R0          ;SIGN EXTEND SECTION INDEX
481         MOVL    G^CTL$G_L_PHD,R1          ;GET ADDRESS OF PROCESS HEADER
482         ADDL    PHD$L_PSTBASOFF(R1),R1    ;CALCULATE BASE ADDRESS OF SECTION TABLE
483         MOVL    SEC$L_WINDOW(R1)[R0],-4(R2) ;GET ADDRESS OF REAL WINDOW
484         BRB     NOSECT                    ;
485
486 DIRECT:  ADAWI   #-1,PCB$W_DIOCNT(R4)     ;CHARGE FOR ANOTHER DIRECT I/O
487         BLSS    NODCNT                    ;BR IF NONE ALLOWED
488 OK:
489 GTPKT:   MOVL    G^EXE$AR_SYSTEM_PRIMITIVES_DATA,R2
490         ;ADDRESS OF SYSTEM PRIMITIVES DATA AREA
491         $REMQHI IOC_GQ_IRPIQ(R2),R2      ;GET I/O PACKET FROM LOOK ASIDE LIST
492         ;NOTE: R0 is scratch in $REMQHI macro
493         BVS     ALLOC                      ;IF VS EMPTY LIST

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 11  
X-25 QUEUE I/O REQUEST 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (4)

```

494
495 ;
496 ; BUILD DEVICE INDEPENDENT PART OF I/O PACKET
497 ;
498 ; R2 - IRP Address
499 ; R4 - PCB Address
500 ; R5 - UCB Address
501 ; R6 - CCB Address
502 ; R7 - Function code (original)
503 ; R8 - FDT address
504 ; R9 - Channel index @ 16 + (IRP$M_BUFIO -- if buffered I/O)
505 ; R10 - Function code (transformed)
506 ; R11 - Access mode
507 ;
508 SUCCES: INCW   CCB$W_IOC(R6)           ;INCREMENT OUTSTANDING I/O ON CHANNEL
509           ASSUME IRP$W_SIZE EQ 8       ;FOR FOLLOWING OPTIMIZATION
510           MOVAQ  (R2)+,R3              ;COPY ADDRESS AND ADD IRP$W_SIZE TO R2
511           MOVQ   ASTADR(AP),R0         ;INSERT AST ADDRESS AND PARAMETER
512           MOVL   PCB$L_ARB(R4),IRP$L_ARB(R3) ;COPY ACCESS RIGHTS BLOCK ADDRESS
513           ASSUME IRP$B_RMOD EQ 11      ;FOR SHIFT BELOW
514           TSTL   R0                    ;CHECK FOR AST
515           BEQL   5$                     ;NONE
516           BISB   #ACB$M_QUOTA,R11      ;NOTE QUOTA CHARGE
517           ADAWI  #-1,PCB$W_ASTCNT(R4)  ;CHARGE QUOTA
518 5$:        ASHL  #24,R11,R11          ;ALIGN ACCESS MODE
519           ASSUME IRP$B_TYPE EQ 10      ;FOR BISL BELOW
520           BISL3  R11,#<<DYN$C_IRP@16>!IRP$C_LENGTH>,(R2)+ ;INSERT TYPE AND LENGTH
521           ROTL   #16,PCB$B_PRI-3(R4),R11;FETCH AND ALIGN PRIORITY
522           ASSUME IRP$L_PID EQ 12       ;
523           MOVL   PCB$L_PID(R4),(R2)+   ;INSERT PROCESS ID OF CURRENT PROCESS
524           MOVB   EFN(AP),R11          ;MERGE EVENT FLAG NUMBER
525
526           ASSUME IRP$L_AST EQ 16       ;FOR MOVQ BELOW
527           ASSUME IRP$L_ASTPRM EQ 20    ;FOR MOVQ BELOW
528           MOVQ   R0,(R2)+             ;INSERT AST ADDRESS AND PARAMETER
529           ROTL   #16,R11,R11          ;ALIGN PRIORITY AND EVENT FLAG NUMBER
530
531           ASSUME IRP$L_WIND EQ 24
532           MOVL   CCB$L_WIND(R6),(R2)+   ;GET WINDOW ADDRESS
533           BLEQ   NOSECT                ;Branch if no section index
534           BRW    SECTION              ;BR IF SECTION INDEX
535
536           ASSUME IRP$L_UCB EQ 28
537 NOSECT:    MOVL   R5,(R2)+             ;INSERT DEVICE UCB ADDRESS
538           ASSUME IRP$W_FUNC EQ 32
539           MOVW   R10,R11              ;MERGE I/O FUNCTION CODE
540           ASSUME IRP$B_EFN EQ 34
541           ASSUME IRP$B_PRI EQ 35
542           ASSUME IRP$L_IOSB EQ 36
543           MOVL   R11,(R2)+             ;INSERT PRI,EFN,FUNC
544           ADDL   #FDTACT-12,R8         ;POINT TO ACTION ROUTINE MASKS
545           MOVL   IOSB(AP),(R2)+        ;INSERT I/O STATUS BLOCK ADDRESS
546           ADDL   #P1,AP                ;POINT TO FIRST FUNCTION DEPENDENT PARAMETER
547           ASSUME IRP$W_CHAN EQ 40
548           ASSUME IRP$W_STS EQ 42
549           ROTL   #16,R9,(R2)+         ;INSERT CHANNEL INDEX AND STATUS
550           BICL3  #1,R7,R9             ;PREPARE FOR VIRTUAL CHECK BELOW

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 12  
X-25 QUEUE I/O REQUEST 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (4)

```

551      CLRQ      (R2)+          ;CLEAR PTE ADDRESS, BYTE OFFSET, AND BYTE CO
552      CLRL      (R2)          ;
553      CLRL      IRP$$_DIAGBUF (R3) ;INIT ORIGINAL PTE ADDR (LOG OR VIRT I/O
554
555      .IF DF   CA$ MEASURE IOT
556      TSTL      G^PMS$$_GL_IOPFMADB ;DATA COLLECTION ENABLED?
557      BEQL      3$             ;BR IF NO
558      JSB       G^PMS$$_START_RQ  ;INSERT START OF I/O REQUEST MESSAGE
559      BRB       4$             ;
560      .ENDC
561
562 3$:    LOCK      LOCKNAME=PERFMON, - ;LOCK THE PERFMON DATABASE
563          PRESERVE=NO          ; Don't preserve R0
564      MOVL      G^PMS$$_GL_IOPFMSEQ, IRP$$_SEQNUM(R3) ;INSERT PACKET SEQUENCE NUMBER
565      INCL      G^PMS$$_GL_IOPFMSEQ ;INCREMENT I/O TRANSACTION SEQUENCE NUMBER
566      UNLOCK    LOCKNAME=PERFMON, - ;UNLOCK THE PERFMON DATABASE
567          NEWIPL=#IPL$$_ASTDEL, - ;DROP IPL
568          PRESERVE=NO          ; Don't preserve R0
569 4$:    MOVL      UCB$$_DEVCHAR (R5), R11 ;GET DEVICE CHARACTERISTICS FOR MANY
570          ;COMPARES BELOW
571 ;
572 ; CHECK IF REQUESTING PROCESS HAS PRIVILEGE TO ACCESS DEVICE
573 ;
574 ; NOTE: LOW BIT OF FUNCTION CODE WAS CLEARED ABOVE
575 ;
576      ASSUME    IO$ READVBLK-IO$ WRITEVBLK EQ 1
577      CMLP      S^#IO$ WRITEVBLK, R9 ;VIRTUAL READ OR WRITE?
578      BNEQ      15$             ;IF NEQ NO
579      BBC       S^#DEV$$_V_FOD, R11, 5$ ;IF CLR, NOT FILE DEVICE
580
581 ;
582 ; THE FOLLOWING TEST IS NECESSITATED BY THE SYSTEM INITIALIZATION SEQUENCE
583 ;
584
585      TSTL      IRP$$_WIND (R3)    ;WINDOW ADDRESS SPECIFIED?
586      BNEQ      90$             ;IF NEQ YES
587      BBC       S^#DEV$$_V_MNT, R11, 60$ ;IF CLR, DEVICE NOT MOUNTED
588      BBC       S^#DEV$$_V_FOR, R11, 80$ ;IF CLR, MOUNTED STRUCTURED
589
590 ;
591 ; CONVERT VIRTUAL READ/WRITE FUNCTION TO ITS LOGICAL COUNTERPART
592 ;
593
594 5$:    SUBL      S^#IO$ READVBLK-IO$ READLBLK, R7 ;CONVERT TO LOGICAL FUNCTION
595      SUBW      S^#IO$ READVBLK-IO$ READLBLK, IRP$$_FUNC (R3) ;
596      BITL      #<DEV$$_M_SPL!- ;NOT SPOOLED,
597          DEV$$_M_FOD!- ;NOT FILE DEVICE,
598          DEV$$_M_SHR>, R11 ;AND NOT SHARABLE
599      BNEQ      15$             ;BR IF SATISFIED
600 ;
601 ; CHECK IF AST QUOTA IS EXCEEDED
602 ;
603
604 90$:   TSTW      PCB$$_ASTCNT (R4) ;AST QUEUE ENTRY QUOTA EXCEEDED?
605      BLSS      75$             ;IF LEQ YES
606
607 ;

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 13  
X-25 QUEUE I/O REQUEST 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (4)

```

608 ; SCAN FUNCTION DECISION TABLE CALLING EACH SELECTED ACTION ROUTINE WITH:
609 ;
610 ;     R0 = ADDRESS OF ACTION ROUTINE ENTRY POINT.
611 ;     R1 = SCRATCH.
612 ;     R2 = SCRATCH.
613 ;     R3 = ADDRESS OF I/O REQUEST PACKET.
614 ;     R4 = CURRENT PROCESS PCB ADDRESS.
615 ;     R5 = ASSIGNED DEVICE UCB ADDRESS.
616 ;     R6 = ADDRESS OF CCB.
617 ;     R7 = I/O FUNCTION CODE BIT NUMBER.
618 ;     R8 = FDT DISPATCH ADDRESS. (UPDATED TO POINT TO ACTION ROUTINE MASKS)
619 ;     R9 = SCRATCH.
620 ;     R10 = SCRATCH.
621 ;     R11 = SCRATCH.
622 ;     AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
623 ;
624 ;     NB: in the Guide to Writing a Device Driver, we document the contents
625 ;     of R0 as being the address of the FDT action routine entry point.
626 ;     This is the only reason that the dispatch code below does not read:
627 ;         JSB      @8(R8)
628 ;     Should future generations wish to modify FDT dispatching to use the
629 ;     single dispatch instruction, they must bear the responsibility for
630 ;     breaking user written drivers.
631
632 110$:  ADDL      #12,R8                ;POINT TO NEXT FUNCTION MASK
633      BBC       R7,(R8),110$          ;IF CLR, THEN ACTION NOT SELECTED
634      MOVL     8(R8),R0              ;GET ADDRESS OF ACTION ROUTINE
635      JSB      (R0)                  ;CALL ACTION ROUTINE
636      BRB     110$                  ;
637
638 ;
639 ; CONTINUE DECODING FUNCTIONS OTHER THAN VIRTUAL READ/WRITE
640 ;
641
642 15$:  CMPL     S^#IO$_LOGICAL,R7     ;VIRTUAL I/O FUNCTION?
643      BLSS    80$                    ;IF LSS YES
644
645 ;
646 ; LOGICAL OR PHYSICAL I/O FUNCTION
647 ;
648
649      IFPRIV  PHY_IO,80$              ;PROCESS HAVE PHYSICAL I/O PRIVILEGE?
650      CMPL     S^#IO$_PHYSICAL,R7    ;PHYSICAL I/O FUNCTION?
651      BLSS    20$                    ;IF LSS NO - MUST BE LOGICAL I/O
652      IFNPRIV LOG_IO,60$            ;PROCESS HAVE LOGICAL I/O PRIVILEGE?
653      MOVL     #CCB$_PHYCHKDON,R9    ;
654      MOVAB   G^EXE$_CHKPHYACCES,R10 ;SET FOR PHYSICAL I/O FUNCTION CHECK
655      BRB     30$                    ;
656
657 40$:  BBC     S^#DEV$_SHR,R11,80$    ;IF CLR, DEVICE NOT SHAREABLE
658 50$:  ; R4 - PCB ADDRESS
659      ; R5 - UCB ADDRESS
660      BBS     R9,CCB$_STS(R6),80$    ;HAS PROT CHECK BEEN MADE?
661      JSB     (R10)                  ;CHECK ACCESS TO VOLUME
662      BLBC    R0,70$                 ;EXIT ON FAILURE
663      BBSS   R9,CCB$_STS(R6),55$    ;MARK PROT CHECK DONE
664 55$:  BRB     80$                    ;ACCESS ALLOWED

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 14  
X-25 QUEUE I/O REQUEST 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (4)

```

665
666 ;
667 ; ERROR EXITS
668 ;
669 60$:   MOVZWL  #SS$ _NOPRIV, R0           ;SET NO PRIVILEGE STATUS
670 70$:   JMP      G^EXE$ABORTIO           ;
671 75$:   TSTL   IRP$L _AST(R3)           ;DOES THIS REQUEST NEED AN AST?
672       BEQL   110$                       ;NO, THEN CAN'T BE QUOTA EXCEEDED.
673       MOVZWL #SS$ _EXQUOTA, R0         ;AST QUOTA EXCEEDED
674       BRB    70$                         ;
675
676 ;
677 ; PROCESS HAS ACCESS TO DEVICE
678 ;
679 80$:   Cmpl   S^#IO$ _PHYSICAL, R7       ;LOGICAL OR VIRTUAL I/O FUNCTION?
680       BLSS  90$                           ;IF LSS YES
681       BISW  #IRP$M PHYSIO, IRP$W_STS (R3) ;SET PHYSICAL I/O FLAG
682       MOVL  <P6-P1> (AP), R9            ;GET ADDRESS OF DIAGNOSTIC BUFFER
683       BEQL  90$                           ;IF EQL THEN NOT SPECIFIED
684 ;
685 ; Process diagnostic buffer parameter
686 ;
687       IFNPRIV DIAGNOSE, 60$              ;PROCESS HAVE PRIVILEGE TO DIAGNOSE?
688       MOVL  UCB$L DDT (R5), R1           ;GET ADDRESS OF DDT
689       MOVZWL DDT$W _DIAGBUF (R1), R1     ;GET SIZE OF DIAGNOSTIC BUFFER
690       BEQL  130$                           ;IF EQL NO DIAGNOSTIC FUNCTIONS
691       JSB   G^EXE$ALLOCBUF              ;ALLOCATE DIAGNOSTIC BUFFER
692       BLBC  R0, 70$                       ;IF LBC ALLOCATION FAILURE
693       MOVL  R2, IRP$L DIAGBUF (R3)       ;SAVE ADDRESS OF DIAGNOSTIC BUFFER
694       MOVAB 12 (R2), (R2)+               ;SET POINTER TO DATA AREA
695       MOVL  R9, (R2)                      ;SAVE USER ADDRESS OF DIAGNOSTIC BUFFER
696       BISW  #IRP$M _DIAGBUF, IRP$W_STS (R3) ;SET DIAGNOSTIC BUFFER PRESENT
697 130$:   BRW    90$
698
699 ;
700 ; LOGICAL I/O FUNCTION
701 ;
702
703 20$:   IFPRIV LOG_IO, 130$              ;PROCESS HAVE LOGICAL I/O PRIVILEGE?
704       MOVL  #CCB$V _LOGCHKDON, R9       ;
705       MOVAB G^EXE$CHKLOGACCES, R10      ;SET FOR LOGICAL I/O FUNCTION CHECK
706
707 ;
708 ; PHYSICAL OR LOGICAL I/O FUNCTION - CHECK ACCESSIBILITY OF DEVICE
709 ;
710
711 30$:   BBS    S^#DEV$V _SPL, R11, 60$    ;IF SET, SPOOLED DEVICE
712       BBC    S^#DEV$V _FOD, R11, 40$    ;IF CLR, NOT FILE DEVICE
713       BBC    S^#DEV$V _MNT, R11, 60$    ;IF CLR, DEVICE NOT MOUNTED
714       BBC    S^#DEV$V _FOR, R11, 60$    ;IF CLR, MOUNTED STRUCTURED
715       BRW    50$

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 15  
X-25 BUILD I/O PACKET FOR PAGE READ/WRITE 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (5)

```
717      .SBTTL  BUILD I/O PACKET FOR PAGE READ/WRITE
718 ;+
719 ; EXE$BUILDPKTR - BUILD I/O PACKET FOR PAGE READ
720 ; EXE$BUILDPKTW - BUILD I/O PACKET FOR PAGE WRITE
721 ; EXE$BLDPKTSWPR - BUILD I/O PACKET FOR SWAP READ
722 ; EXE$BLDPKTSWPW - BUILD I/O PACKET FOR SWAP WRITE
723 ; EXE$BLDPKTGSR - BUILD I/O PACKET FOR SHARED MEMORY GLOBAL SECTION READ
724 ; EXE$BLDPKTGSW - BUILD I/O PACKET FOR SHARED MEMORY GLOBAL SECTION WRITE
725 ; EXE$BLDPKTMPW - BUILD I/O PACKET FOR MODIFIED PAGE WRITER
726 ;
727 ; THIS ROUTINE IS CALLED TO FILL OUT AND QUEUE AN I/O PACKET
728 ; FOR A SWAPPING OR PAGING READ OR WRITE.
729 ;
730 ; INPUTS:
731 ;
732 ;      R0 = VIRTUAL BLOCK NUMBER
733 ;      R1 = NUMBER OF BYTES TO TRANSFER (PAGE INCREMENTS)
734 ;      R2 = WINDOW ADDRESS FOR MAPPING VBN TO LBN
735 ;      R3 = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE ENTRY
736 ;      R4 = CURRENT PROCESS CONTROL BLOCK ADDRESS
737 ;          PCB$W_DIOCNT(R4) IS ASSUMED GREATER THAN ZERO
738 ;          AND MUST BE CHECKED BY THE CALLER.
739 ;      R5 = I/O REQUEST PACKET ADDRESS
740 ;          WITH THE FOLLOWING FIELDS ALREADY FILLED IN
741 ;
742 ;      IRP$W_SIZE(R5) AND IRP$B_TYPE(R5)
743 ;          FOR ENTRY AT EXE$BUILDPKTW, EXE$BLDPKTGSR, EXE$BLDPKTGSW, AND
744 ;          EXE$BLDPKTMPW, THESE ARE FILLED IN BY THE CALL.  FOR ALL
745 ;          OTHER ENTRY POINTS, THEY ARE FILLED IN BY THIS CODE.
746 ;      IRP$L_AST(R5) =
747 ;          FOR PAGE READ CASE - SYSTEM VIRTUAL ADDRESS OF SLAVE (PROCESS)
748 ;          PAGE TABLE ENTRY FOR THE CASE OF A GLOBAL PAGE READ.
749 ;          THIS MUST BE 0 FOR A SYSTEM OR PROCESS PAGE READ.
750 ;          FOR PAGE WRITE CASE - STANDARD QI/O AST ADDRESS
751 ;          FOR SWAPIO CASE - THIS PARAMETER IS CURRENTLY NOT USED
752 ;
753 ;      IRP$L_ASTPRM(R5) =
754 ;          FOR PAGE READ CASE - THE CONTENTS OF THE FAULTED PAGE TABLE ENTRY
755 ;          USED TO RECOVER THE ORIGINAL BACKING STORE ADDRESS WHEN A PAGE
756 ;          READ ERROR OCCURRED FOR A COPY ON REFERENCE PAGE.
757 ;          FOR PAGE WRITE CASE - STANDARD QI/O AST PARAMETER
758 ;          FOR SWAPIO CASE - ADDRESS OF KERNEL AST ROUTINE TO CALL
759 ;
760 ;      IRP$B_PRI(R5) = THE PRIORITY AT WHICH THE TRANSFER IS TO BE QUEUED
761 ;
762 ;      IRP$B_RMOD(R5) =
763 ;          FOR PAGE WRITE CASE - STANDARD QI/O MODE OF REQUESTER
764 ;          FOR ALL OTHER CASES - CONTAINS GARBAGE WHICH IS IGNORED
765 ;
766 ;      IRP$B_EFN(R5) =
767 ;          FOR PAGE WRITE CASE - STANDARD QI/O EVENT FLAG NUMBER
768 ;          FOR ALL OTHER CASES - CONTAINS GARBAGE WHICH IS IGNORED
769 ;
770 ;      IRP$L_IOSB(R5) =
771 ;          FOR PAGE WRITE CASE - STANDARD QI/O I/O STATUS BLOCK ADDRESS
772 ;          FOR ALL OTHER CASES - CONTAINS GARBAGE WHICH IS IGNORED
773 ;
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 16  
X-25 BUILD I/O PACKET FOR PAGE READ/WRITE 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (5)

```

774 ; OUTPUTS:
775 ;
776 ;     R4,R5 ALTERED
777 ;--
778
779     .ENABL  LSB
780
781 ; Note that the differentiation between READ and WRITE operations is
782 ; encoded in the setting of the IRP$M_FUNC bit.
783 ;
784 ;     IRP$M_FUNC = 1 => IO$_READBLK ; Read operation
785 ;     IRP$M_FUNC = 0 => IO$_WRITEBLK ; Write operation
786
787     UNIVERSAL_SYMBOL      EXE$BLDPKTGSR
788 ;EXE$BLDPKTGSR::          ;BUILD PACKET FOR SHMGSD READ
789     PUSHL  #<IRP$M_SWAPIO ! IRP$M_VIRTUAL ! IRP$M_FUNC>@16
790     BRB    20$             ;TYPE/SIZE ALREADY SET IN PACKET
791
792     UNIVERSAL_SYMBOL      EXE$BLDPKTGSW
793 ;EXE$BLDPKTGSW::          ;BUILD PACKET FOR SHMGSD WRITE
794     PUSHL  #<IRP$M_SWAPIO ! IRP$M_VIRTUAL>@16
795     BRB    20$             ;TYPE/SIZE ALREADY SET IN PACKET
796
797     UNIVERSAL_SYMBOL      EXE$BLDPKTSWPR
798 ;EXE$BLDPKTSWPR::          ;BUILD SWAP READ PACKET
799     PUSHL  #<IRP$M_SWAPIO ! IRP$M_VIRTUAL ! IRP$M_FUNC>@16
800     BRB    10$             ;
801
802     UNIVERSAL_SYMBOL      EXE$BLDPKTSWPW
803 ;EXE$BLDPKTSWPW::          ;BUILD SWAP WRITE PACKET
804     PUSHL  #<IRP$M_SWAPIO ! IRP$M_VIRTUAL>@16
805     BRB    10$             ;
806
807     UNIVERSAL_SYMBOL      EXE$BLDPKTMPW
808 ;EXE$BLDPKTMPW::          ;BUILD I/O PACKET FOR MODIFIED PAGE WRITER
809     PUSHL  #<IRP$M_SWAPIO ! IRP$M_VIRTUAL>@16
810     BRB    20$             ;
811
812     UNIVERSAL_SYMBOL      EXE$BUILDPKTW
813 ;EXE$BUILDPKTW::          ;BUILD I/O PACKET FOR PAGE WRITE
814     PUSHL  #<IRP$M_PAGIO ! IRP$M_VIRTUAL>@16
815     BRB    20$             ;
816
817     UNIVERSAL_SYMBOL      EXE$BUILDPKTR
818 ;EXE$BUILDPKTR::          ;BUILD I/O PACKET FOR PAGE READ
819     PUSHL  #<IRP$M_PAGIO ! IRP$M_VIRTUAL ! IRP$M_FUNC>@16
820
821 10$:   INSV  #<DYN$C_IRP@16 ! IRP$C_LENGTH>,- ;SET SIZE
822       #0,#24,IRP$W_SIZE(R5) ;AND TYPE OF PACKET
823 20$:   MOVL  R3,IRP$L_SVAPTE(R5) ;SYSTEM VIRTUAL ADR OF PAGE TABLE ENTRY
824       MOVL  R3,IRP$L_DIAGBUF(R5) ;NEED COPY OF ORIGINAL FOR SEGMENTED XFERS
825       MOVL  R5,R3 ;PACKET ADDRESS TO R3
826       EXTV  #<IRP$V_FUNC+16>,#1,(SP),R5
827
828 ; R5 = -1 for read
829 ; R5 = 0 for write
830

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 17  
X-25 BUILD I/O PACKET FOR PAGE READ/WRITE 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (5)

```
831      ASSUME <IO$_WRITEPBLK + 1> EQ IO$_READPBLK
832      ASSUME <WCB$_WRITES - 4> EQ WCB$_READS
833
834      INCL   WCB$_WRITES(R2)(R5)      ;USE CODE TO BUMP READ OR WRITE COUNT
835      SUBW3  R5,#IO$_WRITEPBLK,IRP$_FUNC(R3) ;SET REAL FUNCTION CODE
836      BICL3  #^XFFFF,(SP)+,R5        ;GET STATUS BITS AND CLEAR CHANNEL
837
838      ASSUME  IRP$_STS EQ IRP$_CHAN+2
839
840      MOVL   R5,IRP$_CHAN(R3)          ;SET CHANNEL AND STATUS
841      MOVL   WCB$_ORGUCB(R2),R5        ;GET UCB ADDRESS FROM WINDOW
842      MOVL   R5,IRP$_UCB(R3)          ;SET UCB ADDRESS
843      MOVL   PCB$_PID(R4),IRP$_PID(R3) ;PROCESS ID FROM PCB
844      MOVL   R0,IRP$_SEGVN(R3)        ;STARTING VIRTUAL BLOCK NUMBER
845      MOVL   R2,IRP$_WIND(R3)         ;WINDOW ADDRESS
846      MOVL   PCB$_ARB(R4),IRP$_ARB(R3) ;ACCESS RIGHTS BLOCK ADDRESS
847
848      CLRW   IRP$_BOFF(R3)             ;ZERO BYTE OFFSET
849      MOVL   R1,IRP$_BCNT(R3)         ;SET BYTE COUNT
850      CLRL   IRP$_ABCNT(R3)           ;ZERO ACCUMULATED BYTE COUNT
851      MOVL   R1,IRP$_OBCNT(R3)        ;SET ORIGINAL BYTE COUNT
852      CLRW   IRP$_STS2(R3)            ;CLEAR STATUS EXTENSION
853
854      .IF    DF,CA$_MEASURE_IOT
855
856      JSB    G^PMS$START_RQ           ;INSERT START OF I/O REQUEST MESSAGE
857
858      .ENDC
859
860      ADAWI  #-1,PCB$_DIOCNT(R4)       ;CHARGE DIRECT I/O TO PROCESS
861      JMP    IOC$QNXTSEG1              ;QUEUE THE FIRST SEGMENT OF THE I/O REQUEST
862      ;AND RETURN
863      .DSABL LSB
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 18  
X-25 COMPLETE I/O OPERATION 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (6)

```

865      .SBTTL  COMPLETE I/O OPERATION
866 ;+
867 ; EXE$ABORTIO - ABORT I/O OPERATION
868 ;
869 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
870 ; TO FINISH AN I/O OPERATION WITHOUT RETURNING THE FINAL I/O STATUS.
871 ;
872 ; EXE$FINISHIO - FINISH I/O OPERATION
873 ;
874 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
875 ; TO FINISH AN I/O OPERATION AND RETURN THE FINAL I/O STATUS.
876 ;
877 ; EXE$FINISHIOC - FINISH I/O OPERATION WITH SECOND I/O STATUS LONGWORD CLEARED
878 ;
879 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DESCISION TABLE ACTION ROUTINE
880 ; TO FINISH AN I/O OPERATION AND RETURN THE FINAL I/O STATUS WITH THE
881 ; SECOND I/O STATUS LONGWORD CLEARED.
882 ;
883 ; INPUTS:
884 ;
885 ;     R0 = FIRST LONGWORD OF FINAL I/O STATUS.
886 ;     R1 = SECOND LONGWORD OF FINAL I/O STATUS.
887 ;     R3 = ADDRESS OF I/O REQUEST PACKET.
888 ;     R4 = CURRENT PROCESS PCB ADDRESS.
889 ;     R5 = UCB ADDRESS OF DEVICE UNIT.
890 ;
891 ; OUTPUTS:
892 ;
893 ;     THE FINAL I/O STATUS IS STORED IN THE I/O PACKET AND THE PACKET IS
894 ;     INSERTED IN THE I/O POST PROCESSING QUEUE. A SOFTWARE INTERRUPT
895 ;     IS GENERATED TO INITIATE I/O POST PROCESSING AND THE FIRST WORD
896 ;     OF THE FINAL I/O STATUS IS RETURNED AS THE SERVICE STATUS.
897 ;
898 ; ENVIRONMENT:
899 ;
900 ;     THE FORKLOCKS ARE ACQUIRED TO INSURE THAT THE LOCK IS HELD WHILE
901 ;     UPDATING OPERATIONS COUNT ON THE UCB. AND TO MAKE SURE THAT WE CAN
902 ;     ALWAYS RELEASE THE FORKLOCK, IN CASE THE DRIVER HAD ACQUIRED IT.
903 ;-
904
905      .ENABL  LSB
906 10$:  SETIPL  UCB$B_FIPL(R5),-          ; RAISE IPL
907      ENVIRON=UNIPROCESSOR
908      BRB     20$                       ; CONTINUE
909
910      UNIVERSAL_SYMBOL      EXE$ABORTIO
911 ;EXE$ABORTIO::            ;ABORT I/O OPERATION
912      ASSUME  UCB$B_FIPL  EQ  UCB$B_FLCK
913      ASSUME  SPL$_MIN_INDEX  EQ  32
914      BBC     #5,UCB$B_FLCK(R5),10$     ;BR IF FIPL
915      FORKLOCK UCB$B_FLCK(R5)           ;LOCK FORK THREADS/USE IPL OF LOCK
916 20$:  CLRL  IRP$L_IOSB(R3)             ;CLEAR ADDRESS OF I/O STATUS BLOCK
917      BBCC   #ACB$V_QUOTA,IRP$B_RMOD(R3),60$ ;IF CLR, NO AST SPECIFIED
918      ADAWI  #1,PCB$W_ASTCNT(R4)        ;UPDATE AVAILABLE AST QUEUE ENTRIES
919      BRB     60$                       ;
920
921 30$:  SETIPL  UCB$B_FIPL(R5),-          ; RAISE IPL

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 19  
X-25 COMPLETE I/O OPERATION 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (6)

```

922          ENVIRON=UNIPROCESSOR
923          BRB      40$                ; CONTINUE
924
925          UNIVERSAL_SYMBOL          EXE$FINISHIOC
926 ;EXE$FINISHIOC::                  ;FINISH I/O OPERATION CLEAR SECOND L
927          CLRL    R1                  ;CLEAR SECOND I/O STATUS LONGWORD
928
929          UNIVERSAL_SYMBOL          EXE$FINISHIO
930 ;EXE$FINISHIO::                    ;FINISH I/O OPERATION
931          MOVQ    R0,IRP$L MEDIA(R3)  ;STORE FINAL I/O STATUS
932          ASSUME  UCB$B_FIPL EQ UCB$B_FLCK
933          ASSUME  SPL$ MIN_INDEX EQ 32
934          BBC     #5,UCB$B_FLCK(R5),30$ ;BR IF FIPL
935          FORKLOCK UCB$B_FLCK(R5),-    ;LOCK FORK THREADS/USE IPL OF LOCK
936          PRESERVE=NO                  ; DON'T PRESERVE R0
937 40$:      INCL   UCB$L OPCNT(R5)      ;INCREMENT OPERATIONS COMPLETED
938          MOVZWL  S^#SS$ NORMAL,R0    ;SET NORMAL COMPLETION STATUS
939
940 60$:      ; Make these I/O's complete synchronously by using per-CPU queue
941
942          find_cpu_data R1              ; Get per-CPU database address
943          INSQUE  (R3),@CPU$L PSBL(R1)  ; INSQUE into Q for this CPU.
944          SOFTINT #IPL$ IOPOST        ; Signal I/O post interrupt
945          FORKUNLOCK UCB$B_FLCK(R5)    ;UNLOCK FORK THREADS/USE SAME IPL
946          BRW    QIORETURN            ;
947
948          .DSABL  LSB

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 20  
X-25 QUEUE I/O PACKET TO DRIVER 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (7)

```
950      .SBTTL  QUEUE I/O PACKET TO DRIVER
951 ;+
952 ; EXE$QIODRVPKT - QUEUE I/O PACKET TO DRIVER
953 ;
954 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
955 ; TO QUEUE AN I/O PACKET TO THE APPROPRIATE DRIVER.
956 ;
957 ; INPUTS:
958 ;
959 ;      R3 = ADDRESS OF I/O REQUEST PACKET.
960 ;      R4 = CURRENT PROCESS PCB ADDRESS.
961 ;      R5 = UCB ADDRESS OF DEVICE UNIT.
962 ;
963 ; OUTPUTS:
964 ;
965 ;      THE I/O PACKET IS QUEUED BY PRIORITY IN THE APPROPRIATE DEVICE
966 ;      QUEUE AND A NORMAL COMPLETION STATUS IS RETURNED.
967 ;-
968
969      UNIVERSAL_SYMBOL      EXE$QIODRVPKT
970 ;EXE$QIODRVPKT::          ;QUEUE I/O PACKET
971      BSBW      EXE$INSIOQ      ;INSERT I/O PACKET IN DEVICE QUEUE
972      BRW      EXE$QIORETURN1      ;
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 21  
X-25 EXE\$ALTQUEPKT - Call driver ALTSTART ent 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1

```

974      .SBTTL  EXE$ALTQUEPKT - Call driver ALTSTART entry point
975
976 ;
977 ; EXE$ALTQUEPKT - activates a driver at its ALTSTART entry point
978 ;
979 ; Routine description:
980 ;
981 ;     Locates and calls a driver entry point supplied as an alternate
982 ;     START I/O entry point. Does test for unit busy before the
983 ;     call. Exits by returning to caller.
984 ;
985 ;     The routine expects to gain control at or below driver fork
986 ;     level. The routine raises to driver fork IPL before the call,
987 ;     and restores the previous IPL before returning to its caller.
988 ;
989 ; Inputs:
990 ;
991 ;     R3     - address of IRP
992 ;     R5     - address of UCB
993 ;
994 ; Outputs:
995 ;
996 ;     Control returns to the requesting process.
997 ;
998 ;     The routine destroys R0-R1.  R3 is destroyed by SMP$SWITCH_CPU.
999 ;
1000 ;--
1001
1002      .ENABL  LSB
1003
1004 5$:      ; Unmodified device driver - just use FIPLs
1005
1006      DSBINT  UCB$B_FIPL(R5),-          ; RAISE IPL
1007              ENVIRON=UNIPROCESSOR
1008      MOVL   UCB$L_DDT(R5),R0          ; Get address of unit's DDT.
1009      JSB    @DDT$L_ALTSTART(R0)      ; Call alternate start I/O routine.
1010      ENBINT
1011      RSB    ; RESTORE IPL
1012              ; RETURN TO CALLER
1013 ;
1014 ; Device is busy - Place IRP on alternate startio wait queue.
1015 ; NOTE: This wait queue should be a doubly linked list so that an INSQUE
1016 ; may be performed.  But since this would involve adding 2 longwords to the UCB
1017 ; and this is not a major release, this is not allowed.  So for now this
1018 ; will be a singly linked list.  This should be changed as soon as possible.
1019 ;
1020      ASSUME  IRP$L_IOQFL EQ 0
1021 10$:     CLRL  (R3)                   ; Clear link to indicate end of queue.
1022      MOVAB  UCB$L_ALTIOWQ(R5),R0     ; Search for the end of the
1023 11$:     TSTL  (R0)                   ;
1024      BEQL  12$                       ; alternate IO wait queue.
1025      MOVL  (R0),R0                   ; Follow the link.
1026      BRB   11$                       ; Keep going until hit the end.
1027 12$:     MOVL  R3,(R0)               ; Insert at end of queue.
1028      FORKUNLOCK LOCK=UCB$B_FLCK(R5),- ; Unlock the FORK spinlock
1029              NEWIPL=(SP)+,-         ; Restore previous IPL
1030      CONDITION=RESTORE,-           ; Conditionally release spinlock

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 22  
X-25 EXE\$ALTQUEPKT - Call driver ALTSTART ent 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1

```

1031             PRESERVE=NO             ; Don't preserve R0
1032             RSB                      ;
1033
1034             UNIVERSAL_SYMBOL          EXE$ALTQUEPKT
1035 ;EXE$ALTQUEPKT::                     ; Start I/O in driver.
1036             ASSUME  UCB$B_FIPL EQ  UCB$B_FLCK
1037             ASSUME  SPL$ _MIN_INDEX EQ 32
1038             BBC     #5,UCB$B_FLCK(R5),5$ ;BR IF FIPL
1039             FORKLCK LOCK=UCB$B_FLCK(R5),- ; Lock the FORK spinlock
1040             SAVIPL=- (SP),-           ; Save the current IPL
1041             PRESERVE=NO             ; Don't preserve R0
1042             BBS     S^#UCB$V_ALTBSY,UCB$L_STS(R5),10$ ;IF SET, THEN DEVICE IS BUSY
1043             PUSHL  R5                 ; Preserve R5
1044             ASSUME  SMP$V_ENABLED EQ  0
1045             BLBC   G^SMP$GL_FLAGS,20$ ; Br if SMP is not enabled
1046             find_cpu_data R0          ; Get address of CPU-specific database
1047             MOVL   CPU$L_PHY_CPUID(R0),R0 ; Get our CPU ID
1048             BBC    R0,UCB$L_AFFINITY(R5),60$ ; Br if can't run on this CPU
1049 20$:         MOVL   UCB$L_DDT(R5),R0    ; Get address of unit's DDT.
1050             JSB    @DDT$L_ALTSTART(R0)  ; Call alternate start I/O routine.
1051 40$:         POPL   R5                 ; Restore R5
1052             FORKUNLOCK LOCK=UCB$B_FLCK(R5),- ; Unlock the FORK spinlock
1053             NEWIPL=(SP)+,-           ; Restore previous IPL
1054             CONDITION=RESTORE,-     ; Conditionally release spinlock
1055             PRESERVE=NO             ; Don't preserve R0
1056             RSB                      ; Return to caller.
1057
1058 60$:         ;
1059             ; One last check to see if we can run on this CPU.  If UCB$L_AFFINITY
1060             ; is equal to 0, then this implies that this device is tied to the
1061             ; primary CPU.  So if this CPU is the primary, then the I/O can be
1062             ; started now.  Otherwise, a switch to the right CPU must be made.
1063             ;
1064             TSTL   UCB$L_AFFINITY(R5)   ; Check for logical primary
1065             BNEQ   65$                  ; Br if not primary
1066             CMPL   R0,G^SMP$GL_PRIMID  ; Now is this CPU the primary?
1067             BEQL   20$                  ; Br if yes, all set to start I/O
1068             ;
1069             ; We will now fork onto the correct CPU to start this I/O.
1070             ; We will have to fork on the CDRP portion of the IRP.  First,
1071             ; set the busy bit to indicate that the device is busy with an I/O.
1072             ; This is necessary since we are about to relinquish the fork-
1073             ; lock and we need a way to guarantee that no other I/O will be
1074             ; started before this I/O is started on the correct CPU.  The correct
1075             ; ordering of I/O must be maintained.
1076             ;
1077 65$:         BISL   #UCB$M_ALTBSY,UCB$L_STS(R5) ; Set busy bit.
1078             PUSHAB B^40$                ; Push return address
1079             JSB    G^SMP$SWITCH_CPU    ; Switch to right CPU
1080             ;
1081             ; This code executes as fork thread on correct CPU
1082             ;
1083             ; R3 = IRP address
1084             ; R5 = UCB address
1085             ;
1086             FORKLCK LOCK=UCB$B_FLCK(R5),- ; Lock the FORK spinlock
1087             SAVIPL=- (SP),-           ; Save the current IPL

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 23  
X-25 EXE\$ALTQUEPKT - Call driver ALTSTART ent 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1

```
1088          PRESERVE=NO          ; Don't preserve R0
1089      PUSHL   R5                ; Preserve R5.
1090 70$:      find_cpu_data   R0    ; Get address of CPU-specific database
1091      MOVL   CPU$$_PHY_CPUID(R0),R0 ; Get our CPU ID
1092      BBS    R0,UCB$$_AFFINITY(R5),75$ ; Br if can run on this CPU
1093      TSTL   UCB$$_AFFINITY(R5)    ; Check for logical primary
1094      BNEQ   65$                 ; Br if not primary
1095      CMPL   R0,G^SMP$$_GL_PRIMID   ; Now is this CPU the primary?
1096      BNEQ   65$                 ; Br if not primary, go switch to it.
1097 75$:      MOVL   UCB$$_DDT(R5),R0 ; Get address of unit's DDT.
1098      JSB    @DDT$$_ALTSTART(R0)   ; Call alternate start I/O routine.
1099 ;
1100 ; Check for any IRPs that may have been waiting while the busy bit was
1101 ; set.
1102 ;
1103      MOVL   (SP),R5              ; Retrieve R5 from the stack.
1104      MOVL   UCB$$_ALTIOWQ(R5),R3  ; Any IRPs in waiting
1105      BEQL   80$                 ; EQL, none waiting - done.
1106      ASSUME IRP$$_IOQFL EQ 0
1107      MOVL   (R3),UCB$$_ALTIOWQ(R5) ; Remove entry
1108      BRB    70$                 ; and process it.
1109 ;
1110 80$:      BICL   #UCB$$_ALTBSY,UCB$$_STS(R5) ; Clear busy bit.
1111      BRW    40$
1112      .DSABL  LSB
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 24  
X-25 QUEUE I/O PACKET TO ACP 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (9)

```

1114      .SBTTL  QUEUE I/O PACKET TO ACP
1115 ;+
1116 ; EXE$QIOACPPKT - QUEUE I/O PACKET TO ACP
1117 ;
1118 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
1119 ; TO QUEUE AN I/O PACKET TO THE APPROPRIATE ACP OR XQP.
1120 ;
1121 ; INPUTS:
1122 ;
1123 ;      R3 = ADDRESS OF I/O REQUEST PACKET.
1124 ;      R4 = CURRENT PROCESS PCB ADDRESS.
1125 ;      R5 = UCB ADDRESS OF DEVICE UNIT.
1126 ;
1127 ;      CURRENT IPL MUST BE IPL$ _ASTDEL OR IPL$ _SCHED
1128 ;
1129 ; OUTPUTS:
1130 ;
1131 ;      R4 ALTERED
1132 ;      THE I/O PACKET IS QUEUED AT THE END OF THE APPROPRIATE ACP OR XQP QUEUE
1133 ;      AND A NORMAL COMPLETION STATUS IS RETURNED.
1134 ;--
1135
1136      UNIVERSAL_SYMBOL      EXE$QIOACPPKT
1137 ;EXE$QIOACPPKT::          ;QUEUE I/O PACKET TO ACP
1138      MOVL      UCB$L_VCB (R5) ,R2      ;GET ADDRESS OF VCB
1139      MOVL      VCB$L_AQB (R2) ,R2      ;GET ADDRESS OF ACP AQB
1140      TSTL      AQB$L_ACPPID (R2)      ;GET ADDRESS OF AQB
1141      BNEQ      40$      ;NEQ - IT'S FOR XQP
1142      BRW      XQP      ;ELSE, GIVE TO XQP
1143
1144 40$:      $INSQTI (R3) ,AQB$Q_ACPIQ (R2) ,R0 ;INSERT I/O PACKET AT END OF ACP QUEUE
1145          ;NOTE: R0 IS DESTROYED BY $INSQTI MACRO
1146      BNEQ      EXE$QIORETURN1      ;IF NEQ NOT FIRST ENTRY IN QUEUE
1147      MOVL      AQB$L_ACPPID (R2) ,R1      ;GET ACP PROCESS ID
1148      LOCK      LOCKNAME=SCHED, -      ;LOCK SCHED DATABASE
1149          PRESERVE=NO      ; Don't preserve R0
1150      JSB      G^SCH$WAKE      ;WAKE UP ACP PROCESS
1151      UNLOCK   LOCKNAME=SCHED, -      ;UNLOCK SCHED DATABASE
1152          NEWIPL=#0      ;DROP IPL
1153      BLBS      R0, EXE$QIORETURN1      ;IF LBS ACP STILL PRESENT
1154      BUG_CHECK NONEXSTACP      ;NONEXISTENT ACP PROCESS
1155
1156      UNIVERSAL_SYMBOL      EXE$QIORETURN
1157
1158 ;EXE$QIORETURN::          ;QUEUE I/O REQUEST COMPLETION STATUS RETURN
1159 ;
1160 ; SMP NOTE- THE FOLLOWING NONSENSICAL CODE INSURES THAT THE FORKLCK
1161 ; IS HELD, SO WE CAN UNLOCK THE FORKLCK. THE PROBLEM IS THAT SOME
1162 ; DRIVERS RETURN HERE WITH THE FORKLCK HELD, WHILE OTHERS DO NOT
1163 ; OWN THE FORKLCK. THIS CODE SEQUENCE MAKES BOTH CASES WORK!
1164 ;
1165      ASSUME   UCB$B_FIPL EQ  UCB$B_FLCK
1166      ASSUME   SPL$ _MIN_INDEX EQ  32
1167      BBC      #5,UCB$B_FLCK (R5) ,EXE$QIORETURN1 ;BR IF FIPL
1168      FORKLCK UCB$B_FLCK (R5) , -      ;LOCK FORK THREADS/USE IPL OF LOCK
1169          PRESERVE=NO      ; Don't preserve R0
1170 EXE$QIORETURNL:

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 25  
X-25 QUEUE I/O PACKET TO ACP 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (9)

```

1171     ASSUME   UCB$B_FIPL EQ UCB$B_FLCK
1172     ASSUME   SPL$ _MIN_INDEX EQ 32
1173     BBC      #5,UCB$B_FLCK(R5),EXE$QIORETURN1 ;BR IF FIPL
1174     FORKUNLOCK LOCK=UCB$B_FLCK(R5),- ;UNLOCK FORK SPINLOCK
1175     PRESERVE=NO ; Don't preserve R0
1176
1177 EXE$QIORETURN1:: ;QUEUE I/O REQUEST COMPLETION STATUS RETURN
1178     MOVZWL   #SS$ _NORMAL,R0 ;SET NORMAL COMPLETION STATUS
1179 QIORETURN: ;RETURN SPECIFIED STATUS
1180     SETIPL   #0 ;ALLOW ALL INTERRUPTS
1181     RET      ;
1182 XQP:
1183     SETIPL   #IPL$ _ASTDEL,- ;ALLOW PAGEFAULTS
1184     ENVIRON=UNIPROCESSOR
1185     MOVAB    IRP$ _FQFL(R3), R5 ;USE CDRP PART OF IRP AS ACB
1186     PUSHAB   QIORETURN ;RETURN ADDRESS FROM EXE$QXQPPKT
1187 ;
1188 ; FALL THROUGH TO XQP QUEUEING ROUTINE IMMEDIATELY FOLLOWING.
1189 ; RSB FROM THIS ROUTINE RETURNS TO EXIT ABOVE.
1190 ;
1191
1192     .SBTTL   EXE$QXQPPKT - QUEUE I/O PACKET TO XQP
1193 ;+
1194 ; EXE$QXQPPKT - INSERT I/O PACKET IN XQP QUEUE
1195 ;
1196 ; THIS ROUTINE IS CALLED TO INSERT AN I/O PACKET IN THE XQP QUEUE
1197 ; AND START THE THREAD OF EXECUTION IF IT IS THE ONLY REQUEST.
1198 ;
1199 ; CALLING SEQUENCE:
1200 ;     BSB/JSB EXE$QXQPPKT - THIS IS EITHER CALLED FROM QIO OR
1201 ;     AS A SPECIAL KERNEL AST INVOKED BY IOPOST.
1202 ;
1203 ; INPUTS:
1204 ;
1205 ;     R4 = CURRENT PROCESS PCB ADDRESS.
1206 ;     R5 = ADDRESS OF TEMP ACB PART OF IRP.
1207 ;
1208 ; OUTPUTS:
1209 ;
1210 ;     R0 = status from SCH$QAST.
1211 ;
1212 ;     IF SUCCESS:
1213 ;     A KERNEL AST IS QUEUED TO THE DISPATCH ROUTINE OF THE XQP
1214 ;     IF NO PACKETS WERE ALREADY ON THE REQUEST QUEUE OF THE XQP.
1215 ;
1216 ;     THIS ROUTINE MUST BE CALLED AT IPL ASTDEL SO THAT THE
1217 ;     IRP CANNOT BE LOST (BECAUSE OF PROCESS DELETION) UNTIL IT
1218 ;     IS PLACED ON THE XQP REQUEST QUEUE.
1219 ;
1220 ;-
1221
1222     UNIVERSAL_SYMBOL      EXE$QXQPPKT
1223 ;EXE$QXQPPKT::
1224     MOVL     G^CTL$GL_F11BXQP, R0 ;ADDR OF XQP QUEUE HEAD
1225     MOVAB    CDRP$ _I_QFL(R5), - ;ADDRESS OF IRP
1226     ACB$ _ASTPRM(R5) ;IS AST PARAMETER.
1227     MOVB     #PSL$ _KERNEL!ACB$ _NODELETE,- ;KERNEL MODE, DON'T DELETE IRP

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 26  
X-25 EXE\$QXQPPKT - QUEUE I/O PACKET TO XQP 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (9)

```
1228          ACB$B_RMOD(R5)
1229      MOVL   PCB$P_PID(R4), ACB$P_PID(R5) ;COPY PID.
1230      MOVL   F11B$L_DISPATCH(R0), ACB$L_AST(R5) ;XQP DISPATCHER ADDRESS.
1231      MOVL   #PRI$RESAVL, R2          ;SAME AS AFTER WAITING FOR A LOCK.
1232      JMP    G^SCH$QAST              ;QUEUE THE AST AND RETURN.
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 27  
X-25 INSERT I/O PACKET IN UNIT QUEUE 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (10)

```

1234      .SBTTL  INSERT I/O PACKET IN UNIT QUEUE
1235 ;+
1236 ; EXE$INSIOQ - INSERT I/O PACKET IN UNIT QUEUE
1237 ;
1238 ; THIS ROUTINE IS CALLED TO INSERT AN I/O PACKET IN A UNIT QUEUE AND CALL
1239 ; THE APPROPRIATE I/O DRIVER IF THE UNIT IS NOT BUSY.
1240 ;
1241 ; INPUTS:
1242 ;
1243 ;     R3 = ADDRESS OF I/O REQUEST PACKET.
1244 ;     R5 = UCB ADDRESS OF DEVICE UNIT.
1245 ;
1246 ; ENVIRONMENT:
1247 ;     THE FORK SPINLOCK IS ACQUIRED, AND THEN UNCONDITIONALLY RELEASED.
1248 ;-
1249
1250      UNIVERSAL_SYMBOL      EXE$INSIOQ
1251 ;EXE$INSIOQ::              ;INSERT IN I/O QUEUE
1252      ASSUME  UCB$B_FIPL EQ  UCB$B_FLCK
1253      ASSUME  SPL$ _MIN_INDEX EQ 32
1254      BBC     #5,UCB$B_FLCK(R5),Q_FIPL ;BR IF FIPL
1255      FORKLOCK LOCK=UCB$B_FLCK(R5),- ; Lock the FORK spinlock
1256              SAVIPL=- (SP),- ; Save the current IPL
1257              PRESERVE=NO ; Don't preserve R0
1258      INCW   UCB$W_QLEN(R5) ; Bump device queue length
1259      BBSS   #UCB$V_BSY,UCB$W_STS(R5),20$ ;IF SET, THEN DEVICE IS BUSY
1260      PUSHL  R5 ; Save UCB address
1261      BSBW   IOC$INITIATE ;INITIATE I/O FUNCTION
1262      POPL   R5 ; Restore UCB address
1263      FORKUNLOCK LOCK=UCB$B_FLCK(R5),- ; Unlock the FORK spinlock
1264              NEWIPL=(SP)+,- ; Restore previous IPL
1265              PRESERVE=NO ; Don't preserve R0
1266      RSB ;
1267
1268 20$:      MOVAL  UCB$L_IOQFL(R5),R2 ;GET ADDRESS OF I/O QUEUE LISTHEAD
1269      BSBW   EXE$INSERTIRP ;INSERT I/O PACKET IN DEVICE QUEUE
1270      FORKUNLOCK LOCK=UCB$B_FLCK(R5),- ; Unlock the FORK spinlock
1271              NEWIPL=(SP)+,- ; Restore previous IPL
1272              PRESERVE=NO ; Don't preserve R0
1273      RSB ;
1274
1275 Q_FIPL:
1276      DSBINT  UCB$B_FIPL(R5),- ; RAISE IPL
1277              ENVIRON=UNIPROCESSOR
1278      INCW   UCB$W_QLEN(R5) ; Bump device queue length
1279      BBSS   #UCB$V_BSY,UCB$W_STS(R5),60$ ;IF SET, THEN DEVICE IS BUSY
1280      PUSHL  R5 ; Save UCB address
1281      BSBW   IOC$INITIATE ;INITIATE I/O FUNCTION
1282      POPL   R5 ; Restore UCB address
1283      ENBINT ; Restore previous IPL
1284      RSB ;
1285
1286 60$:      MOVAL  UCB$L_IOQFL(R5),R2 ;GET ADDRESS OF I/O QUEUE LISTHEAD
1287      BSBB   EXE$INSERTIRP ;INSERT I/O PACKET IN DEVICE QUEUE
1288      ENBINT ; Restore previous IPL
1289      RSB ;
1290

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 28  
X-25 INSERT I/O PACKET IN UNIT QUEUE 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (10)

```

1291 ;
1292 ; INSERT PACKET ON I/O AND CONDITIONALLY RELEASE SPINLOCK
1293 ;
1294         UNIVERSAL_SYMBOL           EXE$INSIOQC
1295 ;EXE$INSIOQC:;           ;INSERT IN I/O QUEUE
1296         ASSUME   UCB$B_FIPL EQ   UCB$B_FLCK
1297         ASSUME   SPL$MIN_INDEX EQ  32
1298         BBC      #5,UCB$B_FLCK(R5),Q_FIPL ;BR IF FIPL
1299         FORKLOCK LOCK=UCB$B_FLCK(R5),- ; Lock the FORK spinlock
1300         SAVIPL=-(SP),- ; Save the current IPL
1301         PRESERVE=NO ; Don't preserve R0
1302         INCW     UCB$W_QLEN(R5) ; Bump device queue length
1303         BBSS     #UCB$V_BSY,UCB$W_STS(R5),20$ ;IF SET, THEN DEVICE IS BUSY
1304         PUSHL    R5 ; Save UCB address
1305         BSBW     IOC$INITIATE ;INITIATE I/O FUNCTION
1306         POPL     R5 ; Restore UCB address
1307         FORKUNLOCK LOCK=UCB$B_FLCK(R5),- ; Unlock the FORK spinlock
1308         NEWIPL=(SP)+,- ; Restore previous IPL
1309         CONDITION=RESTORE,- ; Conditionally release lock
1310         PRESERVE=NO ; Don't preserve R0
1311         RSB ;
1312
1313 20$: MOVAL     UCB$L_IOQFL(R5),R2 ;GET ADDRESS OF I/O QUEUE LISTHEAD
1314         BSBB     EXE$INSERTIRP ;INSERT I/O PACKET IN DEVICE QUEUE
1315         FORKUNLOCK LOCK=UCB$B_FLCK(R5),- ; Unlock the FORK spinlock
1316         NEWIPL=(SP)+,- ; Restore previous IPL
1317         CONDITION=RESTORE,- ; Conditionally release lock
1318         PRESERVE=NO ; Don't preserve R0
1319         RSB ;
1320

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE 10-MAY-1989 16:24:35 VAX MACRO V5.0-8 Page 29  
X-25 INSERT I/O PACKET IN QUEUE BY PRIORITY 24-APR-1989 14:49:16 [SYS.SRC]SYSQIOREQ.MAR;1 (1

```

1322      .SBTTL  INSERT I/O PACKET IN QUEUE BY PRIORITY
1323 ;+
1324 ; EXE$INSERTIRP - INSERT I/O PACKET IN QUEUE BY PRIORITY
1325 ;
1326 ; THIS ROUTINE IS CALLED TO INSERT AN I/O PACKET IN A SPECIFIED QUEUE BY
1327 ; PRIORITY.
1328 ;
1329 ; INPUTS:
1330 ;
1331 ;     R2 = ADDRESS OF QUEUE LISTHEAD.
1332 ;     R3 = ADDRESS OF I/O PACKET.
1333 ;
1334 ;     CURRENT IPL MUST BE THE FORK LEVEL OF THE RESPECTIVE DRIVER PROCESS
1335 ;     OR HIGHER.
1336 ;
1337 ; OUTPUTS:
1338 ;
1339 ;     THE I/O PACKET IS INSERTED IN THE SPECIFIED QUEUE BY PRIORITY AND
1340 ;     THE 'Z' CONDITION CODE IS RETURNED TO THE CALLER.
1341 ;
1342 ;     'Z' = 1 = ENTRY WAS FIRST ENTRY IN THE QUEUE.
1343 ;
1344 ;     'Z' = 0 = ENTRIES WERE ALREADY IN THE QUEUE.
1345 ;
1346 ;     R2 AND R3 ARE PRESERVED ACROSS THE CALL.
1347 ;-
1348
1349      UNIVERSAL_SYMBOL      EXE$INSERTIRP
1350 ;EXE$INSERTIRP::      ;INSERT I/O PACKET IN QUEUE BY PRIOR
1351      MOVL      R2,R1      ;COPY LISTHEAD ADDRESS
1352 10$:      MOVL      IRP$L_IOQBL(R1),R1      ;GET ADDRESS OF NEXT ENTRY
1353      CML      R2,R1      ;END OF QUEUE?
1354      BEQL      20$      ;IF EQL YES
1355      CMPB      IRP$B_PRI(R3),IRP$B_PRI(R1) ;NEW ENTRY PRIORITY GREATER?
1356      BLSSU      10$      ;IF LSS YES
1357 20$:      INSQUE      IRP$L_IOQFL(R3),IRP$L_IOQFL(R1) ;INSERT PACKET IN I/O QUEUE
1358      RSB      ;
1359
1360      .END

```



# 4 SYSIMGACT.LIS

SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-32 V4.5-  
862 Page 1

10-Apr-1989 10:48:52 \_\$254\$

```
0001 0 %TITLE 'SYSIMGACT - Image Activator System Service'
0002 0 MODULE SYS$IMGACT (
0003 0 IDENT = 'X-14' ! File: SRC$:SYSIMGACT.B32
0004 0 ) =
0005 1 BEGIN
0006 1
0007 1 !*****
0008 1 !*
0009 1 !* COPYRIGHT (c) 1978, 1980, 1982, 1984, 1987 BY *
0010 1 !* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
0011 1 !* ALL RIGHTS RESERVED. *
0012 1 !*
0013 1 !* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0014 1 !* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
0015 1 !* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
0016 1 !* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0017 1 !* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
0018 1 !* TRANSFERRED. *
0019 1 !*
0020 1 !* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
0021 1 !* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
0022 1 !* CORPORATION. *
0023 1 !*
0024 1 !* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
0025 1 !* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
0026 1 !*
0027 1 !*
0028 1 !*****
0029 1
0030 1 !++
0031 1 ! Facility:
0032 1 !
0033 1 ! Executive, System Service
0034 1 !
0035 1 ! Abstract:
0036 1 !
0037 1 ! This module contains the code necessary to map a portion of process
0038 1 ! address space to a particular image file.
0039 1 !
0040 1 ! Environment:
0041 1 !
0042 1 ! The bulk of the code in this module executes in executive mode, in
0043 1 ! a layer outside RMS. One routine executes a small amount of code in
0044 1 ! kernel mode.
0045 1 !
0046 1 ! Note that the image activator is not reentrant.
0047 1 !
0048 1 ! Author:
0049 1 !
0050 1 ! Lawrence J. Kenah
0051 1 !
0052 1 ! The original version of the image activator was written by Peter Lipma
0053 1 ! During Version 2 of VMS, extensive enhancements were made by Kathy
0054 1 ! Morse.
0055 1 !
0056 1 ! Creation Date:
0057 1 !
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSSIMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 2

X-14

10-Apr-1989 10:48:52 \_\$25

```

; 0058 1 !      15 April 1983
; 0059 1 !
; 0060 1 ! Modified By:
; 0061 1 !
; 0062 1 !      X-14      WMC0014      Wayne Cardoza      10-Apr-1988
; 0063 1 !      More checks on protected images.
; 0064 1 !
; 0065 1 !      X-13      HH0334      Hai Huang      31-Aug-1988
; 0066 1 !      Treat SS$_SYSVERDIF as a fatal error.
; 0067 1 !
; 0068 1 !      X-12      HH0317      Hai Huang      25-Apr-1988
; 0069 1 !      Reset VVIEF transfer address when activating a main program.
; 0070 1 !
; 0071 1 !      X-11      WMC0011      Wayne Cardoza      29-Oct-1987
; 0072 1 !      Check flag bit before cleaning up image.
; 0073 1 !
; 0074 1 !      X-10      JWT0297      Jim Teague      13-Aug-1987
; 0075 1 !      Use ADAWI for manipulating JIB$_FILCNT.
; 0076 1 !
; 0077 1 !      X-9       JDC0370      Jon Callas      30-JUL-1987
; 0078 1 !      Implement ECOs 24 and 65 in source.
; 0079 1 !
; 0080 1 !      X-8       WMC0008      Wayne Cardoza      06-Jul-1987
; 0081 1 !      Allow more message sections.
; 0082 1 !
; 0083 1 !      X-7       AKS0002      Al Simons      16-Feb-1987
; 0084 1 !      Fix bug which kept init routines in /header_res shareable
; 0085 1 !      images from being activated.
; 0086 1 !
; 0087 1 !      X-6       AKS0001      Al Simons      26-Jan-1987
; 0088 1 !      Create a valid resultant name string area for the
; 0089 1 !      secondary image being activated. Allocate it from
; 0090 1 !      the P1 scratch space. Fix the $ASSUME checking P1
; 0091 1 !      scratch space allocation to properly account for all
; 0092 1 !      items allocated there.
; 0093 1 !
; 0094 1 !      X-5       SF04002      Stephen Fiorelli      03-Nov-1986
; 0095 1 !      Add support for new version checking mechanism.
; 0096 1 !      Version checking will be performed with a call
; 0097 1 !      to the routine EXE$CHECK_VERSION. Also remove
; 0098 1 !      the case where the version is checked for an
; 0099 1 !      image whose header is resident. This is
; 0100 1 !      unnecessary since this check will have been made
; 0101 1 !      when the image was installed.
; 0102 1 !
; 0103 1 !      X-4       SSA0001      Stan Amway      16-Oct-1986
; 0104 1 !      Support for exec mode rundown handlers.
; 0105 1 !
; 0106 1 !      X-1D5     RNH0042      Richard N. Holstein,      28-Jan-1986
; 0107 1 !      Correct typo in previous.
; 0108 1 !
; 0109 1 !      X-1D4     RNH0042      Richard N. Holstein,      28-Jan-1986
; 0110 1 !      Reference VECTORS from LIB.L32, also.
; 0111 1 !
; 0112 1 !      X-1D3     RNH0042      Richard N. Holstein,      28-Jan-1986
; 0113 1 !      Move SYS_OFFSETS into LIB.L32.
; 0114 1 !

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 3  
X-14

10-Apr-1989 10:48:52   \_\$25

|        |   |   |          |         |  |             |
|--------|---|---|----------|---------|--|-------------|
| ; 0115 | 1 | ! | X-1D2    | RNH0042 | Richard N. Holstein,   | 28-Jan-1986 |
| ; 0116 | 1 | ! |          |         | Reference SYS_OFFSETS from SHRLIB\$ instead of LIB\$.        |             |
| ; 0117 | 1 | ! |          |         |  |             |
| ; 0118 | 1 | ! | X-1C6    | SF04001 | Stephen Fiorelli   | 17-Nov-1985 |
| ; 0119 | 1 | ! |          |         | Addition of system_service macro to build a                  |             |
| ; 0120 | 1 | ! |          |         | system service descriptor block for the image                |             |
| ; 0121 | 1 | ! |          |         | activator. Also added require files SYS_OFFSETS              |             |
| ; 0122 | 1 | ! |          |         | and VECTORS to support the macro.                            |             |
| ; 0123 | 1 | ! |          |         |  |             |
| ; 0124 | 1 | ! | V04-002  | TCM0001 | Trudy C. Matthews  | 26-Sep-1985 |
| ; 0125 | 1 | ! |          |         | Use require file to define standard psects.                  |             |
| ; 0126 | 1 | ! |          |         |  |             |
| ; 0127 | 1 | ! | V04-001  | LJK0289 | Lawrence J. Kenah  | 7-Sep-1984  |
| ; 0128 | 1 | ! |          |         | Make SET_VECTORS loop into a zero-pass loop if there is no   |             |
| ; 0129 | 1 | ! |          |         | work to do.  |             |
| ; 0130 | 1 | ! |          |         |  |             |
| ; 0131 | 1 | ! | V03B-021 | MSH0056 | Michael S. Harvey  | 2-Jul-1984  |
| ; 0132 | 1 | ! |          |         | Rearrange new code for previous fix so the merged            |             |
| ; 0133 | 1 | ! |          |         | activation path for a CLI doesn't get screwed up.            |             |
| ; 0134 | 1 | ! |          |         |  |             |
| ; 0135 | 1 | ! | V03B-020 | MSH0056 | Michael S. Harvey  | 21-Jun-1984 |
| ; 0136 | 1 | ! |          |         | When a secondary image must be substituted (such as          |             |
| ; 0137 | 1 | ! |          |         | with an AME or CLI) for a primary image, make sure to        |             |
| ; 0138 | 1 | ! |          |         | reinitialize the proper context variables, thus preventing   |             |
| ; 0139 | 1 | ! |          |         | Executive mode bugchecks.                                    |             |
| ; 0140 | 1 | ! |          |         |  |             |
| ; 0141 | 1 | ! | V03B-019 | LJK0286 | Lawrence J. Kenah  | 5-Jun-1984  |
| ; 0142 | 1 | ! |          |         | Set flag bit in IMAGCTX that indicates to \$IMGFIX that      |             |
| ; 0143 | 1 | ! |          |         | shareable images contain initialization code.                |             |
| ; 0144 | 1 | ! |          |         |  |             |
| ; 0145 | 1 | ! | V03B-018 | LJK0283 | Lawrence J. Kenah  | 11-May-1984 |
| ; 0146 | 1 | ! |          |         | Set the DONE bit in all ICBs that result from a successful   |             |
| ; 0147 | 1 | ! |          |         | activation so that they do not disappear in a later          |             |
| ; 0148 | 1 | ! |          |         | activation that fails.                                       |             |
| ; 0149 | 1 | ! |          |         |  |             |
| ; 0150 | 1 | ! | V03B-017 | LJK0276 | Lawrence J. Kenah  | 8-May-1984  |
| ; 0151 | 1 | ! |          |         | Make sure that ICBs are not left dangling when an image      |             |
| ; 0152 | 1 | ! |          |         | has already been activated or along error paths.             |             |
| ; 0153 | 1 | ! |          |         |  |             |
| ; 0154 | 1 | ! | V03B-016 | LJK0274 | Lawrence J. Kenah  | 16-Apr-1984 |
| ; 0155 | 1 | ! |          |         | Fix ERROR_CLEAN_UP routine.                                  |             |
| ; 0156 | 1 | ! |          |         |  |             |
| ; 0157 | 1 | ! | V03B-015 | LJK0269 | Lawrence J. Kenah  | 31-Mar-1984 |
| ; 0158 | 1 | ! |          |         | Miscellaneous small changes.                                 |             |
| ; 0159 | 1 | ! |          |         | Add code that cleans up if an error is detected after        |             |
| ; 0160 | 1 | ! |          |         | some pages have been successfully mapped.                    |             |
| ; 0161 | 1 | ! |          |         | Set internal status bit that indicates that the P0 half      |             |
| ; 0162 | 1 | ! |          |         | of a P1 merge operation is taking place.                     |             |
| ; 0163 | 1 | ! |          |         | Change the way that privileged vector context is stored.     |             |
| ; 0164 | 1 | ! |          |         | Return correct context if image is already mapped.           |             |
| ; 0165 | 1 | ! |          |         | Add state flags passed from \$IMGACT to \$IMGFIX.            |             |
| ; 0166 | 1 | ! |          |         | Add error routine that performs a complete cleanup if        |             |
| ; 0167 | 1 | ! |          |         | an error occurs after an image has been successfully mapped. |             |
| ; 0168 | 1 | ! |          |         | Defer addition of privileged vectors until fixups are done.  |             |
| ; 0169 | 1 | ! |          |         |  |             |
| ; 0170 | 1 | ! | V03B-014 | LJK0268 | Lawrence J. Kenah  | 29-Mar-1984 |
| ; 0171 | 1 | ! |          |         | Turn on code that records SHRCNT and USECNT in KFE.          |             |

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 4  
X-14

10-Apr-1989 10:48:52 \_\$25

```

; 0172 1 !
; 0173 1 ! V03B-013 LJK0267 Lawrence J. Kenah 28-Mar-1984
; 0174 1 ! Use name stored in KFE as global section name. This allows
; 0175 1 ! correct redirection of shareable images installed /SHARED.
; 0176 1 !
; 0177 1 ! V03B-013 LJK0266 Lawrence J. Kenah 27-Mar-1984
; 0178 1 ! Add major and minor ID consistency checks on the
; 0179 1 ! image header.
; 0180 1 !
; 0181 1 ! V03B-012 MSH0022 Michael S. Harvey 25-Mar-1984
; 0182 1 ! Unconceal known file lookups.
; 0183 1 !
; 0184 1 ! V03B-011 WMC0003 Wayne Cardoza 24-Mar-1984
; 0185 1 ! Call RM$SET to set up image I/O area.
; 0186 1 !
; 0187 1 ! V03B-010 WMC0002 Wayne Cardoza 23-Jan-1984
; 0188 1 ! Misc small fixes.
; 0189 1 ! Add sequential loading of images.
; 0190 1 !
; 0191 1 ! V03B-009 LJK0244 Lawrence J. Kenah 23-Aug-1983
; 0192 1 ! Turn on image accounting. Set default stack size.
; 0193 1 !
; 0194 1 ! V03B-008 WMC0001 Wayne Cardoza 05-Aug-1983
; 0195 1 ! Remove code for passing back FAB on failure.
; 0196 1 !
; 0197 1 ! V03B-007 LJK0242 Lawrence J. Kenah 2-Aug-1983
; 0198 1 ! Add support for writable global sections.
; 0199 1 !
; 0200 1 ! V03B-006 LJK0235 Lawrence J. Kenah 26-Jul-1983
; 0201 1 ! Add concept of image base address, different from starting
; 0202 1 ! address. Continue fixing bugs and cleaning up loose ends.
; 0203 1 !
; 0204 1 ! V03B-005 LJK0232 Lawrence J. Kenah 21-Jul-1983
; 0205 1 ! The starting address of the image I/O segment should
; 0206 1 ! be page aligned.
; 0207 1 !
; 0208 1 ! V03B-004 LJK0230 Lawrence J. Kenah 18-Jul-1983
; 0209 1 ! Add system version check. Add _001 suffix to main image name
; 0210 1 !
; 0211 1 ! V03B-003 LJK0228 Lawrence J. Kenah 12-Jul-1983
; 0212 1 ! Propagate setting of EXPREG flag into the ICB.
; 0213 1 !
; 0214 1 ! V03B-002 LJK0219 Lawrence J. Kenah 29-Jun-1983
; 0215 1 ! Add support for compatibility mode and other alias images.
; 0216 1 ! Fix the many bugs that were discovered during debugging.
; 0217 1 !
; 0218 1 ! V03B-001 LJK0200 Lawrence J. Kenah 15-Apr-1983
; 0219 1 ! Being a complete rewrite of the original image activator
; 0220 1 ! system service.
; 0221 1 !--
; 0222 1

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 5

```

X-14          Declarations                                     10-Apr-1989 10:48:52    _$25

; 0224 1      %SBTTL 'Declarations'
; 0225 1
; 0226 1      SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
; 0227 1
; 0228 1      REQUIRE 'SRC$:PAGED_PSECTS.R32';                ! Define standard psects
; 0376 1      LIBRARY 'SYS$LIBRARY:LIB.L32';                  ! Define system data structures
; 0377 1
; 0378 1      REQUIRE 'LIB$:IMGMMSGDEF.R32';                 ! Get status code definitions
; 0464 1      REQUIRE 'LIB$:IMGACTCTX.R32';                   ! Define internal structures
; 0621 1
; 0622 1      ! Machine dependent features
; 0623 1
; 0624 1      BUILTIN
; 0625 1          MOVCS,
; 0626 1          MOVPSL,
; 0627 1          MTPR,
; 0628 1          PROBER,
; 0629 1          PROBEW,
; 0630 1          INSQUE,
; 0631 1          REMQUE;
; 0632 1
; 0633 1      ! Miscellaneous internal symbols
; 0634 1
; 0635 1      LITERAL
; 0636 1          TRUE = 1,
; 0637 1          FALSE = 0,
; 0638 1          BYTES_PER_PAGE = 512,
; 0639 1          RETURN_BUFFER_SIZE = 512,
; 0640 1          EXTRA_USER_STACK = 2,
; 0641 1          END_OF_PO_SPACE = %X'3FFFFFFF',
; 0642 1          EXEC_PROT = (PRT$C_UREW ^ 8) OR PSL$C_EXEC,
; 0643 1          MESSAGE_OFFSET = 1024;
; 0644 1
; 0645 1      ! Linkage declaration for procedures invoked with the $CMKRNL system service
; 0646 1
; 0647 1      LINKAGE
; 0648 1          SYS_CMKRNL = CALL : GLOBAL (PCB = 4);
; 0649 1
; 0650 1      ! Internal references
; 0651 1
; 0652 1      FORWARD ROUTINE
; 0653 1          CHECK_PARAMS,
; 0654 1          INIT_WINDOW           : SYS_CMKRNL,
; 0655 1          CHECK_MATCH_CONTROL,
; 0656 1          GET_OTHER_IMAGE,
; 0657 1          END_PROCESSING,
; 0658 1          SET_CONTROL_REGION   : SYS_CMKRNL,
; 0659 1          GET_LOCK,
; 0660 1          RELEASE_LOCK,
; 0661 1          ERROR_CLEAN_UP       : NOVALUE,
; 0662 1          SET_VECTORS;
; 0663 1
; 0664 1      ! Routines that will be referenced by the ISD mapping routines as well
; 0665 1      ! as internally
; 0666 1
; 0667 1      FORWARD ROUTINE
; 0668 1          IMG$GET_HEADER,

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SY\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 6

```

X-14          Declarations          10-Apr-1989 10:48:52    _$25

; 0669 1      IMG$OPEN_IMAGE,
; 0670 1      IMG$ALLOCATE_ICB,
; 0671 1      IMG$DEALLOCATE_ICB : NOVALUE;
; 0672 1
; 0673 1      ! Linkage declarations to JSB routines in exec
; 0674 1
; 0675 1      LINKAGE
; 0676 1      RM_RESET              = JSB : NOPRESERVE (0,1,2)
; 0677 1      NOTUSED (3,4,5,6,7,8,9,10,11),
; 0678 1      RM_SET                = JSB (REGISTER = 0, REGISTER = 1) :
; 0679 1      NOPRESERVE (0,1,2)
; 0680 1      NOTUSED (3,4,5,6,7,8,9,10,11),
; 0681 1      EXE_ALOP1PROC         = JSB (REGISTER = 1; REGISTER = 1, REGISTER = 2) :
; 0682 1      NOPRESERVE (3)
; 0683 1      NOTUSED (4,5,6,7,8,9,10,11),
; 0684 1      EXE_DEAP1             = JSB (REGISTER = 0, REGISTER = 1) :
; 0685 1      NOPRESERVE (2,3)
; 0686 1      NOTUSED (4,5,6,7,8,9,10,11),
; 0687 1      EXE_MAXACMODE         = JSB (REGISTER = 0; REGISTER = 0) :
; 0688 1      NOPRESERVE (1)
; 0689 1      NOTUSED (2,3,4,5,6,7,8,9,10,11),
; 0690 1      EXE_PROBE_DSC         = JSB (REGISTER = 1; REGISTER = 1, REGISTER = 2) :
; 0691 1      NOPRESERVE (3)
; 0692 1      NOTUSED (4,5,6,7,8,9,10,11),
; 0693 1      IMG_IS_IT_MAPPED     = JSB (REGISTER = 0; REGISTER = 1) :
; 0694 1      PRESERVE (2,3,4,5,6,7)
; 0695 1      NOTUSED (8,9,10,11),
; 0696 1      IOC_VERIFYCHAN       = JSB (REGISTER = 0; REGISTER = 1, REGISTER = 2) :
; 0697 1      NOPRESERVE (3)
; 0698 1      NOTUSED (4,5,6,7,8,9,10,11),
; 0699 1      FIL_INIWCB           = JSB (REGISTER = 1, REGISTER = 2, REGISTER = 3;
; 0700 1      REGISTER = 2) :
; 0701 1      PRESERVE (3,4,5)
; 0702 1      NOTUSED (6,7,8,9,10,11);
; 0703 1
; 0704 1      ! External references with explicit linkage mechanisms
; 0705 1
; 0706 1      EXTERNAL ROUTINE
; 0707 1      EXE$ALOP1PROC         : EXE_ALOP1PROC,
; 0708 1      EXE$DEAP1            : EXE_DEAP1,
; 0709 1      EXE$MAXACMODE        : EXE_MAXACMODE,
; 0710 1      EXE$PROBER_DSC       : EXE_PROBE_DSC,
; 0711 1      EXE$PROBEW_DSC       : EXE_PROBE_DSC,
; 0712 1      IMG$IS IT MAPPED    : IMG_IS IT MAPPED,
; 0713 1      IOC$VERIFYCHAN       : IOC_VERIFYCHAN,
; 0714 1      FIL$INIWCB           : FIL_INIWCB,
; 0715 1      RM$RESET             : RM_RESET,
; 0716 1      RM$SET               : RM_SET;
; 0717 1
; 0718 1      ! External procedure references
; 0719 1
; 0720 1      EXTERNAL ROUTINE
; 0721 1      EXE$CHECK_VERSION,
; 0722 1      FIL$OPENFILE,
; 0723 1      IMG$DECODE_IHD,
; 0724 1      IMG$DO_WORK_LIST,
; 0725 1      MMG$CRETVA           : SYS_CMKRNL;

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 7

X-14

Declarations

10-Apr-1989 10:48:52

\_\$25

```

; 0726 1
; 0727 1      ! External data cells in P1 space
; 0728 1
; 0729 1      EXTERNAL
; 0730 1          CTL$AG_CMEDATA,
; 0731 1          CTL$AL_STACK                : VECTOR [4],
; 0732 1          CTL$A_DISPVEC,
; 0733 1          CTL$GL_CTLBASVA,
; 0734 1          CTL$GL_FIXUPLNK,
; 0735 1          CTL$GL_IMGHDRBF,
; 0736 1          CTL$GL_PHD                  : REF $BBLOCK,
; 0737 1          CTL$GL_RMSBASE,
; 0738 1          CTL$GL_VOLUMES,
; 0739 1          CTL$GQ_PROCPRIV            : VECTOR [2],
; 0740 1          IAC$AW_VECSET              : VECTOR [5, WORD],
; 0741 1          IAC$GL_ICBFL              : VECTOR [2],
; 0742 1          IAC$AL_IMGACTBUF,
; 0743 1          IAC$GL_FIRST_ICB,
; 0744 1          IAC$GL_IMAGCTX            : $BBLOCK,
; 0745 1          IAC$GL_MAIN_ICB,
; 0746 1          IAC$GL_WORK_LIST          : VECTOR [2],
; 0747 1          IAC$GL_IMAGE_LIST         : VECTOR [2],
; 0748 1          IAC$GL_STACK_SIZE,
; 0749 1          PIO$GW_IIOIMPA           : $BBLOCK,
; 0750 1          CTL$GL_VVIEF_ADDR,
; 0751 1          MMG$GL_VVIEF_ADDR;
; 0752 1
; 0753 1      ! The following cells in P1 space are used exclusively for image accounting
; 0754 1
; 0755 1      EXTERNAL
; 0756 1          CTL$GL_ICPUTIM,
; 0757 1          CTL$GL_IFAULTS,
; 0758 1          CTL$GL_IFAULTIO,
; 0759 1          CTL$GL_IWSPEAK,
; 0760 1          CTL$GL_IPAGEEFL,
; 0761 1          CTL$GL_IDIOCNT,
; 0762 1          CTL$GL_IBIOCNT,
; 0763 1          CTL$GL_IVOLUMES,
; 0764 1          CTL$GQ_ISTART              : VECTOR [2];
; 0765 1
; 0766 1      ! External data cells in system space
; 0767 1
; 0768 1      EXTERNAL
; 0769 1          EXE$GL_ACMFLAGS           : $BBLOCK,
; 0770 1          EXE$GL_FLAGS              : $BBLOCK,
; 0771 1          EXE$GL_KNOWN_FILES,
; 0772 1          EXE$GL_SYSID_LOCK,
; 0773 1          EXE$GQ_KFE_LCKNAM,
; 0774 1          EXE$GQ_SYSTIME           : VECTOR [2],
; 0775 1          SGN$GW_IMGIOCNT         : WORD;
; 0776 1
; 0777 1      ! Miscellaneous constants defined elsewhere
; 0778 1
; 0779 1      EXTERNAL LITERAL
; 0780 1          EXE$C_SYSEFN              : UNSIGNED (6),
; 0781 1          EXE$V_INIT                : UNSIGNED (6),
; 0782 1          SYS$K_VERSION            : UNSIGNED (31),

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYS\$IMGACT SYS\$IMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 8

```

X-14          Declarations                                10-Apr-1989 10:48:52    _$25

; 0783 1          SYS$SO_VECTOR_BASE;
; 0784 1
; 0785 1          ! Make the bit position of the INIT flag available to BLISS
; 0786 1
; 0787 1          MACRO EXE_V_INIT = 0, EXE$V_INIT, 1, 0 %;
; 0788 1
; 0789 1          ! Some miscellaneous address definitions
; 0790 1
; 0791 1          ! The first part of the image activator scratch area is divided up into two
; 0792 1          ! large pieces, each of which is further subdivided into a FAB, a NAM block,
; 0793 1          ! a 512-byte block into which each succeeding block of the image header will
; 0794 1          ! be read, and a buffer that will receive the decoded image header. The
; 0795 1          ! decoded image header is assumed to be smaller than a page. The area that
; 0796 1          ! follows these buffers is used as OWN storage by the image activator.
; 0797 1
; 0798 1          LITERAL
; 0799 1          INPUT_BUFFER_SIZE = BYTES_PER_PAGE,
; 0800 1          IHD_BUFFER_SIZE  = BYTES_PER_PAGE;
; 0801 1
; 0802 1          BIND
; 0803 1          INPUT_BUFFER      = IAC$AL_IMGACTBUF,
; 0804 1          PRIMARY_IHD       = INPUT_BUFFER + 512,
; 0805 1          AUX_BUFFER       = PRIMARY_IHD  + 512,
; 0806 1          AUX_IHD          = AUX_BUFFER  + 512,
; 0807 1          PRIMARY_FAB      = AUX_IHD    + 512,
; 0808 1          PRIMARY_NAM      = PRIMARY_FAB + FAB$K_BLN,
; 0809 1          AUX_FAB          = PRIMARY_NAM + NAM$K_BLN,
; 0810 1          AUX_NAM          = AUX_FAB    + FAB$K_BLN,
; 0811 1          RESULT_NAME      = AUX_NAM    + NAM$K_BLN,
; 0812 1          AUX_RESULT_NAME  = RESULT_NAME + NAM$C_MAXRSS,
; 0813 1          OWN_STORAGE      = AUX_RESULT_NAME + NAM$C_MAXRSS : $BBLOCK;
; 0814 1
; 0815 1          ! There are eight pages set aside in P1 space (in module SHELL) for
; 0816 1          ! the image activator scratch area. The following assumption guarantees
; 0817 1          ! that the scratch area that is defined here fits into eight pages.
; 0818 1
; P 0819 1          $ASSUME ( OWN_STORAGE_SIZE + (2 * (INPUT_BUFFER_SIZE +
; P 0820 1          IHD_BUFFER_SIZE +
; P 0821 1          FAB$K_BLN +
; P 0822 1          NAM$K_BLN +
; P 0823 1          NAM$C_MAXRSS) ),
; P 0824 1          LEQU,
; 0825 1          8 * BYTES_PER_PAGE );

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 9

```

X-14      EXE$IMGACT - Image Activator System Service Rou 10-Apr-1989 10:48:52    _$25
; 0827 1    %SBTTL 'EXE$IMGACT - Image Activator System Service Routine'
; 0828 1
; 0829 1    GLOBAL ROUTINE EXE$IMGACT
; 0830 1      (IMAGE_NAME_ADDR , DEFAULT_NAME_ADDR , BUFFER , CONTROL_FLAGS ,
; 0831 1      INADR , RETADR , IDENT , MODE ) =
; 0832 1
; 0833 1    !++
; 0834 1    ! FUNCTIONAL DESCRIPTION:
; 0835 1    !
; 0836 1    !       This routine receives control from the system service dispatcher
; 0837 1    !       to perform the actual work of activating an image.
; 0838 1    !
; 0839 1    ! CALLING SEQUENCE:
; 0840 1    !
; 0841 1    !       CALLx G^SYSS$IMGACT
; 0842 1    !
; 0843 1    ! FORMAL PARAMETERS:
; 0844 1    !
; 0845 1    !       TBS
; 0846 1    !
; 0847 1    ! STATUS CODES:
; 0848 1    !
; 0849 1    !       TBS
; 0850 1    !--
; 0851 1
; 0852 2    BEGIN
; 0853 2
; P 0854 2    SYSTEM_SERVICE (NAME=IMGACT,      ! This macro builds a system
; P 0855 2      NARG=8,                          ! service descriptor block
; 0856 2      MODE=EXEC)                          ! for the image activator
; 0857 2
; 0858 2    BUILTIN
; 0859 2      AP,
; 0860 2      CALLG;
; 0861 2
; 0862 2    BIND FLAGS = OWN_STORAGE [INPUT_FLAGS] : $BBLOCK;
; 0863 2
; 0864 2    LOCAL
; 0865 2      ICB_ADR      : REF $BBLOCK,
; 0866 2      IHD_CTX      : $BBLOCK [CTX_K_LENGTH],
; 0867 2      STATUS;
; 0868 2
; 0869 2    ! The portion of the impure area that is used as OWN storage is filled with
; 0870 2    ! zeros. This initializes system service parameters, copies of input
; 0871 2    ! parameters, and the like.
; 0872 2
; 0873 2    CH$FILL (0, OWN_STORAGE_SIZE, OWN_STORAGE);
; 0874 2    OWN_STORAGE [FINAL_STATUS] = SS$ NORMAL;      ! Assume successful
; 0875 2    OWN_STORAGE [USER_STACK_SIZE] = EXTRA_USER_STACK;  ! Assume a minimal u
; 0876 2
; 0877 2    ! No access check is required for the activation flags, which are present in
; 0878 2    ! the argument list itself. A safe copy must be made, however, to insure tha
; 0879 2    ! the flags are not destroyed by a mapping request issued by the image
; 0880 2    ! activator. To avoid this, the flags are stored away in a safe place.
; 0881 2
; 0882 2    OWN_STORAGE [INPUT_FLAGS] = .CONTROL_FLAGS;
; 0883 2

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 10

```

X-14      EXE$IMGACT - Image Activator System Service Rou 10-Apr-1989 10:48:52    _$25
; 0884 2      IF .FLAGS [IAC$V_SETVECTOR]
; 0885 2      THEN RETURN SET_VECTORS();
; 0886 2
; 0887 2      ! Save the callers mode for use in various checks
; 0888 2
; 0889 3      BEGIN
; 0890 3      LOCAL psl : BLOCK[4,BYTE];
; 0891 3
; 0892 3      MOVPSL (psl);
; 0893 3      OWN_STORAGE [CALL_MODE] = .psl[psl$v_prvmod];
; 0894 2      END;
; 0895 2
; 0896 2      ! Set up the starting points for the various privileged vectors
; 0897 2
; 0898 2      INCRU I FROM 0 TO 4 DO
; 0899 3          BEGIN
; 0900 3
; 0901 3          BIND
; 0902 3              DISPVEC = CTL$A_DISPVEC + (.I * 256) : LONG;
; 0903 3
; 0904 3              DISPVEC = .IAC$AW_VECSET [.I];
; 0905 3
; 0906 2          END;
; 0907 2
; 0908 2      ! The input parameter list is checked for accessibility and the parameters a
; 0909 2      ! stored in the impure area for later use.
; 0910 2
; 0911 2      STATUS = CALLG (.AP, CHECK_PARAMS);
; 0912 2      IF NOT .STATUS THEN RETURN .STATUS;
; 0913 2
; 0914 2      ! Several other miscellaneous areas need to be initialized if this is
; 0915 2      ! not a activation that merges an additional image into existing address
; 0916 2      ! space. Note that the fixup vector listhead is unconditionally cleared.
; 0917 2
; 0918 2      CTL$GL_FIXUPLNK = 0;                ! Set the fixup vector list to empty
; 0919 2      IF NOT .FLAGS [IAC$V_MERGE]
; 0920 2      THEN
; 0921 3          BEGIN
; 0922 3              RM$RESET ();                ! Clear the image I/O segment
; 0923 3              CTL$GL_VVIEF_ADDR = .MMG$GL_VVIEF_ADDR; ! Reset VVIEF transfer address
; 0924 3              IAC$GL_IMAGCTX = 0;        ! Start with a clean context slate
; 0925 3              OWN_STORAGE [MAIN_PROGRAM] = TRUE; ! Indicate that this is the activati
; 0926 3              END                        ! of a main program
; 0927 2      ELSE
; 0928 3          BEGIN
; 0929 3
; 0930 3          BIND CONTEXT = IAC$GL_IMAGCTX : VECTOR [2,WORD];
; 0931 3
; 0932 3          CONTEXT [1] = 0;                ! Only clear flags passed to $IMGFIX
; 0933 2          END;
; 0934 2
; 0935 2      ! Before we open the image file, we need to check whether the image is alrea
; 0936 2      ! mapped. If it is, we simply return successfully, passing back as much data
; 0937 2      ! as is available about the image.
; 0938 2
; 0939 2      STATUS = IMG$ALLOCATE_ICB (ICB_ADR); ! Allocate an ICB for the primary im
; 0940 2

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYS\$IMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 11

```

X-14          EXE$IMGACT - Image Activator System Service Rou 10-Apr-1989 10:48:52    _$25

; 0941 2      IF NOT .STATUS THEN RETURN .STATUS;
; 0942 2
; 0943 2      ! The following check is only made for a merge activation
; 0944 2
; 0945 2      IF .FLAGS [IAC$V_MERGE]
; 0946 2      THEN
; 0947 3          BEGIN
; 0948 3
; 0949 3          BIND
; 0950 3              INPUT_NAME = OWN_STORAGE [IMAGE_NAME_DESC] : $BBLOCK,
; 0951 3              ICB_NAME = ICB_ADR [ICB$I_IMAGE_NAME] : VECTOR [,BYTE],
; 0952 3              RETADR = .OWN_STORAGE [RETURN_ARRAY_ADDRESS] : VECTOR [2],
; 0953 3              BUFFER = .OWN_STORAGE [BUFFER_ADDRESS] : VECTOR [128];
; 0954 3
; 0955 3          LOCAL
; 0956 3              MAPPED_ICB : REF $BBLOCK;
; 0957 3
; 0958 3          ICB_NAME [0] = .INPUT_NAME [DSC$W_LENGTH];
; 0959 3          MOVCS (
; 0960 3              INPUT_NAME [DSC$W_LENGTH],
; 0961 3              .INPUT_NAME [DSC$A_POINTER],
; 0962 3              %REF(0),
; 0963 3              %REF(ICB$I_IMAGE_NAME-1),
; 0964 3              ICB_NAME [1]);
; 0965 3          STATUS = IMG$I$IT_MAPPED (ICB_NAME; MAPPED_ICB);
; 0966 3
; 0967 3          ! If the shareable image has already been mapped, we return successfully
; 0968 3          ! after passing back to the caller whatever information is available.
; 0969 3
; 0970 3      IF .STATUS EQL SS$_NORMAL
; 0971 3      THEN
; 0972 4          BEGIN
; 0973 4
; 0974 4          IF RETADR NEQU 0
; 0975 4          THEN
; 0976 5              BEGIN
; 0977 5                  RETADR [0] = .MAPPED_ICB [ICB$L_STARTING_ADDRESS];
; 0978 5                  RETADR [1] = .MAPPED_ICB [ICB$L_END_ADDRESS];
; 0979 4              END;
; 0980 4
; 0981 4          IF BUFFER NEQU 0
; 0982 4          THEN
; 0983 5              BEGIN
; 0984 5
; 0985 5                  BIND IFD = BUFFER [3] : $BBLOCK;
; 0986 5
; 0987 5                  BUFFER [0] = 0;
; 0988 5                  BUFFER [1] = IFD;
; 0989 5                  BUFFER [2] = 0;
; 0990 5                  IFD [IFD$W_CHAN] = .MAPPED_ICB [ICB$W_CHAN];
; 0991 5                  IFD [IFD$W_FLAGS] = .IAC$GL_IMAGCTX;
; 0992 5
; 0993 4              END;
; 0994 4
; 0995 4          IMG$DEALLOCATE_ICB (.ICB_ADR);
; 0996 4
; 0997 4          RETURN SS$_NORMAL;

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 12

X-14 EXE\$IMGACT - Image Activator System Service Rou 10-Apr-1989 10:48:52 \_\$25

```

; 0998 4
; 0999 3          END;
; 1000 3
; 1001 2          END;
; 1002 2
; 1003 2  IAC$GL_FIRST_ICB = .ICB_ADR;          ! Remember address of this ICB
; 1004 2  IF .OWN_STORAGE [MAIN_PROGRAM]
; 1005 2  THEN IAC$GL_MAIN_ICB = .ICB_ADR;      ! Remember it here, too, if main pro
; 1006 2  ICB_ADR [ICB$L_CONTEXT] = IHD_CTX;    ! Store address before opening image
; 1007 2
; 1008 2  STATUS = GET_LOCK();                  ! Lock the KFE data base for read ac
; 1009 2  IF NOT .STATUS THEN RETURN .STATUS;
; 1010 2
; 1011 2  STATUS = IMG$OPEN_IMAGE (             ! Open the image file
; 1012 2  OWN_STORAGE [IMAGE_NAME_DESC],
; 1013 2  OWN_STORAGE [DFLT_NAME_DESC],
; 1014 2  PRIMARY_FAB,
; 1015 2  PRIMARY_NAM,
; 1016 2  RESULT_NAME,
; 1017 2  .ICB_ADR);
; 1018 2  IF NOT .STATUS
; 1019 2  THEN
; 1020 3  BEGIN
; 1021 3  IMG$DEALLOCATE_ICB (.ICB_ADR);
; 1022 3  RELEASE_LOCK ();
; 1023 3  RETURN .STATUS
; 1024 2  END;
; 1025 2
; 1026 2  ! The IHD_CTX context block stores the image header data that does not need
; 1027 2  ! to exist once the image activator is done. Because this ICB represents the
; 1028 2  ! image whose name was passed directly to the image activator, the primary
; 1029 2  ! input buffer and IHD buffer are used.
; 1030 2
; 1031 2  IHD_CTX [CTX_L_BUFFER] = INPUT_BUFFER;
; 1032 2  IHD_CTX [CTX_L_IHDBUF] = PRIMARY_IHD;
; 1033 2
; 1034 2  STATUS = IMG$GET_HEADER (.ICB_ADR);    ! Decode and store away the IHD cont
; 1035 2  IF NOT .STATUS
; 1036 2  THEN
; 1037 3  BEGIN
; 1038 3  $DASSGN (CHAN = .ICB_ADR [ICB$W_CHAN]);
; 1039 3  ERROR_CLEAN_UP ();
; 1040 3  IMG$DEALLOCATE_ICB (.ICB_ADR);
; 1041 3  RELEASE_LOCK ();
; 1042 3  RETURN .STATUS
; 1043 2  END;
; 1044 2
; 1045 2  ! There are several alias images that cause a secondary image to be activate
; 1046 2  ! leaving the primary image opened and its name stored in P1 space for later
; 1047 2  ! possible use by the secondary image, the one actually activated.
; 1048 2
; 1049 2  IF .IHD_CTX [CTX_W_ALIAS] NEQ IHD$C_NATIVE
; 1050 2  THEN
; 1051 3  BEGIN
; 1052 3  STATUS = GET_OTHER_IMAGE (.ICB_ADR);
; 1053 3  IF NOT .STATUS
; 1054 3  THEN

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 13

X-14 EXE\$IMGACT - Image Activator System Service Rou 10-Apr-1989 10:48:52 \_\$25

```

; 1055 4          BEGIN
; 1056 4          IMG$DEALLOCATE_ICB (.ICB_ADR);
; 1057 4          RELEASE_LOCK ();
; 1058 4          RETURN .STATUS
; 1059 3          END;
; 1060 2          END;
; 1061 2
; 1062 2          ! Several other fields in the ICB must be loaded with information obtained
; 1063 2          ! from the input parameter list.
; 1064 2
; 1065 2          ! Note that EXPREG is only valid when mapping into P0 space. Note further th
; 1066 2          ! the input map range is specified explicitly, rather than using the EXPREG
; 1067 2          ! flag in the ICB, which is only meaningful in ICBs for shareable images
; 1068 2          ! implicitly referenced during the activation of the primary image.
; 1069 2
; 1070 2          IF .FLAGS [IAC$V_EXPREG]
; 1071 2          THEN
; 1072 3              BEGIN
; 1073 3
; 1074 3              BIND
; 1075 3                  PHD = .CTL$GL_PHD          : $BBLOCK;
; 1076 3
; 1077 3                  ICB_ADR [ICB$SL_STARTING_ADDRESS] = .PHD [PHD$SL_FREPOVA];
; 1078 3                  ICB_ADR [ICB$SL_END_ADDRESS] = END_OF_PO_SPACE;
; 1079 3                  END
; 1080 2          ELSE
; 1081 3              BEGIN
; 1082 3                  ICB_ADR [ICB$SL_STARTING_ADDRESS] = .OWN_STORAGE [INPUT_START_ADDRESS];
; 1083 3                  ICB_ADR [ICB$SL_END_ADDRESS] = .OWN_STORAGE [INPUT_END_ADDRESS];
; 1084 2              END;
; 1085 2
; 1086 2          ICB_ADR [ICB$B_ACCESS_MODE] = .OWN_STORAGE [ACCESS_MODE];
; 1087 2          ICB_ADR [ICB$B_ACT_CODE] =
; 1088 3              (IF .FLAGS [IAC$V_MERGE]
; 1089 3                  THEN ICB$K_MERGED_IMAGE
; 1090 2                  ELSE ICB$K_MAIN_PROGRAM);
; 1091 2          ICB_ADR [ICB$SL_MATCH_CONTROL] = .OWN_STORAGE [MATCH_CONTROL];
; 1092 2          ICB_ADR [ICB$SL_VERSION] = .OWN_STORAGE [VERSION];
; 1093 2
; 1094 2          ! The image control block is inserted into the work list where it can be
; 1095 2          ! retrieved by the routine that converts ISDs into mapping requests.
; 1096 2
; 1097 2          INSQUE (.ICB_ADR , .IAC$GL_WORK_LIST [1]);          ! Insert at tail of work lis
; 1098 2          STATUS = IMG$DO_WORK_LIST();
; 1099 2          IF NOT .STATUS THEN
; 1100 3              BEGIN
; 1101 3                  ERROR_CLEAN_UP ();
; 1102 3                  RELEASE_LOCK ();
; 1103 3                  RETURN .STATUS
; 1104 2              END;
; 1105 2
; 1106 2          STATUS = END_PROCESSING (.ICB_ADR);          ! Set final state for image
; 1107 2          IF NOT .STATUS THEN
; 1108 3              BEGIN
; 1109 3                  ERROR_CLEAN_UP ();
; 1110 3                  RELEASE_LOCK ();
; 1111 3                  RETURN .STATUS

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 14

```
X-14          EXE$IMGACT - Image Activator System Service Rou 10-Apr-1989 10:48:52    _$25
; 1112 2          END;
; 1113 2
; 1114 2          RELEASE_LOCK();                ! Allow exclusive access again
; 1115 2          RETURN .OWN_STORAGE [FINAL_STATUS]; ! ... and return
; 1116 2
; 1117 1          END;                            ! End of routine EXE$IMGACT main rou
```

```
.TITLE SYSS$IMGACT SYSIMGACT - Image Activator System S
      ervice
```

```
.IDENT \X-14\
```

```
.PSECT EXEC$INIT_SSTBL_001, PIC,2
```

```
00000000G 00000 BASE_VECTOR_IMGACT:
      .LONG SYS$SO_VECTOR_BASE+400 ;
00000000' 00004 SELF_RELATIVE_IMGACT:
      .ADDRESS EXE$IMGACT ;
08 00008 MINIMUM_ARG_IMGACT:
      .BYTE 8 ;
08 00009 MAXIMUM_ARG_IMGACT:
      .BYTE 8 ;
01 0000A MODE_IMGACT:
      .BYTE 1 ;
81 0000B IMASK_IMGACT:
      .BYTE -127 ;
00 0000C TYPE_IMGACT:
      .BYTE 0 ;
00 0000D EXIT_IMGACT:
      .BYTE 0 ;
0000 0000E RESERVED_IMGACT:
      .WORD 0 ;
```

```
.EXTRN EXE$ALOP1PROC, EXE$DEAPI
.EXTRN EXE$MAXACMODE, EXE$PROBER_DSC
.EXTRN EXE$PROBEW_DSC, IMG$IS_IT_MAPPED
.EXTRN IOC$VERIFYCHAN, FIL$INIWCB
.EXTRN RM$RESET, RM$SET
.EXTRN EXE$CHECK_VERSION
.EXTRN FIL$OPENFILE, IMG$DECODE_IHD
.EXTRN IMG$DO_WORK_LIST
.EXTRN MMG$CRETVA, CTL$AG_CMEDATA
.EXTRN CTL$AL_STACK, CTL$A_DISPVEC
.EXTRN CTL$GL_CTLBASVA
.EXTRN CTL$GL_FIXUPLNK
.EXTRN CTL$GL_IMGHDRBF
.EXTRN CTL$GL_PHD, CTL$GL_RMSBASE
.EXTRN CTL$GL_VOLUMES, CTL$GQ_PROCPRIV
.EXTRN IAC$AW_VECSET, IAC$GL_ICBFL
.EXTRN IAC$AL_IMGACTBUF
.EXTRN IAC$GL_FIRST_ICB
.EXTRN IAC$GL_IMAGCTX, IAC$GL_MAIN_ICB
.EXTRN IAC$GL_WORK_LIST
.EXTRN IAC$GL_IMAGE_LIST
.EXTRN IAC$GL_STACK_SIZE
.EXTRN PIO$GW_IOIMPA, CTL$GL_VVIEF_ADDR
.EXTRN MMG$GL_VVIEF_ADDR
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 15  
X-14

EXE\$IMGACT - Image Activator System Service Rou 10-Apr-1989 10:48:52 \_\$25

```

.EXTRN CTL$GL_ICPUTIM, CTL$GL_IFAULTS
.EXTRN CTL$GL_IFAULTIO
.EXTRN CTL$GL_IWSPEAK, CTL$GL_IPAGEFL
.EXTRN CTL$GL_IDIOCNT, CTL$GL_IBIOCNT
.EXTRN CTL$GL_IVOLUMES
.EXTRN CTL$GO_ISTART, EXE$GL_ACMFLAGS
.EXTRN EXE$GL_FLAGS, EXE$GL_KNOWN_FILES
.EXTRN EXE$GL_SYSID_LOCK
.EXTRN EXE$GO_KFE_LCKNAM
.EXTRN EXE$GO_SYSTIME, SGN$GW_IMGIOCNT
.EXTRN EXE$C_SYSEFN, EXE$V_INIT
.EXTRN SYS$K_VERSION, SYS$SO_VECTOR_BASE
.EXTRN SYS$DASSGN

.PSECT EXEC$PAGED_CODE, NOWRT, PIC, 2

; 0829
OFFC 00000
.ENTRY EXE$IMGACT, Save R2, R3, R4, R5, R6, R7, R8, R9, -
R10, R11
;
00000000G 00 9E 00002 MOVAB IAC$GL_IMAGCTX, R11 ;
00000000G 00 9E 00009 MOVAB OWN_STORAGE+68, R10 ;
BB AE 9E 00010 MOVAB -72(SP), SP ;
00 2C 00014 MOVCS #0, (SP), #0, #104, OWN_STORAGE ; 0873
BC AA 0001B ;
01 D0 0001D MOVL #1, OWN_STORAGE+16 ; 0874
02 D0 00021 MOVL #2, OWN_STORAGE+28 ; 0875
10 AC D0 00025 MOVL CONTROL_FLAGS, OWN_STORAGE+68 ; 0882
05 E1 00029 BBC #5, FLAGS+2, 1$ ; 0884
00 FB 0002E CALLS #0, SET_VECTORS ; 0885
04 00033 RET ;
50 DC 00034 1$: MOVPSL PSL ; 0892
16 EF 00036 EXTZV #22, #2, PSL, R1 ; 0893
51 90 0003B MOVB R1, OWN_STORAGE+36 ;
50 D4 0003F CLRL I ; 0898
08 78 00041 2$: ASHL #8, I, R1 ; 0902
00000000G0041 9F 00045 PUSHAB CTL$A_DISPVEC[R1] ; 0904
00000000G0040 3C 0004C MOVZWL IAC$AW_VECSET[I], @(SP)+ ;
50 D6 00054 INCL I ;
50 D1 00056 CMLP I, #4 ;
E6 1B 00059 BLEQU 2$ ;
6C FA 0005B CALLG (AP), CHECK_PARAMS ; 0911
50 D0 00060 MOVL R0, STATUS ;
59 E9 00063 BLBC STATUS, 5$ ; 0912
00000000G 00 D4 00066 CLRL CTL$GL_FIXUPLNK ; 0918
04 E0 0006C BBS #4, FLAGS, 3$ ; 0919
00000000G 00 16 00070 JSB RMSRESET ; 0922
00000000G 00 D0 00076 MOVL MMG$GL_VVIEF_ADDR, CTL$GL_VVIEF_ADDR ; 0923
6B D4 00081 CLRL IAC$GL_IMAGCTX ; 0924
01 88 00083 BISB2 #1, OWN_STORAGE ; 0925
03 11 00087 BRB 4$ ;
02 AB B4 00089 3$: CLRW CONTEXT+2 ; 0932
5E DD 0008C 4$: PUSHL SP ; 0939
01 FB 0008E CALLS #1, IMG$ALLOCATE_ICB ;
50 D0 00093 MOVL R0, STATUS ;
59 E8 00096 5$: BLBS STATUS, 6$ ; 0941
0157 31 00099 BRW 21$ ;
04 E1 0009C 6$: BBC #4, FLAGS, 9$ ; 0945
14 C1 000A0 ADDL3 #20, ICB_ADR, R8 ; 0951

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 16

X-14 EXE\$IMGACT - Image Activator System Service Rou 10-Apr-1989 10:48:52 \$25

|           |    |    |      |       |             |                 |                                  |  |        |
|-----------|----|----|------|-------|-------------|-----------------|----------------------------------|--|--------|
|           | OC | AA | D0   | 000A4 |             | MOVL            | OWN_STORAGE+80, R7               |  | ; 0952 |
|           | FC | AA | D0   | 000A8 |             | MOVL            | OWN_STORAGE+64, R6               |  | ; 0953 |
|           | EC | AA | 90   | 000AC |             | MOVB            | INPUT_NAME, (R8)                 |  | ; 0958 |
|           | FO | AA | D0   | 000B0 |             | MOVL            | INPUT_NAME+4, R0                 |  | ; 0961 |
|           | EC | AA | 2C   | 000B4 |             | MOVCS           | INPUT_NAME, (R0), #0, #39, 1(R8) |  | ; 0964 |
|           | 01 | A8 |      | 000BA |             |                 |                                  |  | ;      |
|           |    |    | 58   | D0    | 000BC       | MOVL            | R8, R0                           |  | ; 0965 |
| 00000000G |    |    | 00   | 16    | 000BF       | JSB             | IMG\$IS_IT MAPPED                |  | ;      |
|           |    |    | 50   | D0    | 000C5       | MOVL            | R0, STATUS                       |  | ;      |
|           |    |    | 59   | D1    | 000C8       | CMP             | STATUS, #1                       |  | ; 0970 |
|           |    |    | 2D   | 12    | 000CB       | BNEQ            | 9\$                              |  | ;      |
|           |    |    | 57   | D5    | 000CD       | TSTL            | R7                               |  | ; 0974 |
|           |    |    | 04   | 13    | 000CF       | BEQL            | 7\$                              |  | ;      |
| 48        |    | A1 | 7D   | 000D1 |             | MOVQ            | 72(MAPPED_ICB), (R7)             |  | ; 0978 |
|           |    |    | 56   | D5    | 000D5       | 7\$: TSTL       | R6                               |  | ; 0981 |
|           |    |    | 16   | 13    | 000D7       | BEQL            | 8\$                              |  | ;      |
|           | OC | A6 | 9E   | 000D9 |             | MOVAB           | 12(R6), R0                       |  | ; 0985 |
|           |    |    | 66   | D4    | 000DD       | CLRL            | (R6)                             |  | ; 0987 |
|           |    |    | 50   | D0    | 000DF       | MOVL            | R0, 4(R6)                        |  | ; 0988 |
|           | 08 | A6 | D4   | 000E3 |             | CLRL            | 8(R6)                            |  | ; 0989 |
|           | 0E | A1 | B0   | 000E6 |             | MOVW            | 14(MAPPED_ICB), 8(R0)            |  | ; 0990 |
|           |    |    | 6B   | B0    | 000EB       | MOVW            | IAC\$GL_IMGCTX, 16(R0)           |  | ; 0991 |
|           |    |    | 6E   | DD    | 000EF       | 8\$: PUSHL      | ICB_ADR                          |  | ; 0995 |
|           |    |    | 01   | FB    | 000F1       | CALLS           | #1, IMG\$DEALLOCATE_ICB          |  | ;      |
|           |    |    | 01   | D0    | 000F6       | MOVL            | #1, R0                           |  | ; 0997 |
|           |    |    |      | 04    | 000F9       | RET             |                                  |  | ;      |
|           |    |    | 6E   | D0    | 000FA       | 9\$: MOVL       | ICB_ADR, R2                      |  | ; 1003 |
|           |    |    | 52   | D0    | 000FD       | MOVL            | R2, IAC\$GL_FIRST_ICB            |  | ;      |
|           | BC | AA | E9   | 00104 |             | BLBC            | OWN_STORAGE, 10\$                |  | ; 1004 |
|           |    |    | 52   | D0    | 00108       | MOVL            | R2, IAC\$GL_MAIN_ICB             |  | ; 1005 |
|           | 04 | AE | 9E   | 0010F | 10\$: MOVAB | IHD_CTX, 88(R2) |                                  |  | ; 1006 |
|           |    |    | 00   | FB    | 00114       | CALLS           | #0, GET_LOCK                     |  | ; 1008 |
|           |    |    | 50   | D0    | 00119       | MOVL            | R0, STATUS                       |  | ;      |
|           |    |    | 59   | E8    | 0011C       | BLBS            | STATUS, 11\$                     |  | ; 1009 |
|           |    |    | 00D1 | 31    | 0011F       | BRW             | 21\$                             |  | ;      |
|           |    |    | 52   | DD    | 00122       | 11\$: PUSHL     | R2                               |  | ; 1011 |
| FD        | BE | CA | 9F   | 00124 |             | PUSHAB          | RESULT_NAME                      |  | ; 1016 |
| FC        | AE | CA | 9F   | 00128 |             | PUSHAB          | PRIMARY_NAM                      |  | ; 1015 |
| FC        | 5E | CA | 9F   | 0012C |             | PUSHAB          | PRIMARY_FAB                      |  | ; 1014 |
|           | F4 | AA | 9F   | 00130 |             | PUSHAB          | OWN_STORAGE+56                   |  | ; 1013 |
|           | EC | AA | 9F   | 00133 |             | PUSHAB          | OWN_STORAGE+48                   |  | ; 1012 |
|           |    |    | 06   | FB    | 00136       | CALLS           | #6, IMG\$OPEN_IMAGE              |  | ;      |
|           |    |    | 50   | D0    | 0013B       | MOVL            | R0, STATUS                       |  | ;      |
|           |    |    | 59   | E9    | 0013E       | BLBC            | STATUS, 13\$                     |  | ; 1018 |
| F4        | 5E | CA | 9E   | 00141 |             | MOVAB           | INPUT_BUFFER, IHD_CTX            |  | ; 1031 |
| F6        | 5E | CA | 9E   | 00147 |             | MOVAB           | PRIMARY_IHD, IHD_CTX+4           |  | ; 1032 |
|           |    |    | 52   | DD    | 0014D       | PUSHL           | R2                               |  | ; 1034 |
|           |    |    | 01   | FB    | 0014F       | CALLS           | #1, IMG\$GET_HEADER              |  | ;      |
|           |    |    | 50   | D0    | 00154       | MOVL            | R0, STATUS                       |  | ;      |
|           |    |    | 59   | E8    | 00157       | BLBS            | STATUS, 12\$                     |  | ; 1035 |
|           | 0E | A2 | 3C   | 0015A |             | MOVZWL          | 14(R2), -(SP)                    |  | ; 1038 |
|           |    |    | 01   | FB    | 0015E       | CALLS           | #1, SYS\$DASSGN                  |  | ;      |
|           |    |    | 00   | FB    | 00165       | CALLS           | #0, ERROR_CLEAN_UP               |  | ; 1039 |
|           |    |    | 15   | 11    | 0016A       | BRB             | 13\$                             |  | ; 1040 |
| 12        |    | AE | B1   | 0016C | 12\$: CMPW  | IHD_CTX+14, #-1 |                                  |  | ; 1049 |
|           |    |    | 16   | 13    | 00172       | BEQL            | 14\$                             |  | ;      |
|           |    |    | 52   | DD    | 00174       | PUSHL           | R2                               |  | ; 1052 |



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 17

X-14 EXE\$IMGACT - Image Activator System Service Rou 10-Apr-1989 10:48:52 \_\$25

|           |    |       |       |       |        |                         |        |
|-----------|----|-------|-------|-------|--------|-------------------------|--------|
|           | 01 | FB    | 00176 |       | CALLS  | #1, GET_OTHER_IMAGE     | ;      |
|           | 50 | D0    | 0017B |       | MOVL   | R0, STATUS              | ;      |
|           | 59 | E8    | 0017E |       | BLBS   | STATUS, 14\$            | ; 1053 |
|           | 52 | DD    | 00181 | 13\$: | PUSHL  | R2                      | ; 1056 |
|           | 01 | FB    | 00183 |       | CALLS  | #1, IMG\$DEALLOCATE_ICB | ;      |
|           | 64 | 11    | 00188 |       | BRB    | 20\$                    | ; 1057 |
|           | 05 | E1    | 0018A | 14\$: | BBC    | #5, FLAGS, 15\$         | ; 1070 |
| 00000000G | 00 | D0    | 0018E |       | MOVL   | CTL\$GL_PHD, R0         | ; 1075 |
| 009C      | C0 | D0    | 00195 |       | MOVL   | 156(R0), 72(R2)         | ; 1077 |
| 3FFFFFFF  | 8F | D0    | 0019B |       | MOVL   | #1073741823, 76(R2)     | ; 1078 |
|           | 05 | 11    | 001A3 |       | BRB    | 16\$                    | ;      |
| 04        | AA | 7D    | 001A5 | 15\$: | MOVQ   | OWN_STORAGE+72, 72(R2)  | ; 1083 |
| 20        | AA | 90    | 001AA | 16\$: | MOVB   | OWN_STORAGE+100, 12(R2) | ; 1086 |
|           | 04 | E1    | 001AF |       | BBC    | #4, FLAGS, 17\$         | ; 1088 |
|           | 02 | D0    | 001B3 |       | MOVL   | #2, R0                  | ; 1089 |
|           | 03 | 11    | 001B6 |       | BRB    | 18\$                    | ;      |
|           | 01 | D0    | 001B8 | 17\$: | MOVL   | #1, R0                  | ; 1090 |
|           | 50 | 90    | 001BB | 18\$: | MOVB   | R0, 13(R2)              | ; 1087 |
| 18        | AA | 7D    | 001BF |       | MOVQ   | OWN_STORAGE+92, 64(R2)  | ; 1092 |
| 00000000G | 00 | 9E    | 001C4 |       | MOVAB  | IAC\$GL_WORK_LIST+4, R0 | ; 1097 |
|           | 62 | 0E    | 001CB |       | INSQUE | (R2), @0(R0)            | ;      |
|           | 00 | FB    | 001CF |       | CALLS  | #0, IMG\$DO_WORK_LIST   | ; 1098 |
|           | 50 | D0    | 001D6 |       | MOVL   | R0, STATUS              | ;      |
|           | 59 | E9    | 001D9 |       | BLBC   | STATUS, 19\$            | ; 1099 |
|           | 6E | DD    | 001DC |       | PUSHL  | ICB_ADR                 | ; 1106 |
|           | 01 | FB    | 001DE |       | CALLS  | #1, END_PROCESSING      | ;      |
|           | 50 | D0    | 001E3 |       | MOVL   | R0, STATUS              | ;      |
|           | 59 | E8    | 001E6 |       | BLBS   | STATUS, 22\$            | ; 1107 |
|           | 00 | FB    | 001E9 | 19\$: | CALLS  | #0, ERROR_CLEAN_UP      | ; 1109 |
|           | 00 | FB    | 001EE | 20\$: | CALLS  | #0, RELEASE_LOCK        | ; 1110 |
|           | 59 | D0    | 001F3 | 21\$: | MOVL   | STATUS, R0              | ; 1111 |
|           | 04 | 001F6 |       |       | RET    |                         | ;      |
|           | 00 | FB    | 001F7 | 22\$: | CALLS  | #0, RELEASE_LOCK        | ; 1114 |
| CC        | AA | D0    | 001FC |       | MOVL   | OWN_STORAGE+16, R0      | ; 1115 |
|           | 04 | 00200 |       |       | RET    |                         | ;      |

; Routine Size: 513 bytes, Routine Base: EXEC\$PAGED\_CODE + 0000

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSSIMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 18

```
X-14      CHECK_PARAMS - Check Accessibility and Store Pa 10-Apr-1989 10:48:52  _$25
; 1119 1    %SBTTL 'CHECK_PARAMS - Check Accessibility and Store Parameters'
; 1120 1
; 1121 1    ROUTINE CHECK_PARAMS (NAME,DEFAULT,BUFFER,FLAGS,INADR,RETADR,IDENT,ACMODE) =
; 1122 1
; 1123 1    !+
; 1124 1    ! Functional Description:
; 1125 1    !
; 1126 1    !   This routine probes each input parameter for read access and stores
; 1127 1    !   the value of the parameter in a safe place. Output parameters are probe
; 1128 1    !   for write access and their addresses are stored for later use.
; 1129 1    !
; 1130 1    ! Calling Sequence:
; 1131 1    !
; 1132 1    !   CHECK_PARAMS (NAME,DEFAULT,BUFFER,FLAGS,INADR,RETADR,IDENT,ACMODE)
; 1133 1    !
; 1134 1    !-
; 1135 1
; 1136 2    BEGIN
; 1137 2
; 1138 2    LOCAL
; 1139 2        SAFE_PLACE,
; 1140 2        STATUS;
; 1141 2
; 1142 2    ! Define some synonyms for the local storage called SAFE_PLACE
; 1143 2
; 1144 2    BIND
; 1145 2        LOCAL_NAME = SAFE_PLACE : REF $BBLOCK,
; 1146 2        LOCAL_DFLT = SAFE_PLACE : REF $BBLOCK,
; 1147 2        LOCAL_BUFFER = SAFE_PLACE,
; 1148 2        LOCAL_INADR = INADR      : REF VECTOR,
; 1149 2        LOCAL_RETADR = RETADR    : REF VECTOR,
; 1150 2        LOCAL_IDENT = SAFE_PLACE : REF VECTOR;
; 1151 2
; 1152 2    ! Define some more synonyms for the two file name descriptors
; 1153 2
; 1154 2    BIND
; 1155 2        NAM_DSC = OWN_STORAGE [IMAGE_NAME_DESC] : $BBLOCK,
; 1156 2        DFLT_DSC = OWN_STORAGE [DFLT_NAME_DESC]  : $BBLOCK;
; 1157 2
; 1158 2    ! The image name is the only required parameter for the image activator. If
; 1159 2    ! image name is not present, this routine returns with an IMG$_NONAME error.
; 1160 2
; 1161 2    LOCAL_NAME = .NAME;
; 1162 2    IF .LOCAL_NAME EQL 0
; 1163 2    THEN
; 1164 2        RETURN IMG$_NONAME
; 1165 2    ELSE
; 1166 3        BEGIN
; 1167 4            IF NOT (STATUS = EXE$PROBER_DSC (
; 1168 4                .LOCAL_NAME;
; 1169 4                NAM_DSC [DSC$W_LENGTH],
; 1170 4                NAM_DSC [DSC$A_POINTER]))
; 1171 3            THEN
; 1172 3                RETURN .STATUS
; 1173 2            END;
; 1174 2
; 1175 2    ! The default name descriptor is probed and stored in a similar fashion. No
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSSIMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 19

```
X-14          CHECK_PARAMS - Check Accessibility and Store Pa 10-Apr-1989 10:48:52    _$25
; 1176 2      ! error occurs if the default name is missing.
; 1177 2
; 1178 2      LOCAL_DFLT = .DEFAULT;
; 1179 2      IF .LOCAL_DFLT EQL 0
; 1180 2      THEN
; 1181 3          BEGIN
; 1182 3              DFLT_DSC [DSC$W_LENGTH] = 0;
; 1183 3              DFLT_DSC [DSC$A_POINTER] = 0;
; 1184 3          END
; 1185 2      ELSE
; 1186 3          BEGIN
; 1187 4              IF NOT (STATUS = EXE$PROBER_DSC (
; 1188 4                  .LOCAL_DFLT;
; 1189 4                  DFLT_DSC [DSC$W_LENGTH],
; 1190 4                  DFLT_DSC [DSC$A_POINTER]))
; 1191 3              THEN
; 1192 3                  RETURN .STATUS
; 1193 2          END;
; 1194 2
; 1195 2      ! The address of a 512-byte buffer is tucked away after the buffer is
; 1196 2      ! checked for write access.
; 1197 2      !
; 1198 2      ! NOTE WELL
; 1199 2      !
; 1200 2      ! Because the image activator issues calls to other memory management system
; 1201 2      ! services, the protection on this buffer may change. This buffer must be pr
; 1202 2      ! again by the completion code before anything is written to the buffer.
; 1203 2
; 1204 2      LOCAL_BUFFER = .BUFFER;
; 1205 3      IF (.LOCAL_BUFFER NEQ 0)
; 1206 2          AND
; 1207 3          (NOT PROBEW (%REF(0), %REF(RETURN_BUFFER_SIZE), .LOCAL_BUFFER))
; 1208 2      THEN RETURN SS$_ACCVIO
; 1209 2      ELSE OWN_STORAGE [BUFFER_ADDRESS] = .LOCAL_BUFFER;
; 1210 2
; 1211 2      !!! NEED TO PROPOGATE SOME OF THE FLAGS INFORMATION INTO OTHER OWN_STORAGE
; 1212 2      !!! CELLS. ALSO NEED TO DO CONSISTENCY CHECKS BETWEEN FLAGS AND OTHER
; 1213 2      !!! INPUT PARAMETERS.
; 1214 2
; 1215 2      ! The input address range array specifies the address range into which the
; 1216 2      ! image is to be mapped. It must be readable by the caller.
; 1217 2
; 1218 2      LOCAL_INADR = .INADR;
; 1219 3      IF (.LOCAL_INADR NEQ 0) AND (NOT PROBER (%REF(0), %REF(8), .LOCAL_INADR))
; 1220 2      THEN RETURN SS$_ACCVIO;
; 1221 2      IF .LOCAL_INADR NEQ 0
; 1222 2      THEN
; 1223 3          BEGIN
; 1224 3
; 1225 3          BIND
; 1226 3              FLAG_BITS = OWN_STORAGE [INPUT_FLAGS] : $BLOCK,
; 1227 3              ADDRESS_RANGE = OWN_STORAGE [INPUT_START_ADDRESS] : $BLOCK;
; 1228 3
; 1229 3              OWN_STORAGE [INPUT_START_ADDRESS] = .LOCAL_INADR [0];
; 1230 3              OWN_STORAGE [INPUT_END_ADDRESS] = .LOCAL_INADR [1];
; 1231 3
; 1232 3          ! We need to remember that this is the P0 part of a P1 merge operation
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SY\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 20

X-14 CHECK\_PARAMS - Check Accessibility and Store Pa 10-Apr-1989 10:48:52 \_\$25

```
; 1233 3      ! because certain steps taken in a REAL activation must not take place.
; 1234 3
; 1235 4      IF (
; 1236 5          (.FLAG_BITS [IAC$V_P1MERGE])
; 1237 4          AND
; 1238 5          (NOT (.ADDRESS_RANGE [VA$V_P1]))
; 1239 4          )
; 1240 3      THEN
; 1241 3          OWN_STORAGE [P1_MERGE_P0] = TRUE;
; 1242 3
; 1243 2      END;
; 1244 2
; 1245 2      ! The caller can specify the address of an array that will receive the
; 1246 2      ! address range into which a collection of images was mapped. This array
; 1247 2      ! is probed for write access and set to contain FFFFFFFF in both elements,
; 1248 2      ! indicating that no mapping has yet occurred. Note that this array, like
; 1249 2      ! the output buffer must be probed again by the completion code before the
; 1250 2      ! actual return address arra is written to make sure that the accessibility
; 1251 2      ! of the array has not changed as a side effect of a system service call
; 1252 2      ! issued by the image activator.
; 1253 2
; 1254 2      LOCAL_RETADR = .RETADR;
; 1255 3      IF (.LOCAL_RETADR NEQ 0) AND (NOT PROBEW (%REF(0), %REF(8), .LOCAL_RETADR))
; 1256 2      THEN RETURN SS$ ACCVIO;
; 1257 2      OWN_STORAGE [RETURN_ARRAY_ADDRESS] = .LOCAL_RETADR;
; 1258 2      OWN_STORAGE [RETURN_START_ADDRESS] = -1;
; 1259 2      OWN_STORAGE [RETURN_END_ADDRESS] = -1;
; 1260 2      IF .LOCAL_RETADR NEQ 0
; 1261 2      THEN
; 1262 3          BEGIN
; 1263 3              LOCAL_RETADR [0] = -1;
; 1264 3              LOCAL_RETADR [1] = -1;
; 1265 2          END;
; 1266 2
; 1267 2      ! The caller can specify explicit match control information that will be
; 1268 2      ! compared to the version number contained in the image that is actually
; 1269 2      ! activated.
; 1270 2
; 1271 2      LOCAL_IDENT = .IDENT;
; 1272 3      IF (.LOCAL_IDENT NEQ 0) AND (NOT PROBER (%REF(0), %REF(8), .LOCAL_IDENT))
; 1273 2      THEN RETURN SS$ ACCVIO;
; 1274 2      IF .LOCAL_IDENT NEQ 0
; 1275 2      THEN
; 1276 3          BEGIN
; 1277 3              OWN_STORAGE [MATCH_CONTROL] = .LOCAL_IDENT [0];
; 1278 3              OWN_STORAGE [VERSION] = .LOCAL_IDENT [1];
; 1279 2          END;
; 1280 2
; 1281 2      ! The final input parameter is the access mode that will be used for two
; 1282 2      ! purposes. The channel on which the image file is opened will have this
; 1283 2      ! access mode associated with it. The pages that are mapped will be owned
; 1284 2      ! by the mode specified by this parameter.
; 1285 2      !
; 1286 2      ! Because an access mode of kernel is meaningless, a missing ACCESS_MODE
; 1287 2      ! parameter is interpreted as USER mode. If a nonzero value is present in th
; 1288 2      ! argument list, it is first maximized with caller's mode and then stored in
; 1289 2      ! a safe place.
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYS\$IMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 21

```
X-14      CHECK_PARAMS - Check Accessibility and Store Pa 10-Apr-1989 10:48:52      _$25
; 1290 2
; 1291 2      IF .ACMODE<0,2> EQL 0
; 1292 2      THEN
; 1293 2          OWN_STORAGE [ACCESS_MODE] = PSL$C_USER
; 1294 2      ELSE
; 1295 2          EXE$MAXACMODE (.ACMODE<0,2>; OWN_STORAGE [ACCESS_MODE]);
; 1296 2
; 1297 2      RETURN SS$_NORMAL
; 1298 2
; 1299 1      END;                                ! End of routine CHECK_PARAMS
```

```
007C 00000 CHECK_PARAMS:
          .WORD      Save R2,R3,R4,R5,R6          ; 1121
00000000G 00 9E 00002      MOVAB      EXE$PROBER_DSC, R6          ;
00000000G 00 9E 00009      MOVAB      DFLT_DSC, R5          ;
          04 AC D0 00010      MOVL       NAME, LOCAL_NAME          ; 1161
          08 12 00014      BNEQ       1$          ; 1162
004D8CCC 8F D0 00016      MOVL       #5082316, R0          ; 1164
          04 0001D      RET          ;
          54 D0 0001E 1$:    MOVL       LOCAL_NAME, R1          ; 1167
          66 16 00021      JSB        EXE$PROBER_DSC          ;
          51 B0 00023      MOVW       R1, NAM_DSC          ; 1169
          52 D0 00027      MOVL       R2, NAM_DSC+4          ; 1170
          50 E9 0002B      BLBC      STATUS, 3$          ;
          08 AC D0 0002E      MOVL       DEFAULT, LOCAL_DFLT          ; 1178
          07 12 00032      BNEQ       2$          ; 1179
          65 B4 00034      CLRW       DFLT_DSC          ; 1182
          04 A5 D4 00036      CLRL      DFLT_DSC+4          ; 1183
          10 11 00039      BRB        4$          ;
          54 D0 0003B 2$:    MOVL       LOCAL_DFLT, R1          ; 1187
          66 16 0003E      JSB        EXE$PROBER_DSC          ;
          51 B0 00040      MOVW       R1, DFLT_DSC          ; 1189
          52 D0 00043      MOVL       R2, DFLT_DSC+4          ; 1190
          50 E8 00047 3$:    BLBS      STATUS, 4$          ;
          04 0004A      RET          ;
          0C AC D0 0004B 4$:    MOVL       BUFFER, LOCAL_BUFFER          ; 1204
          08 13 0004F      BEQL       5$          ; 1205
          00 0D 00051      PROBEW     #0, #512, (LOCAL_BUFFER,          ; 1207
          65 13 00057      BEQL       10$          ;
          54 D0 00059 5$:    MOVL       LOCAL_BUFFER, OWN_STORAGE+64          ; 1209
          14 AC D0 0005D      MOVL       LOCAL_INADR, R0          ; 1219
          51 D4 00061      CLRL      R1          ;
          50 D5 00063      TSTL      R0          ;
          08 13 00065      BEQL       6$          ;
          51 D6 00067      INCL      R1          ;
          00 0C 00069      PROBER     #0, #8, (R0)          ;
          4F 13 0006D      BEQL       10$          ;
          51 E9 0006F 6$:    BLBC      R1, 7$          ; 1221
          60 7D 00072      MOVQ      (R0), OWN_STORAGE+72          ; 1230
          06 E1 00076      BBC        #6, FLAG_BITS, 7$          ; 1236
          06 E0 0007B      BBS      #6, ADDRESS_RANGE+3, 7$          ; 1238
          20 88 00080      BISB2     #32, OWN_STORAGE          ; 1241
          18 AC D0 00084 7$:    MOVL       LOCAL_RETADR, R0          ; 1255
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 22

X-14 CHECK\_PARAMS - Check Accessibility and Store Pa 10-Apr-1989 10:48:52 \_\$25

```

51 D4 00088      CLRL  R1      ;
50 D5 0008A      TSTL  R0      ;
08 13 0008C      BEQL  8$      ;
51 D6 0008E      INCL  R1      ;
00 0D 00090      PROBEW #0, #8, (R0) ;
28 13 00094      BEQL  10$     ;
50 D0 00096 8$:  MOVL  R0, OWN_STORAGE+80 ; 1257
01 CE 0009A      MNEGL #1, OWN_STORAGE+84 ; 1258
01 CE 0009E      MNEGL #1, OWN_STORAGE+88 ; 1259
51 E9 000A2      BLBC  R1, 9$     ; 1260
01 CE 000A5      MNEGL #1, (R0)   ; 1263
01 CE 000A8      MNEGL #1, 4(R0)   ; 1264
1C AC D0 000AC 9$: MOVL  IDENT, LOCAL_IDENT ; 1271
50 D4 000B0      CLRL  R0      ; 1272
54 D5 000B2      TSTL  LOCAL_IDENT ;
0C 13 000B4      BEQL  11$     ;
50 D6 000B6      INCL  R0      ;
00 0C 000B8      PROBER #0, #8, (LOCAL_IDENT) ;
04 12 000BC      BNEQ  11$     ;
0C D0 000BE 10$: MOVL  #12, R0      ; 1273
04 000C1      RET      ;
50 E9 000C2 11$: BLBC  R0, 12$     ; 1274
64 7D 000C5      MOVQ  (LOCAL_IDENT), OWN_STORAGE+92 ; 1278
20 AC 93 000C9 12$: BITB  ACMODE, #3 ; 1291
06 12 000CD      BNEQ  13$     ;
03 D0 000CF      MOVL  #3, OWN_STORAGE+100 ; 1293
10 11 000D3      BRB   14$     ;
00 EF 000D5 13$: EXTZV #0, #2, ACMODE, R0 ; 1295
00000000G 00 16 000DB      JSB   EXE$MAXACMODE ;
50 D0 000E1      MOVL  R0, OWN_STORAGE+100 ;
01 D0 000E5 14$: MOVL  #1, R0      ; 1297
04 000E8      RET      ;

```

; Routine Size: 233 bytes, Routine Base: EXEC\$PAGED\_CODE + 0201

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 23

```

X-14          IMG$OPEN_IMAGE - Open Next Image File          10-Apr-1989 10:48:52    _$25

; 1301 1      %SBTTL 'IMG$OPEN_IMAGE - Open Next Image File'
; 1302 1
; 1303 1      GLOBAL ROUTINE IMG$OPEN_IMAGE (NAME_DESC , DFLT_DESC ,
; 1304 1          FAB_ADDRESS , NAM_BLOCK_ADDRESS ,
; 1305 1          NAME_STRING , ICB_POINTER) =
; 1306 1
; 1307 1      !+
; 1308 1      ! Functional Description:
; 1309 1      !
; 1310 1      ! This routine opens an image file as a process permanent file for
; 1311 1      ! subsequent use by one of the memory management system services.
; 1312 1      !
; 1313 1      ! Calling Sequence:
; 1314 1      !
; 1315 1      ! IMG$OPEN_IMAGE ()
; 1316 1      !
; 1317 1      ! Input Parameters:
; 1318 1      !
; 1319 1      ! NAME_DESC - Address of string descriptor for image file name
; 1320 1      !
; 1321 1      ! DFLT_DESC - Address of default name descriptor
; 1322 1      !-
; 1323 1
; 1324 2      BEGIN
; 1325 2
; 1326 2      MAP
; 1327 2          NAME_STRING : REF VECTOR;
; 1328 2
; 1329 2      BIND
; 1330 2          FAB          = .FAB_ADDRESS          : $BBLOCK,
; 1331 2          NAME_BLOCK = .NAM_BLOCK_ADDRESS      : $BBLOCK,
; 1332 2          ICB          = .ICB_POINTER          : $BBLOCK,
; 1333 2          DEV_CHAR    = FAB [FAB$!_DEV]        : $BBLOCK,
; 1334 2          ICB_NAME    = ICB [ICB$!_IMAGE_NAME] : VECTOR [, BYTE];
; 1335 2
; 1336 2      ! Create synonyms for the CTX and STV fields in the FAB
; 1337 2
; 1338 2      BIND
; 1339 2          CHAN          = FAB [FAB$!_STV],
; 1340 2          KFE          = FAB [FAB$!_CTX]          : REF $BBLOCK;
; 1341 2
; 1342 2      MAP
; 1343 2          NAME_DESC      : REF $BBLOCK,
; 1344 2          DFLT_DESC      : REF $BBLOCK;
; 1345 2
; 1346 2      LOCAL
; 1347 2          STATUS;
; 1348 2
; 1349 2      ! Once the system is fully initialized, RMS and the file system can be used
; 1350 2      ! to open the various image files. Until that time, the bootstrap file
; 1351 2      ! routines must be used. At least two images, SYSINIT.EXE and F11BXQP.EXE,
; 1352 2      ! are activated along this alternate code path.
; 1353 2
; 1354 2      IF .EXE$GL_FLAGS [EXE_V_INIT]
; 1355 2      THEN
; 1356 3          BEGIN
; 1357 3

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SY\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 24

X-14

IMG\$OPEN\_IMAGE - Open Next Image File

10-Apr-1989 10:48:52

\_ \$25

```

; P 1358 3      $FAB_INIT (
; P 1359 3      FAB = FAB,
; P 1360 3      FNS = .NAME_DESC [DSC$W_LENGTH],
; P 1361 3      FNA = .NAME_DESC [DSC$A_POINTER],
; P 1362 3      DNS = .DFLT_DESC [DSC$W_LENGTH],
; P 1363 3      DNA = .DFLT_DESC [DSC$A_POINTER],
; P 1364 3      NAM = NAME_BLOCK,
; P 1365 3      FOP = (
; P 1366 3          KFO ,                ! Use the known file data base
; P 1367 3          PPF ,                ! Process permanent file
; P 1368 3          SQO ,                ! Network optimization
; P 1369 3          UFO ),              ! User file open
; P 1370 3      CTX = 0,                ! KFE address will be returned here
; P 1371 3      RTV = -1                ! Insure that WCB completely maps file
; 1372 3      );
; 1373 3
; 1374 3      FAB [FAB$V_CHAN_MODE] = .OWN_STORAGE [ACCESS_MODE];
; 1375 3
; 1376 3      ! The image files associated with writable global sections are opened
; 1377 3      ! for write access. All other image files are opened for execute access.
; 1378 3
; 1379 3      IF .ICB [ICB$V_OPEN_FOR_WRITE]
; 1380 3      THEN
; 1381 4          BEGIN
; 1382 4          FAB [FAB$B_FAC] = FAB$M_PUT OR FAB$M_EXE;          !Setting the EXE bit
; 1383 4          FAB [FAB$B_SHR] = FAB$M_SHRGET OR FAB$M_SHRPUT OR FAB$M_UPI;
; 1384 4          END
; 1385 3      ELSE
; 1386 4          BEGIN
; 1387 4          FAB [FAB$B_FAC] = FAB$M_EXE;
; 1388 4          FAB [FAB$B_SHR] = 0;
; 1389 3          END;
; 1390 3
; 1391 3      ! If we are currently running an executable image installed with privile
; 1392 3      ! or this image or an ancestor of this image was installed /PROT, then
; 1393 3      ! we must direct RMS to only use the logical name tables that cannot be
; 1394 3      ! redefined by user mode code.
; 1395 3
; 1396 3      FAB [FAB$V_LNM_MODE] =
; 1397 4      (IF .IAC$GL_IMAGCTX [IMAGCTX$V_PRIV] OR .ICB [ICB$V_PARENT_PROT]
; 1398 4      OR .ICB [ICB$V_PROTECTED]
; 1399 4      THEN PSL$C_EXEC
; 1400 3      ELSE PSL$C_USER);
; 1401 3
; 1402 3      ! Load NAM block with descriptor for resultant (or merely expanded) name
; 1403 3      ! string. (If a known file entry is found, then RMS does not return an
; 1404 3      ! resultant string but merely the expanded string.)
; 1405 3
; P 1406 3      $NAM_INIT
; P 1407 3      (NAM = NAME_BLOCK,
; P 1408 3      RSS = NAM$C_MAXRSS,
; P 1409 3      RSA = .NAME_STRING,
; P 1410 3      ESS = NAM$C_MAXRSS,
; P 1411 3      ESA = .NAME_BLOCK [NAM$L_RSA],
; P 1412 3      NOP = <NOCONCEAL>
; 1413 3      );
; 1414 3

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSSIMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 25  
X-14

IMG\$OPEN\_IMAGE - Open Next Image File      10-Apr-1989 10:48:52      \_\$25

```

; 1415 3      STATUS = $OPEN (FAB = FAB);
; 1416 3
; 1417 3      !!! THE FOLLOWING HACK ALLOWS INFORMATION IN THE FAB TO BE PASSED      !!!!
; 1418 3      !!! BACK TO THE CALLER IN THE EVENT THAT THE ACTIVATION FAILS      !!!
; 1419 3      !!!
; 1420 3      if .own_storage [buffer_address] nequ 0
; 1421 3      then
; 1422 4          begin
; 1423 4
; 1424 4          bind
; 1425 4              bufhdr = .own_storage [buffer_address]      : vector [3,long];
; 1426 4
; 1427 4          if probew (%ref(0), %ref(4), bufhdr [2])
; 1428 4          then
; 1429 4              bufhdr [2] = fab;
; 1430 4
; 1431 3          end;
; 1432 3      !!!
; 1433 3      !!! THIS IS THE END OF THE ERROR REPORTING HACK      !!!!
; 1434 3
; 1435 3      IF NOT .STATUS THEN RETURN .STATUS; ! ??? WHAT ELSE ???
; 1436 3      own_storage[fab_fac] = .fab[fab$b_fac];      !Save the one's complement o
; 1437 3                                          !This is part of V4's ECO 24
; 1438 3      END
; 1439 2      ELSE
; 1440 3      BEGIN
; 1441 3
; 1442 3      ! This is the code path that is used to open image files before the file
; 1443 3      ! system and RMS exist.
; 1444 3
; 1445 3      LOCAL
; 1446 3          RTVRBUF : VECTOR [512, BYTE],      ! Retrieval pointer buffer
; 1447 3          IXFHDR  : VECTOR [512, BYTE],      ! Index file header buffer
; 1448 3          FILHDR  : VECTOR [512, BYTE],      ! File header buffer
; 1449 3          STATBLK : VECTOR [2],              ! Statistics block
; 1450 3          RTVRLEN,
; 1451 3          BUF_DESC : VECTOR [2]              ! Retrieval pointer buffer descripto
; 1452 3          INITIAL (512, RTVRBUF);
; 1453 3
; 1454 3      LOCAL
; 1455 3          ARG_LIST : VECTOR [3]
; 1456 3          INITIAL (2, 0, BUF_DESC);
; 1457 3
; 1458 3      STATUS = FIL$OPENFILE (
; 1459 3          FAB [FAB$L_STV],
; 1460 3          .NAME_DESC,
; 1461 3          IXFHDR,
; 1462 3          FILHDR,
; 1463 3          STATBLK,
; 1464 3          RTVRLEN,
; 1465 3          BUF_DESC);
; 1466 3      IF NOT .STATUS
; 1467 3      THEN RETURN .STATUS;
; 1468 3
; 1469 3      IF .RTVRLEN GTR .BUF_DESC [0]
; 1470 3      THEN RETURN SS$_BADIMGHDR;
; 1471 3

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 26

```

X-14          IMG$OPEN_IMAGE - Open Next Image File          10-Apr-1989 10:48:52    _$25
; 1472 3      IF .RTVRLN EQL 0                                ! If the file is empty,
; 1473 3      THEN RETURN SS$_BADIMGHDR                       ! then return an error
; 1474 3      ELSE BUF_DESC [0] = .RTVRLN;                   ! Otherwise, use correct buffer size
; 1475 3
; 1476 3      ARG_LIST [1] = .CHAN;                            ! Store channel number in argument 1
; P 1477 3      STATUS = $CMKRNL (                             ! Allocate and initialize a window
; P 1478 3          ROUTIN = INIT_WINDOW,                      ! control block in kernel mode
; 1479 3          ARGLST = ARG_LIST);
; 1480 3      IF NOT .STATUS
; 1481 3      THEN RETURN .STATUS;                            ! Exit if error detected
; 1482 3
; 1483 3      ! The image file name string passed as input will be stored internally a
; 1484 3      ! passed back to the caller if so requested. The entire file specificati
; 1485 3      ! (and not merely the file name portion) will also be stored in the ICB.
; 1486 3
; 1487 3      NAME_BLOCK [NAM$B_RSL] = .NAME_DESC [DSC$W_LENGTH];
; 1488 3      CH$MOVE (
; 1489 3          .NAME_BLOCK [NAM$B_RSL],                      ! Store the input file specification
; 1490 3          .NAME_DESC [DSC$A_POINTER],                  ! in the name string buffer
; 1491 3          .NAME_STRING);
; 1492 3
; 1493 3      ! Rather than storing the name in the ICB, this code segment will merely
; 1494 3      ! set up the NAM block fields so that the code below that loads the imag
; 1495 3      ! name into the ICB will work correctly. Note that ICB$$_IMAGE_NAME incl
; 1496 3      ! one byte of count and four bytes of _00n suffix. These must be subtrac
; 1497 3      ! before the minimization takes place.
; 1498 3
; 1499 3      NAME_BLOCK [NAM$B_NAME] = MINU (                    ! Only use the beginning of the stri
; 1500 3          .NAME_DESC [DSC$W_LENGTH],                      ! if the input file specification i
; 1501 3          ICB$$_IMAGE_NAME - (1+4));                       ! longer than 39 characters
; 1502 3      NAME_BLOCK [NAM$L_NAME] = .NAME_STRING;
; 1503 3
; 1504 2      END;
; 1505 2
; 1506 2      ICB [ICB$W_CHAN] = .CHAN;                          ! Store channel number in ICB
; 1507 2      ICB [ICB$L_KFE] = .FAB [FAB$L_CTX];              ! Pick up address of KFE
; 1508 2
; 1509 2      ! If ICB does not yet contain an image name, load the name contained in the
; 1510 2      ! NAM block. The name stored in the ICB will be the filename portion of the
; 1511 2      ! full file spec, the same name that the linker uses when constructing the
; 1512 2      ! names of global ISDs.
; 1513 2
; 1514 2      IF .ICB_NAME [0] EQL 0
; 1515 2      THEN
; 1516 3          BEGIN
; 1517 3          ICB_NAME [0] = .NAME_BLOCK [NAM$B_NAME];
; 1518 3          CH$MOVE (
; 1519 3              .ICB_NAME [0],
; 1520 3              .NAME_BLOCK [NAM$L_NAME],
; 1521 3              ICB_NAME [1]);
; 1522 2          END;
; 1523 2
; 1524 2      ! If activating an image across the network or from magtape, the file is
; 1525 2      ! not mapped. Rather, the address space is created and the contents of the
; 1526 2      ! image file are read directly into the newly created pages.
; 1527 2
; 1528 3      IF (.DEV_CHAR [DEV$V_NET] OR

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

32 V4.5-862 Page 27      SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-

```

X-14                    IMG$OPEN_IMAGE - Open Next Image File                    10-Apr-1989 10:48:52      _$25
;    1529    3                    (.DEV_CHAR [DEV$V_SQD] AND (.FAB [FAB$W_BLS] EQL 512)))
;    1530    2                    THEN ICB [ICB$V_LOAD_IMAGE] = TRUE;
;    1531    2
;    1532    2                    IF .ICB [ICB$L_KFE] EQL 0
;    1533    2                    THEN
;    1534    3                    BEGIN
;    1535    3                    IF (.IAC$GL_IMAGCTX [IMAGCTX$V_PRIV]) THEN RETURN SS$_PRIVINSTALL
;    1536    3                    END
;    1537    2                    ELSE
;    1538    3                    BEGIN
;    1539    3
;    1540    3                    ! Make sure that the count byte immediately precedes the name string
;    1541    3
;    1542    3                    $ASSUME (($BYTEOFFSET (KFE$B_FILNAMLEN))
;    1543    3                    EQL
;    1544    3                    ($BYTEOFFSET (KFE$T_FILNAM) - 1));
;    1545    3
;    1546    3                    BIND
;    1547    3                    KFE_NAME = KFE [KFE$T_FILNAM] - 1 : VECTOR [,BYTE],                    ! Include co
;    1548    3                    CTX = .ICB [ICB$L_CONTEXT]                    : $BBLOCK,
;    1549    3                    GSD_NAME = CTX [CTX_T_GSD_NAME]                    : VECTOR [,BYTE];
;    1550    3
;    1551    3                    ! Several bits of housekeeping are in order if this was a successful
;    1552    3                    ! $OPEN of a known file.
;    1553    3
;    1554    3                    IF .KFE [KFE$V_PROCPRIV] AND .OWN_STORAGE [MAIN_PROGRAM]
;    1555    3                    THEN IAC$GL_IMAGCTX [IMAGCTX$V_PRIV] = TRUE;
;    1556    3
;    1557    3                    ICB [ICB$V_PROTECTED] = .KFE [KFE$V_PROTECT];
;    1558    3
;    1559    3                    ! Images installed with the /ACCOUNT qualifier can cause image accountin
;    1560    3                    ! records to be written each time they are run as main programs. The
;    1561    3                    ! ACCOUNT flag is ignored for shareable images and images that are merge
;    1562    3                    ! into an existing image's address space.
;    1563    3
;    1564    3                    IF .KFE [KFE$V_ACCOUNT] AND .OWN_STORAGE [MAIN_PROGRAM]
;    1565    3                    THEN OWN_STORAGE [IMAGE_ACCOUNT] = TRUE;
;    1566    3                    !
;    1567    3                    !Set things up for execute-only access (ECO 24)
;    1568    3                    !
;    1569    3                    IF .kfe[kfe$V_exeonly] AND .own_storage[main_program]
;    1570    4                    AND (.own_storage[call_mode] EQL PSL$C_USER)
;    1571    4                    THEN BEGIN
;    1572    4                                       iac$gl_imagctx [imagctx$V_priv] = true;
;    1573    4                                       iac$gl_imagctx [imagctx$V_exeonly] = true;
;    1574    3                                       END;
;    1575    3
;    1576    3                    ! RMS returned an expanded string. We will store the size of the image
;    1577    3                    ! file in the RSL field in the NAM block. We will also strip off the
;    1578    3                    ! trailing semicolon (or dot) that would inhibit further known file
;    1579    3                    ! lookups if the file name returned by the image activator were to be
;    1580    3                    ! passed back into the image activator at a later time.
;    1581    3
;    1582    3                    IF .NAME_BLOCK [NAM$B_VER] EQL 1
;    1583    3                    THEN NAME_BLOCK [NAM$B_RSL] = .NAME_BLOCK [NAM$B_ESL] - 1;
;    1584    3
;    1585    3                    IF .KFE [KFE$V_SHARED]

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSSIMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 28

X-14 IMG\$OPEN\_IMAGE - Open Next Image File 10-Apr-1989 10:48:52 \_\$25

```

; 1586 3      THEN
; 1587 4      BEGIN
; 1588 4      CH$MOVE (
; 1589 4          (.KFE [KFE$B_FILNAMLEN] + 1),
; 1590 4          KFE_NAME [0],
; 1591 4          GSD_NAME [0]);
; 1592 4      CH$MOVE (
; 1593 4          4,
; 1594 4          CH$PTR (UPLIT ('_000')),
; 1595 4          GSD_NAME [.KFE [KFE$B_FILNAMLEN] + 1]);
; 1596 4      ICB [ICB$V_SHAREABLE] = TRUE;
; 1597 3      END;
; 1598 3
; 1599 3      !!! SOME MORE KFE FLAGS SHOULD BE COPIED INTO THE ICB
; 1600 3
; 1601 3      !!! THE CURRENT CODE WORRIES ABOUT THIS SHMIDENT STUFF
; 1602 3
; 1603 2      END;
; 1604 2
; 1605 2      IF ( (.own_storage[fab_fac] AND FAB$M_EXE) NEQ 0) AND      !if execute access a
; 1606 3      ( (.own_storage[fab_fac] AND FAB$M_GET) EQL 0)          !no read access
; 1607 2      THEN
; 1608 3      BEGIN
; 1609 3      IF .own_storage[main_program]          !Is it a main program?
; 1610 3      THEN
; 1611 3          iac$gl_imagctx[imagctx$v_priv] = true      !set the paranoia bit
; 1612 3      ELSE
; 1613 4      BEGIN
; 1614 4      IF .iac$gl_imagctx[imagctx$v_exeonly]      !Installed execute_only?
; 1615 4      THEN
; 1616 4          RETURN SS$_NORMAL          !yes
; 1617 4      ELSE
; 1618 4          RETURN SS$_ACCONFLICT      !no
; 1619 4      END
; 1620 2      END;
; 1621 2
; 1622 2      RETURN SS$_NORMAL;
; 1623 2
; 1624 1      END;          ! End of routine IMG$OPEN_IMAGE

```

```

.PSECT EXEC$PAGED_DATA, PIC,2
0000000 00000002 0000 P.AAA: .LONG 2, 0 ;
          00000000 00008 .LONG 0 ;
30 30 30 5F 0000C P.AAB: .ASCII \_000\ ;
.EXTRN SYS$OPEN, SYS$CMKRNL
.PSECT EXEC$PAGED_CODE, NOWRT, PIC,2
OFFC 00000 .ENTRY IMG$OPEN_IMAGE, Save R2,R3,R4,R5,R6,R7,R8,-
; 1303      R9,R10,R11 ;
00000000G 00 9E 00002 MOVAB OWN_STORAGE, R11 ;
          F9E0 CE 9E 00009 MOVAB -1568(SP), SP ;
          0C AC 7D 0000E MOVQ FAB_ADDRESS, R6 ; 1331

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYS\$IMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 29

```

X-14          IMG$OPEN_IMAGE - Open Next Image File          10-Apr-1989 10:48:52      _$25

    18  AC  D0 00012      MOVL   ICB_POINTER, R8          ; 1332
    14  A8  9E 00016      MOVAB  20(R8), R10             ; 1334
    04  AC  D0 0001A      MOVL   NAME_DESC, R9          ; 1372
          00G E0 0001E      BBS    S^EXE$V_INIT, EXE$GL_FLAGS, 1$ ; 1354
    00C9 31 00026      BRW    8$                    ;
          00 2C 00029 1$:  MOVCS5  #0, (SP), #0, #80, (R6)      ; 1372
          66          00030      ;
    5003 8F  B0 00031      MOVW   #20483, (R6)           ;
    40060040 8F  D0 00036      MOVL   #1074135104, 4(R6)     ;
          02  90 0003E      MOVB   #2, 22(R6)             ;
          01  8E 00042      MNEGB  #1, 28(R6)             ;
          02  90 00046      MOVB   #2, 31(R6)             ;
          57  D0 0004A      MOVL   R7, 40(R6)            ;
    04  A9  D0 0004E      MOVL   4(R9), 44(R6)          ;
    08  AC  D0 00053      MOVL   DFLT_DESC, R0          ;
    04  A0  D0 00057      MOVL   4(R0), 48(R6)          ;
    04  BC  90 0005C      MOVB   @NAME_DESC, 52(R6)     ;
          60  90 00061      MOVB   (R0), 53(R6)           ;
    64  AB  F0 00065      INSV   OWN_STORAGE+100, #2, #2, 74(R6) ; 1374
          02  E1 0006C      BBC    #2, 16(R8), 2$        ; 1379
    16  A6  9E 00071      MOVAB  22(R6), R9             ; 1382
    81  8F  90 00075      MOVB   #-127, (R9)            ;
    43  8F  90 00079      MOVB   #67, 23(R6)           ; 1383
          0B  11 0007E      BRB    3$                    ;
    16  A6  9E 00080 2$:  MOVAB  22(R6), R9             ; 1387
    80  8F  90 00084      MOVB   #-128, (R9)           ;
    17  A6  94 00088      CLRB   23(R6)                 ; 1388
          01  E0 0008B 3$:  BBS    #1, IAC$GL_IMAGCTX, 4$      ; 1397
          03  E0 00093      BBS    #3, 17(R8), 4$        ;
          02  E1 00098      BBC    #2, 17(R8), 5$        ; 1398
          01  D0 0009D 4$:  MOVL   #1, R0                    ; 1399
          03  11 000A0      BRB    6$                    ;
          03  D0 000A2 5$:  MOVL   #3, R0                    ; 1400
          50  F0 000A5 6$:  INSV   R0, #0, #2, 74(R6)           ; 1396
          00  2C 000AB      MOVCS5  #0, (SP), #0, #96, (R7) ; 1413
          67          000B2      ;
    6002 8F  B0 000B3      MOVW   #24578, (R7)           ;
          01  8E 000B8      MNEGB  #1, 2(R7)              ;
    14  AC  D0 000BC      MOVL   NAME_STRING, 4(R7)     ;
          10  90 000C1      MOVB   #16, 8(R7)             ;
          01  8E 000C5      MNEGB  #1, 10(R7)            ;
    04  A7  D0 000C9      MOVL   4(R7), 12(R7)          ;
          56  DD 000CE      PUSHL  R6                    ; 1415
          01  FB 000D0      CALLS  #1, SYS$OPEN           ;
    40  AB  D0 000D7      MOVL   OWN_STORAGE+64, R1     ; 1420
          0B  13 000DB      BEQL   7$                    ;
          00  0D 000DD      PROBEW #0, #4, 8(R1)         ; 1427
          04  13 000E2      BEQL   7$                    ;
          56  D0 000E4      MOVL   R6, 8(R1)              ; 1429
          50  E9 000E8 7$:  BLBC   STATUS, 11$                ; 1435
          69  90 000EB      MOVB   (R9), OWN_STORAGE+37   ; 1436
    0086 31 000EF      BRW    14$                    ;
    0200 8F  3C 000F2 8$:  MOVZWL  #512, BUF_DESC           ; 1452
    FE00 CD  9E 000F8      MOVAB  RTVRBUF, BUF_DESC+4    ;
          0C  28 000FE      MOVCS3  #12, P.AAA, ARG_LIST  ; 1456
    10  AE  9E 00105      MOVAB  BUF_DESC, ARG_LIST+8   ;
    10  AE  9F 0010A      PUSHAB BUF_DESC              ; 1465

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 30

| X-14  |    | IMG\$OPEN_IMAGE - Open Next Image File | 10-Apr-1989 10:48:52           | _\$25  |
|-------|----|--|--------------------------------|--------|
| 04    | AE | 9F 0010D                               | PUSHAB RTVRLEN                 | ; 1464 |
| 20    | AE | 9F 00110                               | PUSHAB STATBLK                 | ; 1463 |
| 2C    | AE | 9F 00113                               | PUSHAB FILHDR                  | ; 1462 |
| 0230  | CE | 9F 00116                               | PUSHAB IXFHDR                  | ; 1461 |
| 04    | AC | DD 0011A                               | PUSHL NAME_DESC                | ;      |
| 0C    | A6 | 9F 0011D                               | PUSHAB 12(R6)                  | ; 1459 |
|       | 07 | FB 00120                               | CALLS #7, FIL\$OPENFILE        | ; 1458 |
|       | 50 | E9 00127                               | BLBC STATUS, 11\$              | ; 1466 |
|       | 6E | D1 0012A                               | CMPL RTVRLEN, BUF_DESC         | ; 1469 |
|       | 04 | 14 0012E                               | BGTR 9\$                       | ;      |
|       | 6E | D5 00130                               | TSTL RTVRLEN                   | ; 1472 |
|       | 05 | 12 00132                               | BNEQ 10\$                      | ;      |
| 44    | 8F | 9A 00134 9\$:                          | MOVZBL #68, R0                 | ; 1473 |
|       |    | 04 00138                               | RET                            | ;      |
|       | 6E | D0 00139 10\$:                         | MOVL RTVRLEN, BUF_DESC         | ; 1474 |
| 0C    | A6 | D0 0013D                               | MOVL 12(R6), ARG_LIST+4        | ; 1476 |
| 04    | AE | 9F 00142                               | PUSHAB ARG_LIST                | ; 1479 |
| 0000V | CF | 9F 00145                               | PUSHAB INIT_WINDOW             | ;      |
|       | 02 | FB 00149                               | CALLS #2, SYS\$CMKRN           | ;      |
|       | 50 | E8 00150 11\$:                         | BLBS STATUS, 12\$              | ; 1480 |
|       |    | 04 00153                               | RET                            | ;      |
| 04    | BC | 90 00154 12\$:                         | MOVB @NAME_DESC, 3(R7)         | ; 1487 |
| 03    | A7 | 9A 00159                               | MOVZBL 3(R7), R0               | ; 1489 |
|       | 50 | 28 0015D                               | MOVC3 R0, @4(R9), @NAME_STRING | ; 1488 |
| 04    | BC | 3C 00163                               | MOVZWL @NAME_DESC, R0          | ; 1501 |
|       | 50 | B1 00167                               | CMPW R0, #35                   | ;      |
|       | 03 | 1B 0016A                               | BLEQU 13\$                     | ;      |
|       | 23 | D0 0016C                               | MOVL #35, R0                   | ;      |
|       | 50 | 90 0016F 13\$:                         | MOVB R0, 59(R7)                | ;      |
| 14    | AC | D0 00173                               | MOVL NAME_STRING, 76(R7)       | ; 1502 |
| 0C    | A6 | B0 00178 14\$:                         | MOVW 12(R6), 14(R8)            | ; 1506 |
| 18    | A6 | D0 0017D                               | MOVL 24(R6), 84(R8)            | ; 1507 |
|       | 6A | 95 00182                               | TSTB (R10)                     | ; 1514 |
|       | 0D | 12 00184                               | BNEQ 15\$                      | ;      |
| 3B    | A7 | 90 00186                               | MOVB 59(R7), (R10)             | ; 1517 |
|       | 6A | 9A 0018A                               | MOVZBL (R10), R0               | ; 1519 |
|       | 50 | 28 0018D                               | MOVC3 R0, @76(R7), 1(R10)      | ; 1521 |
|       | 05 | E0 00193 15\$:                         | BBS #5, 65(R6), 16\$           | ; 1528 |
|       | 05 | E1 00198                               | BBC #5, 64(R6), 17\$           | ; 1529 |
| 3C    | A6 | B1 0019D                               | CMPW 60(R6), #512              | ;      |
|       | 04 | 12 001A3                               | BNEQ 17\$                      | ;      |
|       | 10 | 88 001A5 16\$:                         | BISB2 #16, 16(R8)              | ; 1530 |
| 54    | A8 | D5 001A9 17\$:                         | TSTL 84(R8)                    | ; 1532 |
|       | 0E | 12 001AC                               | BNEQ 18\$                      | ;      |
|       | 01 | E1 001AE                               | BBC #1, IAC\$G_L_IMAGCTX, 23\$ | ; 1535 |
| 2054  | 8F | 3C 001B6                               | MOVZWL #8276, R0               | ;      |
|       |    | 04 001BB                               | RET                            | ;      |
| 18    | A6 | D0 001BC 18\$:                         | MOVL 24(R6), R0                | ; 1547 |
|       | 14 | C1 001C0                               | ADDL3 #20, 88(R8), R6          | ; 1549 |
| 10    | A0 | 9E 001C5                               | MOVAB 16(R0), R1               | ; 1554 |
|       | 02 | E1 001C9                               | BBC #2, (R1), 19\$             | ;      |
|       | 6B | E9 001CD                               | BLBC OWN_STORAGE, 19\$         | ;      |
|       | 02 | 88 001D0                               | BISB2 #2, IAC\$G_L_IMAGCTX     | ; 1555 |
|       | 61 | F0 001D7 19\$:                         | INSV (R1), #2, #1, 17(R8)      | ; 1557 |
|       | 09 | E1 001DD                               | BBC #9, (R1), 20\$             | ; 1564 |
|       | 6B | E9 001E1                               | BLBC OWN_STORAGE, 20\$         | ;      |
|       | 04 | 88 001E4                               | BISB2 #4, OWN_STORAGE          | ; 1565 |

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 31

```

X-14      IMG$OPEN_IMAGE - Open Next Image File          10-Apr-1989 10:48:52      _$25
          0B E1 001E7 20$:   BBC      #11, (R1), 21$           ; 1569
          6B E9 001EB       BLBC     OWN_STORAGE, 21$         ;
24        AB 91 001EE       CMPB     OWN_STORAGE+36, #3        ; 1570
          07 12 001F2       BNEQ     21$                     ;
          03 88 001F4       BISB2    #3, IAC$GL_IMAGCTX        ; 1572
3D        A7 91 001FB 21$:   CMPB     61(R7), #1             ; 1582
          06 12 001FF       BNEQ     22$                     ;
          01 83 00201       SUBB3    #1, 11(R7), 3(R7)         ; 1583
          05 E1 00207 22$:   BBC      #5, (R1), 23$           ; 1585
36        A0 9A 0020B       MOVZBL  54(R0), R7             ; 1589
          57 D6 0020F       INCL     R7                     ;
          57 28 00211       MOV3    R7, 54(R0), (R6)         ; 1588
6746     9F 00216       PUSHAB (R7)[R6]             ; 1594
0000'    CF D0 00219       MOVL   P.AAB, @(SP)+      ;
          02 88 0021E       BISB2    #2, 16(R8)         ; 1596
25        AB 9A 00222 23$:   MOVZBL  OWN_STORAGE+37, R0 ; 1605
          50 95 00226       TSTB    R0                     ;
          1D 18 00228       BGEQ    25$                     ;
          01 E0 0022A       BBS     #1, R0, 25$         ; 1606
          6B E9 0022E       BLBC     OWN_STORAGE, 24$   ; 1609
          02 88 00231       BISB2    #2, IAC$GL_IMAGCTX ; 1611
          0D 11 00238       BRB     25$                     ;
00000000G 00 E8 0023A 24$:   BLBS    IAC$GL_IMAGCTX, 25$ ; 1614
0800     8F 3C 00241       MOVZWL  #2048, R0          ; 1618
          04 00246       RET                                     ;
          01 D0 00247 25$:   MOVL    #1, R0             ; 1622
          04 0024A       RET                                     ;

```

; Routine Size: 587 bytes, Routine Base: EXEC\$PAGED\_CODE + 02EA

; 1625 1

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 32

```

X-14          INIT_WINDOW - Interface Routine to FIL$INIWCB   10-Apr-1989 10:48:52   _$25
; 1627 1      %SBTTL 'INIT_WINDOW - Interface Routine to FIL$INIWCB'
; 1628 1
; 1629 1      ROUTINE INIT_WINDOW (CHANNEL, BUFFER_DESC) : SYS_CMKRNL =
; 1630 1
; 1631 1      !+
; 1632 1      ! Functional Description:
; 1633 1      !
; 1634 1      !     This routine acts as a jacket to the routine called FIL$INIWCB, which
; 1635 1      !     allocates a window control block in the absence of a complete file
; 1636 1      !     system.
; 1637 1      !
; 1638 1      ! Calling Sequence:
; 1639 1      !
; 1640 1      !     $CMKRNL (INIT_WINDOW, ARGUMENT_LIST)
; 1641 1      !
; 1642 1      ! Formal Parameters:
; 1643 1      !
; 1644 1      !     CHANNEL - Address of channel on which image file is opened
; 1645 1      !
; 1646 1      !     BUFFER_DESC - Address of descriptor of retrieval pointer buffer
; 1647 1      !-
; 1648 1
; 1649 2      BEGIN
; 1650 2
; 1651 2      BUILTIN
; 1652 2          ADAWI;                                ! Interlocked add
; 1653 2
; 1654 2      EXTERNAL REGISTER
; 1655 2          PCB = 4 : REF $BBLOCK;                ! We enter this procedure with the P
; 1656 2          ! address contained in R4
; 1657 2
; 1658 2      BIND
; 1659 2          JIB      = .PCB [PCB$L_JIB] : $BBLOCK,
; 1660 2          BUF_DSC = .BUFFER_DESC      : VECTOR [2];
; 1661 2
; 1662 2      LOCAL
; 1663 2          STATUS,
; 1664 2          CCB : REF $BBLOCK,
; 1665 2          WCB;
; 1666 2
; 1667 2      STATUS = IOC$VERIFYCHAN (.CHANNEL; CCB);
; 1668 2      IF NOT .STATUS THEN RETURN .STATUS;
; 1669 2      STATUS = FIL$INIWCB (.BUF_DSC [0], .BUF_DSC [1], .CCB [CCB$L_UCB]; WCB);
; 1670 2      IF NOT .STATUS THEN RETURN .STATUS;
; 1671 2      CCB [CCB$L_WIND] = .WCB;
; 1672 2      CCB [CCB$L_AMOD] = PSL$C_USER + 1;
; 1673 2      ADAWI ( %REF(-1), JIB[JIB$W_FILCNT] );
; 1674 2      RETURN SS$ _NORMAL
; 1675 1      END;

```

```

00EC 0000 INIT_WINDOW:
                                .WORD   Save R2,R3,R5,R6,R7           ; 1629
7C  A4  DO 0002                MOVL    124(PCB), R7                  ; 1658

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 33

```

X-14          INIT_WINDOW - Interface Routine to FIL$INIWCB   10-Apr-1989 10:48:52   _$25
      08 AC D0 00006      MOVL   BUFFER_DESC, R6                      ; 1659
      04 AC D0 0000A      MOVL   CHANNEL, R0                          ; 1666
00000000G    00 16 0000E      JSB   IOC$VERIFYCHAN                      ;
      51 D0 00014      MOVL   R1, R5                                  ;
      50 E9 00017      BLBC   STATUS, 1$                              ; 1667
      65 D0 0001A      MOVL   (CCB), R3                              ; 1668
      66 7D 0001D      MOVQ   (R6), R1                                ;
00000000G    00 16 00020      JSB   FIL$INIWCB                          ;
      50 E9 00026      BLBC   STATUS, 1$                              ; 1669
      52 D0 00029      MOVL   WCB, 4(CCB)                            ; 1670
      04 90 0002D      MOVB   #4, 9(CCB)                             ; 1671
      FFFF 8F 58 00031      ADAWI  #-1, 48(R7)                       ; 1672
      01 D0 00037      MOVL   #1, R0                                  ; 1673
      04 0003A 1$:      RET                                           ;

```

; Routine Size: 59 bytes, Routine Base: EXEC\$PAGED\_CODE + 0535

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SY\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 34

```

X-14          IMG$GET_HEADER - Get Parameters from Image Head 10-Apr-1989 10:48:52    _$25
; 1677 1      %SBTTL 'IMG$GET_HEADER - Get Parameters from Image Header'
; 1678 1
; 1679 1      GLOBAL ROUTINE IMG$GET_HEADER (ICB_ADDRESS) =
; 1680 1
; 1681 1      !+
; 1682 1      ! Functional Description:
; 1683 1      !
; 1684 1      !   This routine is a jacket routine that handler two kinds of image header
; 1685 1      !   If an image was installed header resident, then there is no need to dec
; 1686 1      !   the header or verify its contents. Other images must have their headers
; 1687 1      !   read into memory before the decode and verification can occur.
; 1688 1      !
; 1689 1      ! Calling Sequence:
; 1690 1      !
; 1691 1      !   IMG$GET_HEADER (ICB_ADDRESS)
; 1692 1      !
; 1693 1      ! Formal Parameter:
; 1694 1      !
; 1695 1      !   ICB_ADDRESS - Address of image control block that describes the image
; 1696 1      !   that is currently being activated.
; 1697 1      !-
; 1698 1
; 1699 2      BEGIN
; 1700 2
; 1701 2      BIND
; 1702 2          ICB_ADR = .ICB_ADDRESS           : $BBLOCK,
; 1703 2          KFE     = .ICB_ADR [ICB$L_KFE]   : $BBLOCK,
; 1704 2          IHD_CTX = .ICB_ADR [ICB$L_CONTEXT] : $BBLOCK;
; 1705 2
; 1706 2      LOCAL
; 1707 2          STATUS;
; 1708 2
; 1709 2      IF
; 1710 3          BEGIN
; 1711 3              IF KFE EQL 0
; 1712 3                  THEN TRUE
; 1713 3                  ELSE NOT .KFE [KFE$V_HDRRES]
; 1714 3                  END
; 1715 2          THEN
; 1716 3              BEGIN
; 1717 3
; 1718 3                  STATUS = IMG$DECODE_IHD (
; 1719 3                      .ICB_ADR [ICB$W_CHAN],
; 1720 3                      .IHD_CTX [CTX_L_BUFFER],
; 1721 3                      .IHD_CTX [CTX_L_IHDBUF],
; 1722 3                      IHD_CTX [CTX_L_VBN],
; 1723 3                      IHD_CTX [CTX_W_ISD_OFFSET],
; 1724 3                      IHD_CTX [CTX_W_GENERATION],
; 1725 3                      IHD_CTX [CTX_W_ALIAS]);
; 1726 3
; 1727 3                  IF NOT .STATUS THEN RETURN .STATUS;
; 1728 3
; 1729 3                  ! Images that contain an alias of either IHD$C_RSX or IHD$C_BPA are not
; 1730 3                  ! native images produced by the linker and, as a result, do not require
; 1731 3                  ! a check against the system version field.
; 1732 3
; 1733 3              IF

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 35

```

X-14          IMG$GET_HEADER - Get Parameters from Image Head 10-Apr-1989 10:48:52    _$25
; 1734 4          (.IHD_CTX [CTX_W_ALIAS] NEQ IHD$C_RSX)
; 1735 3          AND
; 1736 4          (.IHD_CTX [CTX_W_ALIAS] NEQ IHD$C_BPA)
; 1737 3          AND
; 1738 4          (.IHD_CTX [CTX_W_ALIAS] NEQ IHD$C_ALIAS)
; 1739 3          THEN
; 1740 4          BEGIN
; 1741 4
; 1742 4          BIND
; 1743 4          IHD = .IHD_CTX [CTX_L_IHDBUF]          : $BLOCK,
; 1744 4          MINORID_DIGIT = IHD [IHD$W_MINORID] : VECTOR [2,BYTE],
; 1745 4          IHA = IHD + .IHD [IHD$W_ACTIVOFF]   : VECTOR [5];
; 1746 4
; 1747 4          ! MINORID_DIGIT [0] maps the tens digit.
; 1748 4          ! MINORID_DIGIT [1] maps the units digit.
; 1749 4
; 1750 4          LITERAL
; 1751 4          MINOR_ID_TENS = IHD$K_MINORID AND %X'FF',
; 1752 4          MINOR_ID_ONES = IHD$K_MINORID ^ -8;
; 1753 4
; 1754 4          ! The major ID in the image header must be identically equal to
; 1755 4          ! the constant IHD$K_MAJORID. The minor ID in the image header
; 1756 4          ! must be LEQU the constant IHD$K_MINORID. Both IDs are stored
; 1757 4          ! as ASCII strings.
; 1758 4
; 1759 5          IF (.IHD [IHD$W_MAJORID] NEQU IHD$K_MAJORID)
; 1760 4          THEN RETURN SS$_BADIMGHDR;
; 1761 4
; 1762 5          IF (
; 1763 6              (.MINORID_DIGIT [0] GTRU MINOR_ID_TENS)
; 1764 5              OR
; 1765 6              (
; 1766 7                  (.MINORID_DIGIT [0] EQLU MINOR_ID_TENS)
; 1767 6                  AND
; 1768 7                  (.MINORID_DIGIT [1] GTRU MINOR_ID_ONES)
; 1769 6              )
; 1770 5          )
; 1771 4          THEN RETURN SS$_BADIMGHDR;
; 1772 4
; 1773 4          ! Check match control data for shareable images that are being
; 1774 4          ! activated because of global ISD references.
; 1775 4
; 1776 4          IF .ICB_ADR [ICB$B_ACT_CODE] EQLU ICB$K_GLOBAL_IMAGE_SECTION
; 1777 4          THEN
; 1778 5              BEGIN
; 1779 5                  STATUS = CHECK_MATCH_CONTROL (ICB_ADR, IHD);
; 1780 5                  IF NOT .STATUS THEN RETURN .STATUS;
; 1781 4                  END;
; 1782 4
; 1783 4          ! We must record whether we are activating a shareable image with an
; 1784 4          ! initialization section.
; 1785 4
; 1786 4          IF .IHD [IHD$V_INISHR]
; 1787 4          THEN
; 1788 5              BEGIN
; 1789 5                  ICB_ADR [ICB$L_INITIALIZE] = .IHA [4];
; 1790 5                  ICB_ADR [ICB$V_INITIALIZE] = TRUE;

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 36

```

X-14          IMG$GET_HEADER - Get Parameters from Image Head 10-Apr-1989 10:48:52      _$25
; 1791 5          IAC$GL_IMAGCTX [IMAGCTX$V_INITIALIZE] = TRUE;
; 1792 4          END;
; 1793 4
; 1794 4          ! EXE$CHECK_VERSION is the routine used by the various loading
; 1795 4          ! mechanisms to check for version mismatches.
; 1796 4          ! Status values returned are NORMAL, BADIMGHDR, and SYSVERDIF.
; 1797 4
; 1798 4          STATUS = EXE$CHECK_VERSION(IHD);
; 1799 4
; 1800 4          !
; 1801 4          ! The following piece of code here for historical purpose only: prior to
; 1802 4          ! VMS V5.1, system version mismatch was treated as an alternate success
; 1803 4          ! and the image is activated with privileges removed. This is causing more
; 1804 4          ! harm than good (i.e. it breaks the version checking scheme), so it is
; 1805 4          ! removed (i.e. image activation will fail if there is a version mismatch).
; 1806 4          !
; 1807 4          !     IF (.STATUS AND STS$M_CODE) EQL (SS$ SYSVERDIF AND STS$M_CODE)
; 1808 4          !     THEN
; 1809 4          !     BEGIN
; 1810 4          !     OWN_STORAGE [REMOVE_PRIVILEGE] = TRUE;
; 1811 4          !     STATUS = SS$ SYSVERDIF;
; 1812 4          !     END;
; 1813 4          !
; 1814 4
; 1815 3          END;
; 1816 3          END
; 1817 2          ELSE          ! Header is already resident
; 1818 3          BEGIN
; 1819 3
; 1820 3          BIND
; 1821 3          IHD = .KFE [KFE$L_IMGHDR]          : $BBLOCK,
; 1822 3          KFRH = IHD - KFRH$K_LENGTH          : $BBLOCK,
; 1823 3          IHA = IHD + .IHD [IHD$W_ACTIVOFF] : VECTOR [5];
; 1824 3
; 1825 3          ICB_ADR [ICB$V_RES_HEADER] = TRUE; ! Indicate that header is already in
; 1826 3          ICB_ADR [ICB$L_IHD] = IHD;          ! Remember the address of the reside
; 1827 3          IHD_CTX [CTX_L_IHDBUF] = 0;          ! Clear this so it won't get used by
; 1828 3          IHD_CTX [CTX_W_ISD_OFFSET] = .IHD [IHD$W_SIZE];
; 1829 3          IHD_CTX [CTX_W_GENERATION] = .KFRH [KFRH$B_HDRVER];
; 1830 3          IHD_CTX [CTX_W_ALIAS] = .KFRH [KFRH$W_ALIAS];
; 1831 3
; 1832 3          IF .ICB_ADR [ICB$B_ACT_CODE] EQLU ICB$K_GLOBAL_IMAGE_SECTION
; 1833 3          THEN
; 1834 4          BEGIN
; 1835 4          STATUS = CHECK_MATCH_CONTROL (ICB_ADR, IHD);
; 1836 4          IF NOT .STATUS THEN RETURN .STATUS;
; 1837 4          END;
; 1838 3
; 1839 3          ! We must record whether we are activating a shareable image with an
; 1840 3          ! initialization section.
; 1841 3
; 1842 3          IF .IHD [IHD$V_INISHR]
; 1843 3          THEN
; 1844 4          BEGIN
; 1845 4          ICB_ADR [ICB$L_INITIALIZE] = .IHA [4];
; 1846 4          ICB_ADR [ICB$V_INITIALIZE] = TRUE;
; 1847 4          IAC$GL_IMAGCTX [IMAGCTX$V_INITIALIZE] = TRUE;

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 37

```
X-14      IMG$GET_HEADER - Get Parameters from Image Head 10-Apr-1989 10:48:52      _$25
; 1848 3          END;
; 1849 3
; 1850 3          STATUS = SS$ _NORMAL;
; 1851 2          END;
; 1852 2          RETURN .STATUS;
; 1853 2
; 1854 1          END;
```

```

003C 0000      .ENTRY  IMG$GET_HEADER, Save R2,R3,R4,R5      ; 1679
04 AC D0 0002  MOVL   ICB_ADDRESS, R4                        ; 1702
54 A4 D0 0006  MOVL   84(R4), R0                               ; 1703
58 A4 D0 000A  MOVL   88(R4), R2                               ; 1704
    50 D5 000E  TSTL   R0                                       ; 1711
    08 13 0010  BEQL   1$                                       ;
    04 E1 0012  BBC    #4, 16(R0), 1$                          ; 1713
0089 31 0017  BRW    9$                                       ;
OE A2 9F 001A 1$: PUSHAB 14(R2)                               ; 1725
10 A2 9F 001D  PUSHAB 16(R2)                               ; 1724
OC A2 9F 0020  PUSHAB 12(R2)                               ; 1723
08 A2 9F 0023  PUSHAB 8(R2)                                ; 1722
    62 7D 0026  MOVQ   (R2), -(SP)                            ; 1718
OE A4 3C 0029  MOVZWL 14(R4), -(SP)                          ; 1719
    07 FB 002D  CALLS  #7, IMG$DECODE_IHD                    ;
    50 E9 0034  BLBC  STATUS, 6$                               ; 1727
OE A2 32 0037  CVTWL  14(R2), R1                             ; 1734
    08 13 003B  BEQL   2$                                       ;
    51 B1 003D  CMPW  R1, #1                                   ; 1736
    03 13 0040  BEQL   2$                                       ;
    51 B1 0042  CMPW  R1, #2                                   ; 1738
    01 12 0045 2$: BNEQ  3$                                       ;
    04 0047  RET                                       ;
04 A2 D0 0048 3$: MOVL   4(R2), R3                             ; 1742
OE A3 9E 004C  MOVAB  14(R3), R1                             ; 1744
02 A3 3C 0050  MOVZWL 2(R3), R5                             ; 1745
    55 C1 0054  ADDL3  R5, R3, R2                             ;
OC A3 B1 0058  CMPW  12(R3), #12848                          ; 1759
    0D 12 005E  BNEQ  4$                                       ;
    61 91 0060  CMPB  (R1), #48                               ; 1763
    08 1A 0063  BGTRU 4$                                       ;
    0B 12 0065  BNEQ  5$                                       ; 1766
01 A1 91 0067  CMPB  1(R1), #53                              ; 1768
    05 1B 006B  BLEQU 5$                                       ;
44 8F 9A 006D 4$: MOVZBL #68, R0                               ; 1771
    04 0071  RET                                       ;
OD A4 91 0072 5$: CMPB  13(R4), #3                             ; 1776
    0C 12 0076  BNEQ  7$                                       ;
    53 DD 0078  PUSHL R3                                       ; 1779
    54 DD 007A  PUSHL R4                                       ;
    02 FB 007C  CALLS  #2, CHECK_MATCH_CONTROL                ;
    50 E9 0081 6$: BLBC  STATUS, 12$                          ; 1780
    06 E1 0084 7$: BBC    #6, 32(R3), 8$                      ; 1786
10 A2 D0 0089  MOVL   16(R2), 96(R4)                          ; 1789
    20 88 008E  BISB2 #32, 16(R4)                             ; 1790
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 38

```

X-14      IMG$GET_HEADER - Get Parameters from Image Head 10-Apr-1989 10:48:52      _$25
          02 88 00092      BISB2      #2, IAC$GL_IMAGCTX+2      ; 1791
          53 DD 00099 8$:  PUSHL      R3      ; 1798
          01 FB 0009B      CALLS      #1, EXE$CHECK_VERSION      ;
          04 000A2      RET      ;
1C      A0 D0 000A3 9$:  MOVL      28(R0), R3      ; 1821
F4      A3 9E 000A7      MOVAB      -12(R3), R1      ; 1822
02      A3 3C 000AB      MOVZWL     2(R3), R0      ; 1823
          50 C1 000AF      ADDL3     R0, R3, R5      ;
          08 88 000B3      BISB2     #8, 16(R4)      ; 1825
          53 D0 000B7      MOVL      R3, 80(R4)      ; 1826
04      A2 D4 000BB      CLRL     4(R2)      ; 1827
          63 B0 000BE      MOVW     (R3), 12(R2)      ; 1828
0B      A1 9B 000C2      MOVZBW   11(R1), 16(R2)      ; 1829
04      A1 B0 000C7      MOVW     4(R1), 14(R2)      ; 1830
0D      A4 91 000CC      CMPB     13(R4), #3      ; 1832
          0C 12 000D0      BNEQ     10$      ;
          53 DD 000D2      PUSHL     R3      ; 1835
          54 DD 000D4      PUSHL     R4      ;
          02 FB 000D6      CALLS     #2, CHECK_MATCH_CONTROL      ;
          50 E9 000DB      BLBC     STATUS, 12$      ; 1836
          06 E1 000DE 10$:  BBC      #6, 32(R3), 11$      ; 1842
10      A5 D0 000E3      MOVL     16(R5), 96(R4)      ; 1845
          20 88 000E8      BISB2     #32, 16(R4)      ; 1846
          02 88 000EC      BISB2     #2, IAC$GL_IMAGCTX+2      ; 1847
          01 D0 000F3 11$:  MOVL     #1, STATUS      ; 1850
          04 000F6 12$:  RET      ; 1852

```

; Routine Size: 247 bytes, Routine Base: EXEC\$PAGED\_CODE + 0570

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 39

```

X-14          CHECK_MATCH_CONTROL - Check Match Control Ident 10-Apr-1989 10:48:52    _$25
; 1856 1      %SBTTL 'CHECK_MATCH_CONTROL - Check Match Control Identification'
; 1857 1
; 1858 1      ROUTINE CHECK_MATCH_CONTROL (ICB, IHD ) =
; 1859 1
; 1860 1      !+
; 1861 1      ! Functional Description:
; 1862 1      !
; 1863 1      !   This routine checks that the match control information in the image hea
; 1864 1      !   is consistent with the match control information that is actually being
; 1865 1      !   requested. The check is made in three steps.
; 1866 1      !
; 1867 1      !   1. The match control flag that is being requested must be at least as
; 1868 1      !   restrictive as the match control flag in the image header.
; 1869 1      !
; 1870 1      !   2. If the resultant match control is MATCH ALWAYS, no further checks a
; 1871 1      !   made. If the resultant check is MATCH NEVER, ??????.
; 1872 1      !
; 1873 1      !   3. If the match control is either MATCH EQUAL or MATCH LEQUAL, two
; 1874 1      !   further checks are made.
; 1875 1      !
; 1876 1      !       a. The two major IDs (one in ICB and one in IHD) must be equal.
; 1877 1      !
; 1878 1      !       b. The two minor IDs must be related according to the match contrc
; 1879 1      !           In the case of MATCH EQUAL, they must be equal. In the case of
; 1880 1      !           MATCH LEQUAL, the requested minor ID (located in the ICB) must
; 1881 1      !           LEQU the minor ID in the image header of the image being
; 1882 1      !           activated.
; 1883 1      !
; 1884 1      ! Calling Sequence:
; 1885 1      !
; 1886 1      !   CHECK_MATCH_CONTROL (ICB, IHD )
; 1887 1      !
; 1888 1      ! Formal Parameter:
; 1889 1      !
; 1890 1      !   ICB - Address of image control block that describes the image
; 1891 1      !   that is currently being activated.
; 1892 1      !
; 1893 1      !   IHD - Address of image header of image being activated.
; 1894 1      !-
; 1895 1
; 1896 2      BEGIN
; 1897 2
; 1898 2      MACRO
; 1899 2          IHD_V_MINOR_ID = $BYTEOFFSET (IHD$$_IDENT), 0, 24, 0 %,
; 1900 2          IHD_V_MAJOR_ID = $BYTEOFFSET (IHD$$_IDENT), 24, 8, 0 %;
; 1901 2
; 1902 2      MAP
; 1903 2          ICB : REF $BLOCK,
; 1904 2          IHD : REF $BLOCK;
; 1905 2
; 1906 2      CASE .IHD [IHD$$_MATCHCTL]
; 1907 2      FROM ISD$$_MATALL TO ISD$$_MATNEV OF
; 1908 2      SET
; 1909 2
; 1910 2      [ISD$$_MATALL]:
; 1911 2
; 1912 2          SS$$_NORMAL;                                ! Do nothing

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 40

X-14 CHECK\_MATCH\_CONTROL - Check Match Control Ident 10-Apr-1989 10:48:52 \_\$25

```
; 1913 2
; 1914 2 [ISD$K_MATEQU]:
; 1915 2
; 1916 2 IF
; 1917 3 (.ICB [ICB$V_MATCH_CONTROL] EQLU ISD$K_MATALL)
; 1918 2 OR
; 1919 3 (.ICB [ICB$V_MATCH_CONTROL] EQLU ISD$K_MATLEQ)
; 1920 2 THEN RETURN SS$_SHRIDMISMAT;
; 1921 2
; 1922 2 [ISD$K_MATLEQ]:
; 1923 2
; 1924 2 IF .ICB [ICB$V_MATCH_CONTROL] EQLU ISD$K_MATALL
; 1925 2 THEN RETURN SS$_SHRIDMISMAT;
; 1926 2
; 1927 2 [ISD$K_MATNEV]:
; 1928 2
; 1929 2 RETURN SS$_SHRIDMISMAT;
; 1930 2
; 1931 2 TES;
; 1932 2
; 1933 2 CASE .ICB [ICB$V_MATCH_CONTROL]
; 1934 2 FROM ISD$K_MATALL TO ISD$K_MATNEV OF
; 1935 2 SET
; 1936 2
; 1937 2 [ISD$K_MATALL]:
; 1938 2
; 1939 2 RETURN SS$_NORMAL;
; 1940 2
; 1941 2 [ISD$K_MATEQU, ISD$K_MATNEV]:
; 1942 2
; 1943 3 BEGIN
; 1944 3
; 1945 3 IF .ICB [ICB$V_MAJOR_ID] NEQU .IHD [IHD_V_MAJOR_ID]
; 1946 3 THEN RETURN SS$_SHRIDMISMAT;
; 1947 3
; 1948 3 IF .ICB [ICB$V_MINOR_ID] EQLU .IHD [IHD_V_MINOR_ID]
; 1949 3 THEN RETURN SS$_NORMAL
; 1950 3 ELSE RETURN SS$_SHRIDMISMAT;
; 1951 3
; 1952 2 END;
; 1953 2
; 1954 2 [ISD$K_MATLEQ]:
; 1955 2
; 1956 3 BEGIN
; 1957 3
; 1958 3 IF .ICB [ICB$V_MAJOR_ID] NEQU .IHD [IHD_V_MAJOR_ID]
; 1959 3 THEN RETURN SS$_SHRIDMISMAT;
; 1960 3
; 1961 3 IF .ICB [ICB$V_MINOR_ID] LEQU .IHD [IHD_V_MINOR_ID]
; 1962 3 THEN RETURN SS$_NORMAL
; 1963 3 ELSE RETURN SS$_SHRIDMISMAT;
; 1964 3
; 1965 2 END;
; 1966 2
; 1967 2 TES;
; 1968 2
; 1969 1 END;
```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSSIMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 41  
X-14

CHECK\_MATCH\_CONTROL - Check Match Control Ident 10-Apr-1989 10:48:52    \_\$25

```

000C 00000 CHECK_MATCH_CONTROL:
      .WORD   Save R2,R3                ; 1858
08  AC  D0 00002      MOVL   IHD, R1                ; 1906
      00  EF 00006      EXTZV  #0, #3, 35(R1), R2      ;
      52  CF 0000C      CASEL  R2, #0, #3            ;
0026      00010 1$:   .WORD   5$-1$, -              ;
      2$-1$, -              ;
      3$-1$, -              ;
      10$-1$                ;
      1C  11 00018      BRB    5$                    ; 1912
04  AC  D0 0001A 2$:  MOVL   ICB, R0                ; 1917
40  A0  93 0001E      BITB   64(R0), #7            ;
      5A  13 00022      BEQL   10$                  ;
      00  ED 00024      CMPZV  #0, #3, 64(R0), #2      ; 1919
      08  11 0002A      BRB    4$                    ;
04  AC  D0 0002C 3$:  MOVL   ICB, R0                ; 1924
40  A0  93 00030      BITB   64(R0), #7            ;
      48  13 00034 4$:  BEQL   10$                  ;
04  AC  D0 00036 5$:  MOVL   ICB, R0                ; 1933
      00  EF 0003A      EXTZV  #0, #3, 64(R0), R2      ;
      52  CF 00040      CASEL  R2, #0, #3            ;
0036      00044 6$:   .WORD   9$-6$, -              ;
      7$-6$, -              ;
      8$-6$, -              ;
      7$-6$                ;
      2C  11 0004C      BRB    9$                    ; 1939
47  A0  91 0004E 7$:  CMPB   71(R0), 39(R1)          ; 1945
      29  12 00053      BNEQ   10$                  ;
      00  EF 00055      EXTZV  #0, #24, 36(R1), R3      ; 1948
      00  ED 0005B      CMPZV  #0, #24, 68(R0), R3      ;
      1B  12 00061      BNEQ   10$                  ;
      15  11 00063      BRB    9$                    ; 1949
47  A0  91 00065 8$:  CMPB   71(R0), 39(R1)          ; 1958
      12  12 0006A      BNEQ   10$                  ;
      00  EF 0006C      EXTZV  #0, #24, 36(R1), R3      ; 1961
      00  ED 00072      CMPZV  #0, #24, 68(R0), R3      ;
      04  1A 00078      BGTRU  10$                  ;
      01  D0 0007A 9$:  MOVL   #1, R0                ; 1962
      04  0007D      RET
20BC  8F  3C 0007E 10$: MOVZWL #8380, R0
      04  00083      RET

```

; Routine Size: 132 bytes,    Routine Base: EXEC\$PAGED\_CODE + 0667

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 42

```

X-14      END_PROCESSING - Final Steps of Image Activatio 10-Apr-1989 10:48:52    _$25
; 1971 1    %SBTTL 'END_PROCESSING - Final Steps of Image Activation'
; 1972 1
; 1973 1    ROUTINE END_PROCESSING (ICB_ADDRESS) =
; 1974 1
; 1975 1    !+
; 1976 1    ! Functional Description:
; 1977 1    !
; 1978 1    !   This routine performs the steps involved in image activation after all
; 1979 1    !   of the ISDs have been processed. (Some of the details are actually
; 1980 1    !   performed by the kernel mode routine SET_CONTROL_REGION.) The operation
; 1981 1    !   that need to be completed include
; 1982 1    !
; 1983 1    !   o mapping the image I/O segment
; 1984 1    !
; 1985 1    !   o mapping the user stack and setting the initial value of USP
; 1986 1    !
; 1987 1    !   o setting up the process privilege mask
; 1988 1    !
; 1989 1    !   o optionally setting up image accounting information
; 1990 1    !
; 1991 1    !   o bumping the use count in any KFEs referenced by this activation
; 1992 1    !
; 1993 1    !   o bumping the reference count in any shared WCBs referenced by this
; 1994 1    !   activation
; 1995 1    !
; 1996 1    !   o returning image parameters to the caller
; 1997 1    !
; 1998 1    ! Calling Sequence:
; 1999 1    !
; 2000 1    !   END_PROCESSING (ICB_ADDRESS)
; 2001 1    !-
; 2002 1
; 2003 2    BEGIN
; 2004 2
; 2005 2    LOCAL
; 2006 2        IMAGE_IO_PAGE_COUNT,
; 2007 2        STACK_BASE,
; 2008 2        CRETVA_RANGE      : VECTOR [2],
; 2009 2        ARG_LIST          : VECTOR [4] INITIAL (
; 2010 2            3,                ! Argument count
; 2011 2            CRETVA_RANGE,    ! INADR
; 2012 2            0,                ! Null retradr
; 2013 2            EXEC_PROT),      ! Access mode and protection
; 2014 2        IMGIO_SEG_DESC    : VECTOR [2] INITIAL (0,0),
; 2015 2        STATUS;
; 2016 2
; 2017 2    BIND
; 2018 2        ICB = .ICB_ADDRESS      : $BBLOCK,
; 2019 2        KFE = .ICB [ICB$L_KFE] : $BBLOCK,
; 2020 2        PHD = .CTL$GL_PHD     : $BBLOCK,
; 2021 2        STACK_ARRAY = CRETVA_RANGE : VECTOR;
; 2022 2
; 2023 2    ! The location of the image header depends on whether the image was installe
; 2024 2    ! with its header resident.
; 2025 2
; 2026 2    BIND
; 2027 2        IHD =

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 43

```

X-14      END_PROCESSING - Final Steps of Image Activatio 10-Apr-1989 10:48:52      _$25
; 2028 3      (IF .ICB [ICB$L_IHD] EQL 0
; 2029 3      THEN
; 2030 4          BEGIN
; 2031 4
; 2032 4          BIND
; 2033 4              CTX = .ICB [ICB$L_CONTEXT]      : $BBLOCK;
; 2034 4
; 2035 4              .CTX [CTX_L_IHDBUF]
; 2036 4
; 2037 4          END
; 2038 3      ELSE
; 2039 2          .ICB [ICB$L_IHD])      : $BBLOCK;
; 2040 2
; 2041 2      ! If this is the activation of a main program, then the image I/O segment an
; 2042 2      ! user stack must be mapped.
; 2043 2
; 2044 2      IF .OWN_STORAGE [MAIN_PROGRAM]
; 2045 2      THEN
; 2046 3          BEGIN
; 2047 4              IMAGE_IO_PAGE_COUNT = (
; 2048 4                  IF .IHD [IHD$W_IMGIOCNT] NEQ 0
; 2049 4                  THEN
; 2050 4                      .IHD [IHD$W_IMGIOCNT]
; 2051 4                  ELSE
; 2052 3                      .SGN$GW_IMGIOCNT);
; 2053 3
; 2054 3          IF .IMAGE_IO_PAGE_COUNT GTR .SGN$GW_IMGIOCNT
; 2055 3          THEN
; 2056 4              BEGIN
; 2057 4                  IF .IHD [IHD$V_POIMAGE]
; 2058 4                  THEN
; 2059 5                      BEGIN
; 2060 5                          CRETVA_RANGE [0] = .PHD [PHD$L_FREPOVA];
; 2061 5                          CRETVA_RANGE [1] =
; 2062 5                              .CRETVA_RANGE [0]
; 2063 6                              + (.IMAGE_IO_PAGE_COUNT * BYTES_PER_PAGE)
; 2064 5                              - 1;
; 2065 5                      END
; 2066 4                  ELSE
; 2067 5                      BEGIN
; 2068 5                          CRETVA_RANGE [1] = .CTL$GL_CTLBASVA - 1;
; 2069 5                          CRETVA_RANGE [0] =
; 2070 5                              .CRETVA_RANGE [1]
; 2071 5                              - (.IMAGE_IO_PAGE_COUNT * BYTES_PER_PAGE) + 1;
; 2072 4                      END;
; 2073 4
; 2074 4                  ! Create the image I/O segment using the internal routine that allow
; 2075 4                  ! a mixed-mode protection mask to be specified.
; 2076 4
; P 2077 4                  STATUS = $CMKRNL (
; P 2078 4                      ROUTIN = MMG$CRETVA,
; 2079 4                      ARGLST = ARG_LIST);
; 2080 4                  IF NOT .STATUS
; 2081 4                  THEN RETURN .STATUS;
; 2082 4
; 2083 4                  IMGIO_SEG_DESC [0] = .IMAGE_IO_PAGE_COUNT * BYTES_PER_PAGE;
; 2084 4                  IMGIO_SEG_DESC [1] = .CRETVA_RANGE [0];

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSSIMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 45

X-14                   END\_PROCESSING - Final Steps of Image Activatio 10-Apr-1989 10:48:52     \_\$25

```

; 2142 3
; 2143 3      ! Insure that buffer is still writable by caller
; 2144 3
; 2145 3      IF NOT PROBEW (%REF(0), %REF(RETURN_BUFFER_SIZE), .OWN_STORAGE [BUFFER_A
; 2146 3      THEN RETURN SS$_ACCVIO;
; 2147 3
; 2148 3      ! Insure that image header information will fit into a single page
; 2149 3
; 2150 4      IF (
; 2151 4          12 +                ! Three longword header
; 2152 4          .IHD [IHD$W_SIZE]    ! Image header less ISDs
; 2153 4          )
; 2154 3      LEQU RETURN_BUFFER_SIZE
; 2155 3      THEN
; 2156 4          BEGIN
; 2157 4
; 2158 4          ! Fill in the three pointers
; 2159 4
; 2160 4          BUFHDR [0] = BUFFER [12];      ! Image header starts right
; 2161 4          BUFHDR [1] = 0;                ! 0 until we see if IFD fits
; 2162 4          BUFHDR [2] = 0;                ! Third longword is unused o
; 2163 4
; 2164 4          ! Copy the image header
; 2165 4
; 2166 4          CH$MOVE (.IHD [IHD$W_SIZE], IHD, BUFFER [12]);
; 2167 4
; 2168 4          ! Any transfer address array elements that are not located in the
; 2169 4          ! permanent portion of P1 space or in system space must be relocated
; 2170 4          ! by an amount equal to the base address of the image just mapped.
; 2171 4
; 2172 4          IF
; 2173 5              (.OWN_STORAGE [TRANSFER_ARRAY_BIAS] NEQ 0)
; 2174 5              AND
; 2175 5              (NOT .IHD [IHD$V_LNKNOTFR])
; 2176 4          THEN
; 2177 5              BEGIN
; 2178 5
; 2179 5              BIND
; 2180 5                  IHA = .BUFHDR [0] + .IHD [IHD$W_ACTIVOFF]      : VECTOR [3]
; 2181 5
; 2182 5              LOCAL
; 2183 5                  I;
; 2184 5
; 2185 5              INCR I FROM 0 TO 2 DO
; 2186 5                  IF .IHA [.I] LSSU .CTL$GL_CTLBASVA
; 2187 5                  THEN
; 2188 5                      IHA [.I] = .IHA [.I] + .OWN_STORAGE [TRANSFER_ARRAY_BIAS]
; 2189 4              END;
; 2190 4
; 2191 4          ! See if we can also fit in the IFD
; 2192 4
; 2193 5          IF (
; 2194 5              12 +                ! Three longword header
; 2195 5              .IHD [IHD$W_SIZE] +    ! Image header less ISDs
; 2196 5              IFD$K_LENGTH +        ! Fixed portion of IFD
; 2197 5              .NAM [NAM$B_RSL] +    ! Length of image file name
; 2198 5              1 )                ! Count byte in ASCII string

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSSIMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 46

X-14                    END\_PROCESSING - Final Steps of Image Activatio 10-Apr-1989 10:48:52     \_\$25

```

; 2199 4                    LEQU RETURN_BUFFER_SIZE
; 2200 4                    THEN
; 2201 5                    BEGIN
; 2202 5                    IFD = BUFFER [12 + .IHD [IHD$W_SIZE]];           ! IFD follows the im
; 2203 5                    BUFHDR [1] = .IFD;                                 ! Store its address
; 2204 5
; 2205 5                    ! Fill in the fixed portion of the IFD
; 2206 5
; 2207 5                    IFD [IFD$W_SIZE] = IFD$K_LENGTH + .NAM [NAM$B_RSL];
; 2208 5                    IFD [IFD$W_FILNAMOFF] = IFD$K_LENGTH;
; 2209 5                    IFD [IFD$W_CHAN] = .ICB [ICB$W_CHAN];
; 2210 5
; 2211 5                    IFD [IFD$W_CMCHAN] = .OWN_STORAGE [OTHER_CHANNEL];
; 2212 5                    IFD [IFD$L_CMKFIADR] = .OWN_STORAGE [OTHER_KFE_ADDRESS];
; 2213 5
; 2214 5                    IFD [IFD$W_FLAGS] = .IAC$GL_IMAGCTX;
; 2215 5
; 2216 5                    FILE_NAME = .IFD + IFD$K_LENGTH;
; 2217 5
; 2218 6                    BEGIN
; 2219 6
; 2220 6                    BIND
; 2221 6                    FILE_NAME_DESC = IFD [IFD$Q_CURPROG]         : $BLOCK;
; 2222 6
; 2223 6                    FILE_NAME_DESC [DSC$W_LENGTH] = .NAM [NAM$B_RSL];
; 2224 6                    FILE_NAME_DESC [DSC$A_POINTER] = .FILE_NAME + 1;
; 2225 6
; 2226 5                    END;
; 2227 5
; 2228 5                    FILE_NAME [0] = .NAM [NAM$B_RSL];
; 2229 5                    CH$MOVE (
; 2230 5                    .NAM [NAM$B_RSL],
; 2231 5                    RESULT_NAME,
; 2232 5                    FILE_NAME [1]);
; 2233 5
; 2234 5                    END
; 2235 5
; 2236 4                    ELSE
; 2237 4
; 2238 4                    ! Fill the IFD portion with zeros
; 2239 4
; 2240 4                    CH$FILL (0, RETURN_BUFFER_SIZE - 12 - .IHD [IHD$W_SIZE],
; 2241 4                    BUFFER [12 + .IHD [IHD$W_SIZE]] );
; 2242 4                    END
; 2243 4
; 2244 3                    ELSE
; 2245 3
; 2246 3                    !!! FOR LACK OF ANYTHING BETTER TO DO IN THIS CASE, I WILL FILL
; 2247 3                    !!! THE ENTIRE 512-BYTE BUFFER WITH ZEROS
; 2248 3
; 2249 3                    CH$FILL (0, RETURN_BUFFER_SIZE, .OWN_STORAGE [BUFFER_ADDRESS]);
; 2250 3
; 2251 2                    END;
; 2252 2
; 2253 2                    ! If the caller so requested, the address range into which the image and all
; 2254 2                    ! of its associated shareable images were mapped is returned. Like the previ
; 2255 2                    ! buffer, this address range must be probed in case the protection on the

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

32 V4.5-862 Page 47

```

X-14          END_PROCESSING - Final Steps of Image Activatio 10-Apr-1989 10:48:52    _$25
; 2256 2      ! pages changed since the argument was first validated.
; 2257 2
; 2258 2      IF .OWN_STORAGE [RETURN_ARRAY_ADDRESS] NEQA 0
; 2259 2      THEN
; 2260 3          BEGIN
; 2261 3
; 2262 3          BIND
; 2263 3              RETADR = .OWN_STORAGE [RETURN_ARRAY_ADDRESS]      : VECTOR;
; 2264 3
; 2265 3          ! Insure that two longwords are still writable
; 2266 3
; 2267 3          IF NOT PROBEW (%REF(0), %REF(8), .OWN_STORAGE [RETURN_ARRAY_ADDRESS])
; 2268 3          THEN RETURN SS$_ACCVIO;
; 2269 3
; 2270 3          ! Return the start and end addresses
; 2271 3
; 2272 3          RETADR [0] = .OWN_STORAGE [RETURN_START_ADDRESS];
; 2273 3          RETADR [1] = .OWN_STORAGE [RETURN_END_ADDRESS];
; 2274 3
; 2275 2          END;
; 2276 2
; 2277 2          ! The following kernel mode routine executes unconditionally to perform those
; 2278 2          ! completion chores that must be executed in kernel mode.
; 2279 2
; 2280 2          ARG_LIST [1] = .STACK_BASE;
; 2281 2          ARG_LIST [2] = IHD;
; 2282 2          ARG_LIST [3] = KFE;
; 2283 2
; P 2284 2          STATUS = $CMKRNL (
; P 2285 2              ROUTIN = SET_CONTROL_REGION,
; 2286 2              ARGVST = ARG_LIST);
; 2287 2          IF NOT .STATUS
; 2288 2          THEN RETURN .STATUS;
; 2289 2
; 2290 2          RETURN SS$_NORMAL
; 2291 2
; 2292 1          END;

```

```

.PSECT EXEC$PAGED_DATA, PIC,2
00000000 00000003 00010 P.AAC: .LONG 3 ;
00000000 00000000 00014 .LONG 0, 0, 3329 ;
.EXTRN SYS$EXPREG
.PSECT EXEC$PAGED_CODE, NOWRT, PIC,2
OFFC 00000 END_PROCESSING:
.WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 ; 1973
24 C2 00002 SUBL2 #36, SP ;
10 28 00005 MOV C3 #16, P.AAC, ARG_LIST ; 2013
1C AE 9E 0000C MOV AB CRETVA_RANGE, ARG_LIST+4 ; 2011
04 AE 7C 00011 CLR Q IMGIO_SEG_DESC ; 2014
04 AC D0 00014 MOV L ICB_ADDRESS, R9 ; 2018
00000000G 00 D0 00018 MOV L CTL$G_L_PHD, R1 ; 2020

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSSIMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 48

X-14                   END\_PROCESSING - Final Steps of Image Activatio 10-Apr-1989 10:48:52                   \_25

|           |      |       |       |       |            |   |        |
|-----------|------|-------|-------|-------|------------|---|--------|
| 50        | A9   | D5    | 0001F | TSTL  | 80(R9)     |   | ; 2028 |
|           | 0A   | 12    | 00022 | BNEQ  | 1\$        |   | ;      |
| 58        | A9   | D0    | 00024 | MOVL  | 88(R9), R0 |   | ; 2033 |
| 04        | A0   | D0    | 00028 | MOVL  | 4(R0), R8  |   | ; 2035 |
|           | 04   | 11    | 0002C | BRB   | 2\$        |   | ;      |
| 50        | A9   | D0    | 0002E | 1\$:  | MOVL       | 80(R9), R8  | ;      |
| 00000000G | 00   | E8    | 00032 | 2\$:  | BLBS       | OWN_STORAGE, 3\$                                    | ; 2044 |
|           | 00BB | 31    | 00039 |       | BRW        | 11\$  | ;      |
| 1E        | A8   | B5    | 0003C | 3\$:  | TSTW       | 30(R8)  | ; 2048 |
|           | 06   | 13    | 0003F |       | BEQL       | 4\$   | ;      |
| 1E        | A8   | 3C    | 00041 |       | MOVZWL     | 30(R8), IMAGE_IO_PAGE_COUNT                         | ;      |
|           | 07   | 11    | 00045 |       | BRB        | 5\$   | ;      |
| 00000000G | 00   | 3C    | 00047 | 4\$:  | MOVZWL     | SGN\$GW_IMGIOCNT, IMAGE_IO_PAGE_COUNT               | ; 2052 |
|           | 00   | ED    | 0004E | 5\$:  | CMPZV      | #0, #16, SGN\$GW_IMGIOCNT, -<br>IMAGE_IO_PAGE_COUNT | ; 2054 |
|           | 4D   | 18    | 00057 |       | BGEQ       | 8\$   | ;      |
|           | 09   | 78    | 00059 |       | ASHL       | #9, IMAGE_IO_PAGE_COUNT, R2                         | ; 2063 |
|           | 04   | E1    | 0005D |       | BBC        | #4, 32(R8), 6\$                                     | ; 2057 |
| 009C      | C1   | D0    | 00062 |       | MOVL       | 156(R1), CRETVA_RANGE                               | ; 2060 |
| 1C        | AE   | C1    | 00068 |       | ADDL3      | CRETVA_RANGE, R2, R0                                | ; 2063 |
| FF        | A0   | 9E    | 0006D |       | MOVAB      | -1(R0), CRETVA_RANGE+4                              | ; 2064 |
|           | 13   | 11    | 00072 |       | BRB        | 7\$   | ;      |
|           | 01   | C3    | 00074 | 6\$:  | SUBL3      | #1, CTL\$GL_CTLBASVA, CRETVA_RANGE+4                | ; 2068 |
|           | 52   | C3    | 0007D |       | SUBL3      | R2, CRETVA_RANGE+4, R0                              | ; 2071 |
| 01        | A0   | 9E    | 00082 |       | MOVAB      | 1(R0), CRETVA_RANGE                                 | ;      |
| 0C        | AE   | 9F    | 00087 | 7\$:  | PUSHAB     | ARG_LIST  | ; 2079 |
| 00000000G | 00   | 9F    | 0008A |       | PUSHAB     | MMG\$CRETVA   | ;      |
|           | 02   | FB    | 00090 |       | CALLS      | #2, SYSS\$CMKRNL                                    | ;      |
|           | 50   | D0    | 00097 |       | MOVL       | R0, STATUS  | ;      |
|           | 5B   | E9    | 0009A |       | BLBC       | STATUS, 10\$  | ; 2080 |
|           | 52   | D0    | 0009D |       | MOVL       | R2, IMGIO_SEG_DESC                                  | ; 2083 |
| 1C        | AE   | D0    | 000A1 |       | MOVL       | CRETVA_RANGE, IMGIO_SEG_DESC+4                      | ; 2084 |
| 04        | AE   | 9E    | 000A6 | 8\$:  | MOVAB      | IMGIO_SEG_DESC, R1                                  | ; 2088 |
| 20        | A8   | D0    | 000AA |       | MOVL       | 32(R8), R0  | ;      |
| 00000000G | 00   | 16    | 000AE |       | JSB        | RM\$SET   | ;      |
| 00000000G | 00   | D0    | 000B4 |       | MOVL       | OWN_STORAGE+28, R0                                  | ; 2097 |
|           | 04   | E1    | 000BB |       | BBC        | #4, 32(R8), 9\$                                     | ; 2090 |
|           | 03   | 7D    | 000C0 |       | MOVQ       | #3, -(SP)   | ; 2097 |
| 24        | AE   | 9F    | 000C3 |       | PUSHAB     | STACK_ARRAY   | ;      |
|           | 50   | DD    | 000C6 |       | PUSHL      | R0  | ;      |
|           | 04   | FB    | 000C8 |       | CALLS      | #4, SYSS\$EXPREG                                    | ;      |
|           | 50   | D0    | 000CF |       | MOVL       | R0, STATUS  | ;      |
|           | 01   | C1    | 000D2 |       | ADDL3      | #1, STACK_ARRAY+4, STACK_BASE                       | ; 2098 |
|           | 18   | 11    | 000D7 |       | BRB        | 10\$  | ;      |
|           | 01   | DD    | 000D9 | 9\$:  | PUSHL      | #1  | ; 2106 |
|           | 03   | DD    | 000DB |       | PUSHL      | #3  | ;      |
| 24        | AE   | 9F    | 000DD |       | PUSHAB     | STACK_ARRAY   | ;      |
|           | 50   | DD    | 000E0 |       | PUSHL      | R0  | ;      |
|           | 04   | FB    | 000E2 |       | CALLS      | #4, SYSS\$EXPREG                                    | ;      |
|           | 50   | D0    | 000E9 |       | MOVL       | R0, STATUS  | ;      |
|           | 01   | C1    | 000EC |       | ADDL3      | #1, STACK_ARRAY, STACK_BASE                         | ; 2107 |
|           | 5B   | E8    | 000F1 | 10\$: | BLBS       | STATUS, 11\$  | ; 2110 |
| 0112      | 31   | 000F4 |       |       | BRW        | 24\$  | ;      |
| 00000000G | 00   | D0    | 000F7 | 11\$: | MOVL       | OWN_STORAGE+64, R6                                  | ; 2127 |
|           | 03   | 12    | 000FE |       | BNEQ       | 12\$  | ;      |
| 00CB      | 31   | 00100 |       |       | BRW        | 20\$  | ;      |
|           | 00   | 0D    | 00103 | 12\$: | PROBEW     | #0, #512, (R6)                                      | ; 2145 |





**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSSIMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 50

X-14                    END\_PROCESSING - Final Steps of Image Activatio 10-Apr-1989 10:48:52    \_\$25

```
02 FB 001FC            CALLS    #2, SYS$CMKRNL                                    ;  
50 D0 00203            MOVL     R0, STATUS                                    ;  
5B E8 00206            BLBS     STATUS, 25$                                  ; 2287  
5B D0 00209 24$:        MOVL     STATUS, R0                                  ; 2288  
                      04 0020C            RET                                    ;  
01 D0 0020D 25$:        MOVL     #1, R0                                      ; 2290  
                      04 00210            RET                                    ;
```

; Routine Size: 529 bytes,        Routine Base: EXEC\$PAGED\_CODE + 06EB

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSSIMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 51

```

X-14          SET_CONTROL_REGION - Kernel Mode Completion Rou 10-Apr-1989 10:48:52    _$25
; 2294 1      %SBTTL 'SET_CONTROL_REGION - Kernel Mode Completion Routine'
; 2295 1
; 2296 1      ROUTINE SET_CONTROL_REGION (
; 2297 1          USER_STACK_ADDRESS,
; 2298 1          IHD_POINTER,
; 2299 1          KFE_POINTER) : SYS_CMKRNL =
; 2300 1
; 2301 1      !+
; 2302 1      ! Functional Description:
; 2303 1      !
; 2304 1      !     This routine performs the end processing that must be done in kernel
; 2305 1      !     mode. This includes loading the user stack pointer processor register,
; 2306 1      !     and setting up the process privilege mask. Note that these steps are
; 2307 1      !     only taken during the activation of a main program.
; 2308 1      !
; 2309 1      ! Calling Sequence:
; 2310 1      !
; 2311 1      !     $CMKRNL (SET_CONTROL_REGION, ARGUMENT_LIST)
; 2312 1      !
; 2313 1      ! Formal Parameters:
; 2314 1      !
; 2315 1      !     USER_STACK_ADDRESS - Address of base (high address end) of user stack
; 2316 1      !
; 2317 1      !     IHD_POINTER - Pointer to image header for this image
; 2318 1      !
; 2319 1      !     KFE_POINTER - Pointer to known file entry for this image, if one exists
; 2320 1      !-
; 2321 1
; 2322 2      BEGIN
; 2323 2
; 2324 2      MACRO
; 2325 2
; M 2326 2          MOVE_PRIV_MASK (SRC, DST) =
; M 2327 2
; M 2328 2              BEGIN
; M 2329 2                  VECTOR [DST,0] = .VECTOR [SRC,0];
; M 2330 2                  VECTOR [DST,1] = .VECTOR [SRC,1];
; 2331 2                  END% ,
; 2332 2
; M 2333 2          CLEAR_PRIV_MASK (DST) =
; M 2334 2
; M 2335 2              BEGIN
; M 2336 2                  VECTOR [DST,0] = 0;
; M 2337 2                  VECTOR [DST,1] = 0;
; 2338 2                  END% ,
; 2339 2
; M 2340 2          EXCLUDE_PRIV_MASK (MASK, DST) =
; M 2341 2
; M 2342 2              BEGIN
; M 2343 2                  VECTOR [DST,0] = .VECTOR [MASK,0] AND .VECTOR [DST,0];
; M 2344 2                  VECTOR [DST,1] = .VECTOR [MASK,1] AND .VECTOR [DST,1];
; 2345 2                  END% ,
; 2346 2
; M 2347 2          INCLUDE_PRIV_MASK (MASK, DST) =
; M 2348 2
; M 2349 2              BEGIN
; M 2350 2                  VECTOR [DST,0] = .VECTOR [MASK,0] OR .VECTOR [DST,0];

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 52

```

X-14          SET_CONTROL_REGION - Kernel Mode Completion Rou 10-Apr-1989 10:48:52    _$25
; M 2351 2          VECTOR [DST,1] = .VECTOR [MASK,1] OR .VECTOR [DST,1];
; 2352 2          END% ;
; 2353 2
; 2354 2          EXTERNAL REGISTER                                ! We enter this procedure with the P
; 2355 2          PCB = 4 : REF $BBLOCK;                          ! address contained in R4
; 2356 2
; 2357 2          BIND
; 2358 2          IHD = .IHD_POINTER : $BBLOCK,
; 2359 2          KFE = .KFE_POINTER : $BBLOCK,
; 2360 2          PHD = .CTL$GL_PHD : $BBLOCK;
; 2361 2
; 2362 2          LITERAL
; 2363 3          CMKRNL_OR_CMEXEC = (1 ^ $BITPOSITION (PRV$V_CMKRNL))
; 2364 2          OR
; 2365 2          (1 ^ $BITPOSITION (PRV$V_CMEXEC));
; 2366 2
; 2367 2          LOCAL
; 2368 2          PRIVILEGES : VECTOR [2],
; 2369 2          ICB : REF $BBLOCK;
; 2370 2
; 2371 2          ! Most of the operations in this routine are only performed when activating
; 2372 2          ! a main program. The only step that must be performed during a merged
; 2373 2          ! activation is the USECNT adjustment for the KFEs.
; 2374 2
; 2375 2          IF .OWN_STORAGE [MAIN_PROGRAM]
; 2376 2          THEN
; 2377 3          BEGIN
; 2378 3
; 2379 3          ! The high address end of the user stack is loaded into the stack limit
; 2380 3          ! array. The user stack pointer is initialized with a value that is smal
; 2381 3          ! than the input value by a value given by the EXTRA_USER_STACK compile
; 2382 3          ! time constant. The size of the user stack is stored in a cell that wil
; 2383 3          ! be used by the automatic stack expansion logic in EXCEPTION. Note that
; 2384 3          ! this number can never be smaller than 2.
; 2385 3
; 2386 3          CTL$AL_STACK [PSL$C_USER] = .USER_STACK_ADDRESS;
; 2387 3          MTPR (%REF (.USER_STACK_ADDRESS - (EXTRA_USER_STACK*BYTES_PER_PAGE))
; 2388 3          , PR$USP);
; 2389 3          IAC$GL_STACK_SIZE = .OWN_STORAGE [USER_STACK_SIZE];
; 2390 3
; 2391 3          ! The privilege mask that will be used while this image is executing mus
; 2392 3          ! be fabricated.
; 2393 3
; 2394 3          MOVE_PRIV_MASK (CTL$GQ_PROCPRIV, PRIVILEGES);          ! Start with process
; 2395 3
; 2396 3          ! Eliminate those not present in the image header
; 2397 3
; 2398 3          EXCLUDE_PRIV_MASK (IHD [IHD$Q_PRIVREQS], PRIVILEGES);
; 2399 3
; 2400 3          ! If the image was installed with privilege and we were called from othe
; 2401 3          ! user mode, then add the privileges from the KFE
; 2402 3
; 2403 3          IF .OWN_STORAGE [CALL_MODE] NEQ PSL$C_USER
; 2404 3          AND
; 2405 4          BEGIN
; 2406 4          IF .KFE_POINTER EQL 0
; 2407 4          THEN FALSE

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSSIMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 53

X-14

SET\_CONTROL\_REGION - Kernel Mode Completion Rou 10-Apr-1989 10:48:52 \_\$25

```

; 2408 4         ELSE .KFE [KFE$V_PROCPRIV]
; 2409 4         END
; 2410 3     THEN
; 2411 4         BEGIN
; 2412 4             INCLUDE_PRIV_MASK (KFE [KFE$Q_PROCPRIV], PRIVILEGES);
; 2413 4             MOVE_PRIV_MASK (KFE [KFE$Q_PROCPRIV], PHD [PHD$Q_IMAGPRIV]);
; 2414 4         END
; 2415 3     ELSE
; 2416 3         CLEAR_PRIV_MASK (PHD [PHD$Q_IMAGPRIV]);
; 2417 3
; 2418 3         ! Before the final privileges get stored, we need to check whether there
; 2419 3         ! is a system version mismatch in any of the images that was mapped. (Th
; 2420 3         ! mismatch was detected by the image header decode routine and remembere
; 2421 3         ! in the REMOVE PRIVILEGE flag.) If a mismatch was detected, and either
; 2422 3         ! CMEXEC or CMKRNL is set, then CMKRNL and CMEXEC privileges are turned
; 2423 3         ! off and an alternate status (SS$_SYSVERDIF) is returned.
; 2424 3
; 2425 4     IF (.OWN_STORAGE [REMOVE_PRIVILEGE])
; 2426 3         AND
; 2427 4         ((.PRIVILEGES [0] AND CMKRNL_OR_CMEXEC) NEQU 0)
; 2428 3     THEN
; 2429 4         BEGIN
; 2430 4             PRIVILEGES [0] = .PRIVILEGES [0] AND (NOT CMKRNL_OR_CMEXEC);
; 2431 4             OWN_STORAGE [FINAL_STATUS] = SS$_SYSVERDIF;
; 2432 3         END;
; 2433 3
; 2434 3         ! Store the privileges in the process header and in the PCB
; 2435 3
; 2436 3     MOVE_PRIV_MASK (PRIVILEGES, PCB [PCB$Q_PRIV]);
; 2437 3     MOVE_PRIV_MASK (PRIVILEGES, PHD [PHD$Q_PRIVMSK]);
; 2438 3
; 2439 3         ! The address of the image header buffer must be stored in the pointer p
; 2440 3
; 2441 3     CTL$GL_IMGHDRBF = .OWN_STORAGE [BUFFER_ADDRESS];
; 2442 3
; 2443 3         ! Finally, if image accounting was requested for this image, then the va
; 2444 3         ! image accounting cells must be initialized.
; 2445 3
; 2446 3     IF .OWN_STORAGE [IMAGE_ACCOUNT] OR .EXE$GL_ACMFLAGS [ACM$V_IMAGE]
; 2447 3     THEN
; 2448 4         BEGIN
; 2449 4             CTL$GL_ICPUTIM = .PHD [PHD$L_CPUTIM];
; 2450 4             CTL$GL_IFAULTS = .PHD [PHD$L_PAGEFLTS];
; 2451 4             CTL$GL_IFAULTIO = .PHD [PHD$L_PGFLTIO];
; 2452 4             CTL$GL_IWSPEAK = 0;
; 2453 4             CTL$GL_IPAGEFL = 0;
; 2454 4             CTL$GL_IDIOCNT = .PHD [PHD$L_DIOCNT];
; 2455 4             CTL$GL_IBIOCNT = .PHD [PHD$L_BIOCNT];
; 2456 4             CTL$GL_IVOLUMES = .CTL$GL_VOLUMES;
; 2457 4             CTL$GQ_ISTART [0] = .EXE$GQ_SYSTIME [0];
; 2458 4             CTL$GQ_ISTART [1] = .EXE$GQ_SYSTIME [1];
; 2459 3         END;
; 2460 3
; 2461 2     END;
; 2462 2
; 2463 2     ! The USECNT cell in each KFE must be incremented. In addition, any shared W
; 2464 2     ! must have its REFCNT incremented. If the adjusted REFCNT is larger than th

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSS\$IMGACT SYS\$IMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 54

```

X-14      SET_CONTROL_REGION - Kernel Mode Completion Rou 10-Apr-1989 10:48:52      _$25
; 2465 2      ! current high water REFCNT in the KFE, the KFE cell is adjusted. Finally, t
; 2466 2      ! DONE bit in the ICB is turned ON, indicating that the activation for each
; 2467 2      ! these ICBs is complete.
; 2468 2
; 2469 2      ICB = .IAC$GL_IMAGE_LIST;
; 2470 2      DO
; 2471 3          BEGIN
; 2472 3
; 2473 3          LOCAL
; 2474 3              KFE : REF $BBLOCK,
; 2475 3              WCB : REF $BBLOCK;
; 2476 3
; 2477 3          IF NOT .ICB [ICB$V_DONE]
; 2478 3          THEN
; 2479 4              BEGIN
; 2480 4              ICB [ICB$V_DONE] = TRUE;
; 2481 4              KFE = .ICB [ICB$L_KFE];
; 2482 4              IF .KFE NEQU 0
; 2483 4              THEN
; 2484 5                  BEGIN
; 2485 5                  KFE [KFE$L_USECNT] = .KFE [KFE$L_USECNT] + 1;
; 2486 5                  IF .KFE [KFE$V_OPEN]
; 2487 5                  THEN
; 2488 6                      BEGIN
; 2489 6                      WCB = .KFE [KFE$L_WCB];
; 2490 6                      IF .WCB [WCB$W_REFCNT] GTRU .KFE [KFE$W_SHRCNT]
; 2491 6                      THEN KFE [KFE$W_SHRCNT] = .WCB [WCB$W_REFCNT];
; 2492 5                      END;
; 2493 4                  END;
; 2494 3              END;
; 2495 3              ICB = .ICB [ICB$L_FLINK];
; 2496 3          END
; 2497 2          UNTIL .ICB EQLA IAC$GL_IMAGE_LIST;
; 2498 2
; 2499 2          IF .OWN_STORAGE [RMS_BASE] NEQU 0
; 2500 2          THEN CTL$GL_RMSBASE = .OWN_STORAGE [RMS_BASE];
; 2501 2
; 2502 2          RETURN SS$_NORMAL
; 2503 2
; 2504 1      END;

```

```

006C 00000 SET_CONTROL_REGION:
          .WORD      Save R2,R3,R5,R6                      ; 2296
00000000G 00 9E 00002      MOVAB      IAC$GL_IMAGE_LIST, R6      ;
00000000G 00 9E 00009      MOVAB      OWN_STORAGE, R5      ;
          08 C2 00010      SUBL2      #8, SP      ;
          08 AC D0 00013      MOVL      IHD_POINTER, R3      ; 2358
          0C AC D0 00017      MOVL      KFE_POINTER, R2      ; 2359
00000000G 00 D0 0001B      MOVL      CTL$GL_PHD, R1      ; 2360
          65 E8 00022      BLBS      OWN_STORAGE, 1$      ; 2375
          00E2 31 00025      BRW      6$      ;
          04 AC D0 00028 1$:      MOVL      USER_STACK_ADDRESS, CTL$AL_STACK+12      ; 2386
          04 AC D0 00030      MOVL      USER_STACK_ADDRESS, R0      ; 2387

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYS\$IMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 55

X-14 SET\_CONTROL\_REGION - Kernel Mode Completion Rou 10-Apr-1989 10:48:52 \_\$25

|           |    |               |        |                                    |        |
|-----------|----|---------------|--------|------------------------------------|--------|
| FC00      | C0 | 9E 00034      | MOVAB  | -1024(R0), R0                      | ;      |
|           | 50 | DA 00039      | MTPR   | R0, #3                             | ; 2388 |
| 1C        | A5 | D0 0003C      | MOVL   | OWN_STORAGE+28, IAC\$GL_STACK_SIZE | ; 2389 |
| 00000000G | 00 | 7D 00044      | MOVQ   | CTL\$GQ_PROCPRIV, PRIVILEGES       | ; 2394 |
| 14        | A3 | 9E 0004B      | MOVAB  | 20(R3), R0                         | ; 2398 |
|           | 60 | D2 0004F      | MCOML  | (R0), R3                           | ;      |
|           | 53 | CA 00052      | BICL2  | R3, PRIVILEGES                     | ;      |
| 04        | A0 | D2 00055      | MCOML  | 4(R0), R3                          | ;      |
|           | 53 | CA 00059      | BICL2  | R3, PRIVILEGES+4                   | ;      |
| 24        | A5 | 91 0005D      | CMPB   | OWN_STORAGE+36, #3                 | ; 2403 |
|           | 20 | 13 00061      | BEQL   | 2\$                                | ;      |
| 0C        | AC | D5 00063      | TSTL   | KFE_POINTER                        | ; 2406 |
|           | 1B | 13 00066      | BEQL   | 2\$                                | ;      |
|           | 02 | E1 00068      | BBC    | #2, 16(R2), 2\$                    | ; 2408 |
| 20        | A2 | 9E 0006D      | MOVAB  | 32(R2), R0                         | ; 2412 |
|           | 60 | C8 00071      | BISL2  | (R0), PRIVILEGES                   | ;      |
| 04        | A0 | C8 00074      | BISL2  | 4(R0), PRIVILEGES+4                | ;      |
| 0110      | C1 | 9E 00079      | MOVAB  | 272(R1), R2                        | ; 2413 |
|           | 60 | 7D 0007E      | MOVQ   | (R0), (R2)                         | ;      |
|           | 07 | 11 00081      | BRB    | 3\$                                | ;      |
| 0110      | C1 | 9E 00083 2\$: | MOVAB  | 272(R1), R0                        | ; 2416 |
|           | 60 | 7C 00088      | CLRQ   | (R0)                               | ;      |
|           | 03 | E1 0008A 3\$: | BBC    | #3, OWN_STORAGE, 4\$               | ; 2425 |
|           | 6E | 93 0008E      | BITB   | PRIVILEGES, #3                     | ; 2427 |
|           | 09 | 13 00091      | BEQL   | 4\$                                | ;      |
|           | 03 | 8A 00093      | BICB2  | #3, PRIVILEGES                     | ; 2430 |
| 239C      | 8F | 3C 00096      | MOVZWL | #9116, OWN_STORAGE+16              | ; 2431 |
| 0088      | C4 | 9E 0009C 4\$: | MOVAB  | 136(PCB), R0                       | ; 2436 |
|           | 6E | 7D 000A1      | MOVQ   | PRIVILEGES, (R0)                   | ;      |
|           | 6E | 7D 000A4      | MOVQ   | PRIVILEGES, (R1)                   | ; 2437 |
| 40        | A5 | D0 000A7      | MOVL   | OWN_STORAGE+64, CTL\$GL_IMGHDRBF   | ; 2441 |
|           | 02 | E0 000AF      | BBS    | #2, OWN_STORAGE, 5\$               | ; 2446 |
|           | 01 | E1 000B3      | BBC    | #1, EXE\$GL_ACMFLAGS, 6\$          | ;      |
| 00AC      | C1 | D0 000BB 5\$: | MOVL   | 172(R1), CTL\$GL_ICPUTIM           | ; 2449 |
| 00C0      | C1 | D0 000C4      | MOVL   | 192(R1), CTL\$GL_IFAULTS           | ; 2450 |
| 012C      | C1 | D0 000CD      | MOVL   | 300(R1), CTL\$GL_IFAULTIO          | ; 2451 |
| 00000000G | 00 | D4 000D6      | CLRL   | CTL\$GL_IWSPEAK                    | ; 2452 |
| 00000000G | 00 | D4 000DC      | CLRL   | CTL\$GL_IPAGEFL                    | ; 2453 |
| 00CC      | C1 | D0 000E2      | MOVL   | 204(R1), CTL\$GL_IDIOCNT           | ; 2454 |
| 00D0      | C1 | D0 000EB      | MOVL   | 208(R1), CTL\$GL_IBIOCNT           | ; 2455 |
| 00000000G | 00 | D0 000F4      | MOVL   | CTL\$GL_VOLUMES, CTL\$GL_IVOLUMES  | ; 2456 |
| 00000000G | 00 | 7D 000FF      | MOVQ   | EXE\$GQ_SYSTIME, CTL\$GQ_ISTART    | ; 2458 |
|           | 66 | D0 0010A 6\$: | MOVL   | IAC\$GL_IMAGE_LIST, ICB            | ; 2469 |
|           | 06 | E0 0010D 7\$: | BBS    | #6, 16(ICB), 8\$                   | ; 2477 |
| 40        | 8F | 88 00112      | BISB2  | #64, 16(ICB)                       | ; 2480 |
| 54        | A2 | D0 00117      | MOVL   | 84(ICB), KFE                       | ; 2481 |
|           | 18 | 13 0011B      | BEQL   | 8\$                                | ; 2482 |
| 14        | A0 | D6 0011D      | INCL   | 20(KFE)                            | ; 2485 |
|           | 03 | E1 00120      | BBC    | #3, 16(KFE), 8\$                   | ; 2486 |
| 18        | A0 | D0 00125      | MOVL   | 24(KFE), WCB                       | ; 2489 |
| 0E        | A1 | B1 00129      | CMPW   | 14(WCB), 52(KFE)                   | ; 2490 |
|           | 05 | 1B 0012E      | BLEQU  | 8\$                                | ;      |
| 0E        | A1 | B0 00130      | MOVW   | 14(WCB), 52(KFE)                   | ; 2491 |
|           | 62 | D0 00135 8\$: | MOVL   | (ICB), ICB                         | ; 2495 |
|           | 66 | 9E 00138      | MOVAB  | IAC\$GL_IMAGE_LIST, R0             | ; 2497 |
|           | 52 | D1 0013B      | CMPL   | ICB, R0                            | ;      |
|           | CD | 12 0013E      | BNEQ   | 7\$                                | ;      |

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 56

X-14            SET\_CONTROL\_REGION - Kernel Mode Completion Rou 10-Apr-1989 10:48:52        \_\$25

|    |    |    |            |      |                     |  |        |
|----|----|----|------------|------|---------------------|--|--------|
| 2C | A5 | D0 | 00140      | MOVL | OWN_STORAGE+44, R0  |  | ; 2499 |
|    | 07 | 13 | 00144      | BEQL | 9\$                 |  | ;      |
|    | 50 | D0 | 00146      | MOVL | R0, CTL\$GL_RMSBASE |  | ; 2500 |
|    | 01 | D0 | 0014D 9\$: | MOVL | #1, R0              |  | ; 2502 |
|    |    | 04 | 00150      | RET  |                     |  | ;      |

; Routine Size: 337 bytes,        Routine Base: EXEC\$PAGED\_CODE + 08FC



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSSIMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 57

```
X-14          GET_LOCK - Lock Known File Data Base for Read A 10-Apr-1989 10:48:52    _$25
; 2506 1      %SBTTL 'GET_LOCK - Lock Known File Data Base for Read Access'
; 2507 1
; 2508 1      ROUTINE GET_LOCK =
; 2509 1
; 2510 1      !+
; 2511 1      ! Functional Description:
; 2512 1      !
; 2513 1      !   This routine locks the known file data base for read access. The image
; 2514 1      !   activator maintains this lock for the entire time that it will be openi
; 2515 1      !   files. Note that no lock is taken out during system initialization. unt
; 2516 1      !   INSTALL first executes and sets up the known file lists (and loads
; 2517 1      !   EXE$GL_KNOWN_FILES with nonzero contents.
; 2518 1      !
; 2519 1      ! Calling Sequence:
; 2520 1      !
; 2521 1      !   GET_LOCK ()
; 2522 1      !-
; 2523 1
; 2524 2      BEGIN
; 2525 2
; 2526 2      LOCAL STATUS;
; 2527 2
; 2528 2      IF .EXE$GL_KNOWN_FILES NEQ 0
; 2529 2      THEN
; 2530 3          BEGIN
; 2531 3
; P 2532 3          STATUS = $ENQW (
; P 2533 3              EFN = EXE$C_SYSEFN,
; P 2534 3              LKMODE = LCK$K_PMODE,
; P 2535 3              LKSB = OWN_STORAGE [LOCK_STATUS_BLOCK],
; P 2536 3              FLAGS = LCK$M_SYSTEM,
; P 2537 3              RESNAM = EXE$GQ_KFE_LCKNAM,
; P 2538 3              PARID = .EXE$GL_SYSID_LOCK,
; 2539 3              ACMODE = PSL$C_EXEC);          ! End of routine GET_LOCK
; 2540 3
; 2541 3          IF NOT .STATUS
; 2542 3          THEN RETURN .STATUS
; 2543 3          ELSE RETURN .OWN_STORAGE [LOCK_STATUS];
; 2544 3
; 2545 3          END
; 2546 2      ELSE      ! No need to take out a lock yet
; 2547 2          RETURN SS$_NORMAL
; 2548 2
; 2549 1      END;
```

.EXTRN SYS\$ENQW

0004 00000 GET\_LOCK:

```
          .WORD    Save R2          ; 2508
00000000G 00 9E 00002    MOVAB    OWN_STORAGE+4, R2          ;
00000000G 00 D5 00009    TSTL    EXE$GL_KNOWN_FILES      ; 2528
          2A 13 0000F    BEQL    1$                      ;
          01 7D 00011    MOVQ    #1, -(SP)                ; 2539
          7E 7C 00014    CLRQ    -(SP)                    ;
          7E D4 00016    CLRL    -(SP)                    ;
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 58

X-14 GET\_LOCK - Lock Known File Data Base for Read A 10-Apr-1989 10:48:52 \_\$25

```
00000000G 00 DD 00018      PUSHL  EXE$GL_SYSID_LOCK      ;
00000000G 00 9F 0001E      PUSHAB EXE$GQ_KFE_LCKNAM      ;
          10 DD 00024      PUSHL  #16                    ;
          52 DD 00026      PUSHL  R2                      ;
          03 DD 00028      PUSHL  #3                      ;
          00G 9A 0002A      MOVZBL S^EXE$C_SYSEFN, -(SP)   ;
          0B FB 0002D      CALLS  #11, SYS$ENQW          ;
          50 E9 00034      BLBC   STATUS, 2$            ; 2541
          62 3C 00037      MOVZWL OWN_STORAGE+4, R0      ; 2543
          04 0003A      RET                                ;
          01 D0 0003B 1$:  MOVL   #1, R0                ; 2547
          04 0003E 2$:  RET                                ;
```

; Routine Size: 63 bytes, Routine Base: EXEC\$PAGED\_CODE + 0A4D

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 59

```

X-14          RELEASE_LOCK - Unlock Lock Known File Data Base 10-Apr-1989 10:48:52    _$25
; 2551 1      %SBTTL 'RELEASE_LOCK - Unlock Lock Known File Data Base'
; 2552 1
; 2553 1      ROUTINE RELEASE_LOCK =
; 2554 1
; 2555 1      !+
; 2556 1      ! Functional Description:
; 2557 1      !
; 2558 1      !   This routine unlocks the known file data base. This happens when
; 2559 1      !   the image activator reaches the point where there are no more files
; 2560 1      !   to open, or when an error occurs with the lock granted.
; 2561 1      !
; 2562 1      ! Calling Sequence:
; 2563 1      !
; 2564 1      !   RELEASE_LOCK ()
; 2565 1      !-
; 2566 1
; 2567 1      IF .OWN_STORAGE [LOCK_ID] NEQ 0
; 2568 1      THEN
; 2569 2          $DEQ (LKID = .OWN_STORAGE [LOCK_ID])
; 2570 1      ELSE
; 2571 1          SS$NORMAL;                                ! End of routine RELEASE_LOCK

```

.EXTRN SYS\$DEQ

```

0000 00000 RELEASE_LOCK:
; 2553          .WORD      Save nothing
; 2567          0000000G 00 D0 00002      MOVL      OWN_STORAGE+8, R0
; 2569          0E 13 00009      BEQL      1$
; 2569          7E 7C 0000B      CLRQ      -(SP)
; 2569          7E D4 0000D      CLRL      -(SP)
; 2569          50 DD 0000F      PUSHL     R0
; 2569          04 FB 00011      CALLS     #4, SYS$DEQ
; 2569          04 00018      RET
; 2571          01 D0 00019 1$:      MOVL      #1, R0
; 2571          04 0001C      RET

```

; Routine Size: 29 bytes, Routine Base: EXEC\$PAGED\_CODE + 0A8C

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 60

```

X-14      IMG$ALLOCATE_ICB - Lock Known File Data Base fo 10-Apr-1989 10:48:52      _$25
; 2573 1      %SBTTL 'IMG$ALLOCATE_ICB - Lock Known File Data Base for Read Access'
; 2574 1
; 2575 1      GLOBAL ROUTINE IMG$ALLOCATE_ICB (ICB_POINTER) =
; 2576 1
; 2577 1      !+
; 2578 1      ! Functional Description:
; 2579 1      !
; 2580 1      !     This routine allocates an image control block for use by later stages
; 2581 1      !     of the image activator. An allocation request is first made from a
; 2582 1      !     pool of previously used ICBs. A trip into kernel mode is thus only
; 2583 1      !     required if this lookaside list request fails. In either case, the
; 2584 1      !     ICB is entirely filled with zeros.
; 2585 1      !
; 2586 1      !     Note that only the process allocation region is used to insure that
; 2587 1      !     no ICBs are created in P0 space. This would be no problem on merged
; 2588 1      !     activations but would mess up the simple execution of an image.
; 2589 1      !
; 2590 1      ! Calling Sequence:
; 2591 1      !
; 2592 1      !     IMG$ALLOCATE_ICB (ICB_POINTER)
; 2593 1      !
; 2594 1      ! Formal Parameter:
; 2595 1      !
; 2596 1      !     ICB_POINTER - Address of cell that will receive the address of a newly
; 2597 1      !     allocated image control block.
; 2598 1      !
; 2599 1      ! Status Return:
; 2600 1      !
; 2601 1      !     SSS_NORMAL - ICB successfully allocated
; 2602 1      !
; 2603 1      !     SSS_INSFMEM - Unable to allocate ICB
; 2604 1      !-
; 2605 1
; 2606 2      BEGIN
; 2607 2
; 2608 2      LOCAL
; 2609 2          SIZE,
; 2610 2          ICB : REF $BLOCK;
; 2611 2
; 2612 2      IF REMQUE (.IAC$GL_ICBFL, ICB)
; 2613 2      THEN IF NOT EXE$ALOP1PROC (ICB$K_LENGTH; SIZE, ICB)
; 2614 2          THEN RETURN SSS_INSFMEM;
; 2615 2      CH$FILL (0, ICB$K_LENGTH, .ICB);
; 2616 2      ICB [ICB$W_SIZE] = ICB$K_LENGTH;
; 2617 2      ICB [ICB$B_TYPE] = ICB$K_ICB_TYPE_CODE;
; 2618 2      ICB [ICB$L_STARTING_ADDRESS] = -1;
; 2619 2      ICB [ICB$L_END_ADDRESS] = -1;
; 2620 2      .ICB_POINTER = .ICB;
; 2621 2      RETURN SSS_NORMAL;
; 2622 2
; 2623 1      END;

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 61

```

X-14          IMG$ALLOCATE_ICB - Lock Known File Data Base fo 10-Apr-1989 10:48:52      _$25
          007C 00000          .ENTRY  IMG$ALLOCATE_ICB, Save R2,R3,R4,R5,R6          ; 2575
00000000G 00 9E 00002          MOVAB  IAC$GL_ICBFL, R0                          ; 2612
          00  B0 0F 00009          REMQUE @0(R0), ICB                          ;
          16  1C 0000D          BVC   1$                          ;
          68  8F 9A 0000F          MOVZBL #104, R1                          ; 2613
00000000G 00 16 00013          JSB   EXE$ALOP1PROC                          ;
          52  D0 00019          MOVL  R2, R6                          ;
          50  E8 0001C          BLBS  R0, 1$                          ;
          0124 8F 3C 0001F          MOVZWL #292, R0                          ; 2614
          04  00024          RET                                     ;
          00  2C 00025 1$:          MOVCS  #0, (SP), #0, #104, (ICB)          ; 2615
          66      0002C          ;                                     ;
          68  8F 9B 0002D          MOVZBW #104, 8(ICB)                          ; 2616
          7F  8F 90 00032          MOVB  #127, 10(ICB)                          ; 2617
          01  CE 00037          MNEGL #1, 72(ICB)                          ; 2618
          01  CE 0003B          MNEGL #1, 76(ICB)                          ; 2619
          56  D0 0003F          MOVL  ICB, @ICB_POINTER                      ; 2620
          01  D0 00043          MOVL  #1, R0                              ; 2621
          04  00046          RET                                     ;

```

; Routine Size: 71 bytes, Routine Base: EXEC\$PAGED\_CODE + 0AA9

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 62

```

X-14      IMG$DEALLOCATE_ICB - Deallocate an Unused ICB    10-Apr-1989 10:48:52    _$25
; 2625 1      %SBTTL 'IMG$DEALLOCATE_ICB - Deallocate an Unused ICB'
; 2626 1
; 2627 1      GLOBAL ROUTINE IMG$DEALLOCATE_ICB (ICB) : NOVALUE =
; 2628 1
; 2629 1      !+
; 2630 1      ! Functional Description:
; 2631 1      !
; 2632 1      ! This routine deallocates an ICB that was not used. This can be due to
; 2633 1      ! one of three reasons:
; 2634 1      !
; 2635 1      ! The current image is being rundown, thus releasing all of its
; 2636 1      ! image control blocks back to pool.
; 2637 1      !
; 2638 1      ! An ICB was allocated during image activation for an image that
; 2639 1      ! has already been activated.
; 2640 1      !
; 2641 1      ! An error occurred, requiring that all images activated during the
; 2642 1      ! current call be eliminated.
; 2643 1      !
; 2644 1      ! If the ICB was allocated from P1 space, it is deallocated to a linked
; 2645 1      ! list of free ICBs that will be used during later activations. ICBs
; 2646 1      ! allocated from P0 space use the normal deallocation routine.
; 2647 1      !
; 2648 1      ! Calling Sequence:
; 2649 1      !
; 2650 1      ! IMG$DEALLOCATE_ICB (ICB_ADDRESS)
; 2651 1      !
; 2652 1      ! Formal Parameter:
; 2653 1      !
; 2654 1      ! ICB_ADDRESS - Address of ICB that is being deallocated
; 2655 1      !-
; 2656 1
; 2657 2      BEGIN
; 2658 2
; 2659 2      MAP
; 2660 2          ICB : REF $BBLOCK;
; 2661 2
; 2662 2      BIND VA = ICB : $BBLOCK;
; 2663 2
; 2664 2      IF .VA [VA$V_P1]
; 2665 2      THEN INSQUE (.ICB, .IAC$GL_ICBFL [1]) ! Insert at tail of lookaside list
; 2666 2      ELSE EXE$DEAP1 (.ICB, .ICB [ICB$W_SIZE]);
; 2667 2
; 2668 1      END;

```

```

          000C 00000          .ENTRY IMG$DEALLOCATE_ICB, Save R2,R3          ; 2627
04 AC D0 00002          MOVL ICB, R0          ; 2665
          06 E1 00006          BBC #6, VA+3, 1$          ; 2664
00000000G 00 9E 0000B          MOVAB IAC$GL_ICBFL+4, R1          ; 2665
          60 0E 00012          INSQUE (R0), @0(R1)          ;
          04 00016          RET          ;
          08 A0 3C 00017 1$:          MOVZWL 8(R0), R1          ; 2666
00000000G 00 16 0001B          JSB EXE$DEAP1          ;

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 63  
X-14 IMG\$DEALLOCATE\_ICB - Deallocate an Unused ICB 10-Apr-1989 10:48:52 \_\$25  
04 00021 RET ; 2668  
; Routine Size: 34 bytes, Routine Base: EXEC\$PAGED\_CODE + 0AF0

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 64

```

X-14      SET_VECTORS - Prepare privileged vectors for ex 10-Apr-1989 10:48:52      _$25
; 2670 1      %SBTTL 'SET_VECTORS - Prepare privileged vectors for execution'
; 2671 1
; 2672 1      ROUTINE SET_VECTORS =
; 2673 1
; 2674 1      !+
; 2675 1      ! Functional Description:
; 2676 1      !
; 2677 1      !     This routine takes the privileged vector entries that contain RSB
; 2678 1      !     instructions and replaces each with a JSB instruction.
; 2679 1      !
; 2680 1      ! Calling Sequence:
; 2681 1      !
; 2682 1      !     SET_VECTORS ()
; 2683 1      !
; 2684 1      ! Input Parameters:
; 2685 1      !
; 2686 1      !     none
; 2687 1      !
; 2688 1      ! Implicit Input:
; 2689 1      !
; 2690 1      !     IAC$AW_VECSET - Array that locates the dividing point in each vector
; 2691 1      !     list between those vectors that already existed and those that were
; 2692 1      !     added as part of the latest activation. This is the starting point f
; 2693 1      !     the search.
; 2694 1      !
; 2695 1      !     CTL$A_DISPVEC - This address locates the start of the three-page area
; 2696 1      !     containing the privileged vectors. The first longword of each area
; 2697 1      !     contains the current end of the vector list. This is the end point f
; 2698 1      !     the search.
; 2699 1      !-
; 2700 1
; 2701 2      BEGIN
; 2702 2
; 2703 2      LITERAL
; 2704 2          ABSOLUTE_MODE = %X'9F',
; 2705 2          JSB_ABSOLUTE = (ABSOLUTE_MODE ^ 8) OR OP$JSB : UNSIGNED (16),
; 2706 2          RSB_ABSOLUTE = (ABSOLUTE_MODE ^ 8) OR OP$_RSB : UNSIGNED (16);
; 2707 2
; 2708 2      LOCAL
; 2709 2          I, J;
; 2710 2
; 2711 2      ! Do the privileged vectors first
; 2712 2
; 2713 2      INCRU I FROM 0 TO 3 DO
; 2714 3          BEGIN
; 2715 3
; 2716 3          BIND
; 2717 3              END_POINT = CTL$A_DISPVEC + (.I * 256) : LONG,
; 2718 3              DISPVEC = CTL$A_DISPVEC + (.I * 256) : VECTOR [256,BYTE];
; 2719 3
; 2720 3          J = .IAC$AW_VECSET [.I];
; 2721 3          WHILE .J LSSU .END_POINT DO
; 2722 3
; 2723 4              BEGIN
; 2724 4
; 2725 4              BIND OPCODE = DISPVEC [.J] : WORD;
; 2726 4

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 65

```

X-14          SET_VECTORS - Prepare privileged vectors for ex 10-Apr-1989 10:48:52    _$25
; 2727 4          IF .OPCODE EQLU RSB_ABSOLUTE
; 2728 4          THEN OP CODE = JSB_ABSOLUTE;
; 2729 4
; 2730 4          J = .J + 6;
; 2731 4
; 2732 3          END;
; 2733 3
; 2734 3          IAC$AW_VECSET [.I] = .END_POINT;
; 2735 3
; 2736 2          END;
; 2737 2
; 2738 2          ! Now do the message sections
; 2739 2
; 2740 2          IAC$AW_VECSET [4] = .(CTL$A_DISPVEC + MESSAGE_OFFSET);
; 2741 2
; 2742 2          RETURN SS$_NORMAL;
; 2743 2
; 2744 1          END;

```

000C 00000 SET\_VECTORS:

|           |      |       |       |            |                                       |        |
|-----------|------|-------|-------|------------|---------------------------------------|--------|
|           |      |       | .WORD | Save R2,R3 |                                       | ; 2672 |
|           | 50   | D4    | 00002 | CLRL       | I                                     | ; 2713 |
|           | 08   | 78    | 00004 | 1\$: ASHL  | #8, I, R1                             | ; 2717 |
| 00000000G | 041  | 9E    | 00008 | MOVAB      | CTL\$A_DISPVEC[R1], R1                | ; 2720 |
| 00000000G | 040  | 3E    | 00010 | MOVAV      | IAC\$AW_VECSET[I], R2                 | ; 2720 |
|           | 62   | 3C    | 00018 | MOVZWL     | (R2), J                               | ; 2721 |
|           | 53   | D1    | 0001B | 2\$: CML   | J, (R1)                               | ; 2721 |
|           | 17   | 1E    | 0001E | BGEQU      | 4\$                                   | ; 2727 |
|           | 6341 | 9F    | 00020 | PUSHAB     | (J)[R1]                               | ; 2727 |
|           | 9E   | B1    | 00023 | CMPW       | @(SP)+, #40709                        | ; 2728 |
|           | 08   | 12    | 00028 | BNEQ       | 3\$                                   | ; 2728 |
|           | 6341 | 9F    | 0002A | PUSHAB     | (J)[R1]                               | ; 2728 |
| 9F16      | 8F   | B0    | 0002D | MOVW       | #-24810, @(SP)+                       | ; 2730 |
|           | 06   | C0    | 00032 | 3\$: ADDL2 | #6, J                                 | ; 2730 |
|           | E4   | 11    | 00035 | BRB        | 2\$                                   | ; 2734 |
|           | 61   | B0    | 00037 | 4\$: MOVW  | (R1), (R2)                            | ; 2734 |
|           | 50   | D6    | 0003A | INCL       | I                                     | ; 2713 |
|           | 50   | D1    | 0003C | CML        | I, #3                                 | ; 2740 |
|           | C3   | 1B    | 0003F | BLEQU      | 1\$                                   | ; 2742 |
| 00000000G | 00   | B0    | 00041 | MOVW       | CTL\$A_DISPVEC+1024, IAC\$AW_VECSET+8 | ; 2740 |
|           | 01   | D0    | 0004C | MOVL       | #1, R0                                | ; 2742 |
|           | 04   | 0004F | RET   |            |                                       | ; 2744 |

; Routine Size: 80 bytes, Routine Base: EXEC\$PAGED\_CODE + 0B12

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 66

```

X-14      ERROR_CLEAN_UP - Clean Up after an Error is Det 10-Apr-1989 10:48:52    _$25
; 2746 1   %SBTTL 'ERROR_CLEAN_UP - Clean Up after an Error is Detected'
; 2747 1
; 2748 1   ROUTINE ERROR_CLEAN_UP : NOVALUE =
; 2749 1
; 2750 1   !+
; 2751 1   ! Functional Description:
; 2752 1   !
; 2753 1   !     This routine cleans up when an error is detected after some successful
; 2754 1   !     work has been completed.
; 2755 1   !
; 2756 1   !     All ICBs on the WORK list are simply deallocated.
; 2757 1   !
; 2758 1   !     ICBs that exist on the IMAGE (so-called done) list with the DONE bi
; 2759 1   !     clear indicate images that have been successfully activated as a ps
; 2760 1   !     of this activation before an error was detected.
; 2761 1   !
; 2762 1   !     The address space associated with these images is deleted.
; 2763 1   !
; 2764 1   !     The channel on which each image file was opened is closed.
; 2765 1   !
; 2766 1   !     Each ICB is then deallocated.
; 2767 1   !
; 2768 1   !     ICBs on the IMAGE list with the DONE bit clear must have their
; 2769 1   !     ACTIVE_SONS field reset. This is non-zero if some referenced image
; 2770 1   !     (a son) has not yet been fully processed. If we didn't clear it,
; 2771 1   !     on a later mapping attempt we might report a false circularity.
; 2772 1   !
; 2773 1   !     Note that an ICB with the addresses mapped bit not yet set will
; 2774 1   !     not have any address space deleted.
; 2775 1   !
; 2776 1   ! Calling Sequence:
; 2777 1   !
; 2778 1   !     ERROR_CLEAN_UP ()
; 2779 1   !
; 2780 1   ! Formal Parameters:
; 2781 1   !
; 2782 1   !     none
; 2783 1   !
; 2784 1   ! Implicit Input:
; 2785 1   !
; 2786 1   !     IAC$GL_IMAGE_LIST - List of ICBs representing images that have been
; 2787 1   !     successfully activated
; 2788 1   !
; 2789 1   !     IAC$GL_WORK_LIST - List of ICBs representing work left to be done.
; 2790 1   !-
; 2791 1
; 2792 2   BEGIN
; 2793 2
; 2794 2   LOCAL
; 2795 2       ICB      : REF $BBLOCK,
; 2796 2       NEXT_ICB : REF $BBLOCK;
; 2797 2
; 2798 2   ! Simply deallocate the ICBs in the work list
; 2799 2
; 2800 2   WHILE NOT (REMQUE (.IAC$GL_WORK_LIST, ICB)) DO
; 2801 2       IMG$DEALLOCATE_ICB (.ICB);
; 2802 2

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

```

SYS$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-
32 V4.5-862 Page 67
X-14          ERROR_CLEAN_UP - Clean Up after an Error is Det 10-Apr-1989 10:48:52  _$25
; 2803 2      ! Traverse the done list, looking for ICBs with the done bit not yet set.
; 2804 2
; 2805 2      NEXT_ICB = .IAC$GL_IMAGE_LIST;
; 2806 2
; 2807 2      WHILE .NEXT_ICB NEQA IAC$GL_IMAGE_LIST DO
; 2808 2          IF .NEXT_ICB [ICB$V_DONE]
; 2809 2              THEN
; 2810 3                  BEGIN
; 2811 3                      NEXT_ICB [ICB$L_ACTIVE_SONS] = 0; ! Must be reset if error occurred
; 2812 3                      NEXT_ICB = .NEXT_ICB [ICB$L_FLINK];
; 2813 3                  END
; 2814 2              ELSE
; 2815 3                  BEGIN
; 2816 3                      REMQUE (.NEXT_ICB, ICB);
; 2817 3                      IF .ICB [ICB$V_MAPPED]
; 2818 3                          THEN
; P 2819 3                              $DELTV (
; P 2820 3                                  INADR = ICB [ICB$Q_ADDRESS_RANGE],
; 2821 3                                  ACMODE = .OWN_STORAGE[ACCESS_MODE]);
; 2822 3
; 2823 3                      $DASSGN (CHAN = .ICB [ICB$W_CHAN]);
; 2824 3                      NEXT_ICB = .ICB [ICB$L_FLINK];
; 2825 3                      IMG$DEALLOCATE_ICB (.ICB);
; 2826 2                  END;
; 2827 2
; 2828 2      IAC$GL_IMAGCTX [IMAGCTX$V_SETVECTOR] = FALSE;
; 2829 2
; 2830 1      END;

```

```

                                .EXTRN  SYS$DELTV
001C 00000 ERROR_CLEAN_UP:
                                .WORD   Save R2,R3,R4                ; 2748
00000000G 00 9E 00002           MOVAB  IAC$GL_IMAGE_LIST, R4                ;
00000000G 00 9E 00009 1$:      MOVAB  IAC$GL_WORK_LIST, R0                ; 2800
      00  B0 0F 00010           REMQUE @0(R0), ICB                ;
      09 1D 00014           BVS    2$                ;
      52 DD 00016           PUSHL  ICB                ; 2801
      01 FB 00018           CALLS #1, IMG$DEALLOCATE_ICB    ;
      EA 11 0001D           BRB   1$                ;
      64 D0 0001F 2$:      MOVL  IAC$GL_IMAGE_LIST, NEXT_ICB        ; 2805
      64 9E 00022 3$:      MOVAB  IAC$GL_IMAGE_LIST, R0                ; 2807
      53 D1 00025           Cmpl  NEXT_ICB, R0                ;
      3E 13 00028           BEQL  6$                ;
      06 E1 0002A           BBC   #6, 16(NEXT_ICB), 4$    ; 2808
      64 A3 D4 0002F           CLRL  100(NEXT_ICB)          ; 2811
      63 D0 00032           MOVL  (NEXT_ICB), NEXT_ICB    ; 2812
      EB 11 00035           BRB   3$                ;
      63 OF 00037 4$:      REMQUE (NEXT_ICB), ICB        ; 2816
      01 E1 0003A           BBC   #1, 17(ICB), 5$        ; 2817
00000000G 00 DD 0003F           PUSHL OWN_STORAGE+100        ; 2821
      7E D4 00045           CLRL  -(SP)                ;
      48 A2 9F 00047           PUSHAB 72(ICB)                ;
      03 FB 0004A           CALLS #3, SYS$DELTV          ;
      OE A2 3C 00051 5$:      MOVZWL 14(ICB), -(SP)        ; 2823

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

**SYSDACT SYSIMACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 68**

**X-14                    ERROR\_CLEAN\_UP - Clean Up after an Error is Det 10-Apr-1989 10:48:52        \$25**

|    |    |       |       |                         |        |
|----|----|-------|-------|-------------------------|--------|
| 01 | FB | 00055 | CALLS | #1, SYSDASSGN           | ;      |
| 62 | D0 | 0005C | MOVL  | (ICB), NEXT_ICB         | ; 2824 |
| 52 | DD | 0005F | PUSHL | ICB                     | ; 2825 |
| 01 | FB | 00061 | CALLS | #1, IMG\$DEALLOCATE_ICB | ;      |
| BA | 11 | 00066 | BRB   | 3\$                     | ; 2807 |
| 01 | 8A | 00068 | BICB2 | #1, IAC\$GL_IMAGCTX+2   | ; 2828 |
|    | 04 | 0006F | RET   |                         | ; 2830 |

**; Routine Size: 112 bytes,        Routine Base: EXEC\$PAGED\_CODE + 0B62**

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 69

```

X-14          GET_OTHER_IMAGE - Open Primary Image File          10-Apr-1989 10:48:52    _$25

; 2832 1      %SBTTL 'GET_OTHER_IMAGE - Open Primary Image File'
; 2833 1
; 2834 1      ROUTINE GET_OTHER_IMAGE (ICB) =
; 2835 1
; 2836 1      !+
; 2837 1      ! Functional Description:
; 2838 1      !
; 2839 1      ! This routine is called when the image passed to the image activator is
; 2840 1      ! really the image that should be activated. This situation occurs in the
; 2841 1      ! case of compatibility mode images and other specialized cases. The gene
; 2842 1      ! operation of this routine is as follows.
; 2843 1      !
; 2844 1      ! Information about the original image is stored in various places so
; 2845 1      ! that it is available to the image that eventually gets activated.
; 2846 1      !
; 2847 1      ! A new image name is selected based on the alias code.
; 2848 1      !
; 2849 1      ! This new image file is opened and its header decoded.
; 2850 1      !
; 2851 1      ! Activation continues with the secondary image replacing the original im
; 2852 1      ! as the target of activation.
; 2853 1      !
; 2854 1      ! Calling Sequence:
; 2855 1      !
; 2856 1      ! GET_OTHER_IMAGE (ICB address)
; 2857 1      !
; 2858 1      ! Input Parameter:
; 2859 1      !
; 2860 1      ! ICB - Address of image control block that describes the primary image
; 2861 1      !
; 2862 1      ! Output Parameters:
; 2863 1      !
; 2864 1      ! none
; 2865 1      !
; 2866 1      ! Implicit Output:
; 2867 1      !
; 2868 1      ! The image name of the primary image is stored in the compatibility mode
; 2869 1      ! data page. The channel on which the primary image is opened (and its KF
; 2870 1      ! address if any) is stored for later return to the caller.
; 2871 1      !
; 2872 1      ! Assumption:
; 2873 1      !
; 2874 1      ! This routine can only be called from the main loop in the image activat
; 2875 1      ! This means that the primary buffers (FAB, NAM, IHD, etc.) describe the
; 2876 1      ! original image.
; 2877 1      !-
; 2878 1
; 2879 2      BEGIN
; 2880 2
; 2881 2      MAP
; 2882 2          ICB                      : REF $BBLOCK;
; 2883 2
; 2884 2      BIND
; 2885 2          IHD_CTX      = .ICB [ICB$!_CONTEXT]      : $BBLOCK,
; 2886 2          ICB_NAME     = ICB [ICB$!_IMAGE_NAME]    : VECTOR [, BYTE],
; 2887 2          FAB         = PRIMARY_FAB              : $BBLOCK,
; 2888 2          NAM         = PRIMARY_NAM               : $BBLOCK,

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 70

```

X-14      GET_OTHER_IMAGE - Open Primary Image File      10-Apr-1989 10:48:52    _$25
; 2889 2          STORED_NAME = CTL$AG_CMEDATA          : VECTOR [, BYTE];
; 2890 2
; 2891 2          ! The image name for a type 2 image is stored in the last 126 bytes of the
; 2892 2          ! first block of the image header as a counted ASCII string.
; 2893 2
; 2894 2          MAP
; 2895 2          INPUT_BUFFER                          : VECTOR [512, BYTE];
; 2896 2
; 2897 2          BIND
; 2898 2          TYPE_2_IMAGE_NAME =
; 2899 2          INPUT_BUFFER [512 - 128]              : VECTOR [128 - 2, BYTE];
; 2900 2
; 2901 2          ! Three of the four cases handled by this routine activate specific images
; 2902 2          ! whose names are listed here. (The fourth case extracts the image name from
; 2903 2          ! the end of the first block of the image header.)
; 2904 2
; 2905 2          BIND
; 2906 2          RSX_NAME = $DESCRIPTOR ('RSX'),
; 2907 2          BPA_NAME = $DESCRIPTOR ('BPA'),
; 2908 2          LOGIN_NAME = $DESCRIPTOR ('LOGINOUT'),
; 2909 2
; 2910 2          ! All four cases use SYS$SYSTEM as the default directory string
; 2911 2
; 2912 2          SYSTEM_NAME = $DESCRIPTOR ('SYS$SYSTEM:.EXE');
; 2913 2
; 2914 2          LOCAL
; 2915 2          NEW_IMAGE_NAME,
; 2916 2          NEW_IMAGE_NAME_DESC                      : $BBLOCK [DSC$K_S_BLN],
; 2917 2          STATUS;
; 2918 2
; 2919 2          OWN_STORAGE [OTHER_CHANNEL] = .ICB [ICB$W_CHAN];
; 2920 2          OWN_STORAGE [OTHER_KFE_ADDRESS] = .FAB [FAB$L_CTX];
; 2921 2
; 2922 2          ! Now perform the steps that are specific to the type of other image that is
; 2923 2          ! being selected. The name of the image to activate is the most important
; 2924 2          ! part of this step.
; 2925 2
; 2926 2          CASE .IHD_CTX [CTX_W_ALIAS] FROM IHD$C_RSX TO IHD$C_CLI OF
; 2927 2
; 2928 2          SET
; 2929 2
; 2930 2          [IHD$C_RSX]:
; 2931 2
; 2932 2          ! This is an image produced by the RSX-11M task builder. Activate
; 2933 2          ! SYS$SYSTEM:RSX.EXE in its stead.
; 2934 2
; 2935 2          NEW_IMAGE_NAME = RSX_NAME;
; 2936 2
; 2937 2          [IHD$C_BPA]:
; 2938 2
; 2939 2          ! There is no supported way that this type of image can be created.
; 2940 2          ! will activate SYS$SYSTEM:BPA.EXE anyway and let the chips fall ...
; 2941 2
; 2942 2          NEW_IMAGE_NAME = BPA_NAME;
; 2943 2
; 2944 2          [IHD$C_ALIAS]:
; 2945 2

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 71

```

X-14      GET_OTHER_IMAGE - Open Primary Image File      10-Apr-1989 10:48:52      _$25
; 2946 2      ! This is a special form of image that contains the name of a second
; 2947 2      ! image in the last 128 bytes of the first block of the image header
; 2948 2      ! (The actual name is restricted to 125 bytes because the last word
; 2949 2      ! of the first block is reserved to contain the code word and the
; 2950 2      ! string contains a count byte.)
; 2951 2
; 2952 3      BEGIN
; 2953 3
; 2954 3      NEW_IMAGE_NAME_DESC [DSC$W_LENGTH] = .TYPE_2_IMAGE_NAME [0];
; 2955 3      NEW_IMAGE_NAME_DESC [DSC$A_POINTER] = TYPE_2_IMAGE_NAME [1];
; 2956 3
; 2957 3      NEW_IMAGE_NAME = NEW_IMAGE_NAME_DESC;
; 2958 3
; 2959 2      END;
; 2960 2
; 2961 2      [IHD$C_CLI]:
; 2962 2
; 2963 2      ! The image is a command language interpreter whose name was passed
; 2964 2      ! the Create Process system service. If this is the activation of a
; 2965 2      ! program (and not a merged activation, the usual way to put a CLI i
; 2966 2      ! P1 space), we will activate SYS$SYSTEM:LOGINOUT.EXE. In this case,
; 2967 2      ! will close the CLI image file first (by deassigning the channel) b
; 2968 2      ! LOGINOUT uses a more restrictive form of $OPEN than occurred above
; 2969 2
; 2970 2      IF .OWN_STORAGE [MAIN_PROGRAM]
; 2971 2      THEN
; 2972 3          BEGIN
; 2973 3
; 2974 3          $DASSGN (CHAN = .ICB [ICB$W_CHAN]);
; 2975 3          NEW_IMAGE_NAME = LOGIN_NAME;
; 2976 3
; 2977 3          END
; 2978 2      ELSE
; 2979 2          RETURN SS$_NORMAL;
; 2980 2
; 2981 2      [OUTRANGE]:
; 2982 2
; 2983 2          RETURN SS$_BADIMGHDR;
; 2984 2
; 2985 2      TES;
; 2986 2
; 2987 2      ! Any context established by the original image must be cleared before
; 2988 2      ! the activation continues.
; 2989 2
; 2990 2      ICB [ICB$L_FLAGS] = 0;          ! Clear previous activation flags
; 2991 2      ICB [ICB$L_IHD] = 0;          ! Clear pointer to resident header
; 2992 2      IHD_CTX [CTX_L_IHDBUF] = PRIMARY_IHD; ! Reestablish IHD buffer
; 2993 2
; 2994 2      ! If the primary image was a CLI (type 3), only the file name is stored. In
; 2995 2      ! all other cases, the entire resultant (or expanded) string is stored.
; 2996 2
; 2997 2      IF .IHD_CTX [CTX_W_ALIAS] EQL IHD$C_CLI
; 2998 2      THEN
; 2999 3          BEGIN
; 3000 3          STORED_NAME [0] = .NAM [NAM$B_NAME];
; 3001 3          CH$MOVE (.STORED_NAME [0], .NAM [NAM$L_NAME], STORED_NAME [1]);
; 3002 3          END

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSSIMGACT SYSSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 72

```

X-14      GET_OTHER_IMAGE - Open Primary Image File      10-Apr-1989 10:48:52      _$25
; 3003 2      ELSE
; 3004 3          BEGIN
; 3005 3          STORED_NAME [0] = .NAM [NAM$B_RSL];
; 3006 3          CH$MOVE (.STORED_NAME [0], .NAM [NAM$L_RSA], STORED_NAME [1]);
; 3007 2          END;
; 3008 2
; 3009 2      ! Open the secondary image file, decode the image header, and resume process
; 3010 2      ! in the main routine as if the secondary image were the one selected for
; 3011 2      ! activation.
; 3012 2
; 3013 2      ICB_NAME [0] = 0;                                ! Force a new name to be stored in I
; 3014 2
; 3015 2      STATUS = IMG$OPEN_IMAGE (                        ! Open the image file
; 3016 2          .NEW_IMAGE_NAME,
; 3017 2          SYSTEM_NAME,
; 3018 2          PRIMARY_FAB,
; 3019 2          PRIMARY_NAM,
; 3020 2          RESULT_NAME,
; 3021 2          .ICB);
; 3022 2      IF NOT .STATUS
; 3023 2      THEN RETURN .STATUS;
; 3024 2
; 3025 2      STATUS = IMG$GET_HEADER (.ICB);                  ! Decode and store away the IHD cont
; 3026 2      RETURN .STATUS;
; 3027 2
; 3028 1      END;

```

```

.PSECT EXEC$PAGED_DATA, PIC,2
58 53 52 00020 P.AAE: .ASCII \RSX\ ;
00023 .BLKB 1 ;
00000003 00024 P.AAD: .LONG 3 ;
00000000' 00028 .ADDRESS P.AAE ;
41 50 42 0002C P.AAG: .ASCII \BPA\ ;
0002F .BLKB 1 ;
00000003 00030 P.AAF: .LONG 3 ;
00000000' 00034 .ADDRESS P.AAG ;
E 49 47 4F 4C 00038 P.AAI: .ASCII \LOGINOUT\ ;
00000008 00040 P.AAH: .LONG 8 ;
00000000' 00044 .ADDRESS P.AAI ;
3 24 53 59 53 00048 P.AAK: .ASCII \SYSS$SYSTEM:.EXE\ ;
00057 .BLKB 1 ;
0000000F 00058 P.AAJ: .LONG 15 ;
00000000' 0005C .ADDRESS P.AAK ;

```

```

RSX_NAME=          P.AAD
BPA_NAME=          P.AAF
LOGIN_NAME=        P.AAH
SYSTEM_NAME=       P.AAJ

```

.PSECT EXEC\$PAGED\_CODE, NOWRT, PIC,2

03FC 00000 GET\_OTHER\_IMAGE:

.WORD Save R2,R3



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSSIMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 73

```

X-14          GET_OTHER_IMAGE - Open Primary Image File          10-Apr-1989 10:48:52          _$25
00000000G 00 9E 00002      MOVAB  STORED_NAME, R9          ;
00000000G 00 9E 00009      MOVAB  OWN_STORAGE+14, R8      ;
          08 C2 00010      SUBL2  #8, SP                  ;
          04 AC D0 00013      MOVL   ICB, R6                 ; 2885
          58 A6 D0 00017      MOVL   88(R6), R2              ;
          0E A6 B0 0001B      MOVW   14(R6), OWN_STORAGE+14  ; 2919
FCAC      C8 D0 0001F      MOVL   FAB+24, OWN_STORAGE+20  ; 2920
          0E A2 AF 00025      CASEW  14(R2), #0, #3          ; 2926
          000D      0002A 1$: .WORD  2$-1$, -          ;
          3$-1$, -          ;
          4$-1$, -          ;
          5$-1$            ;
          44 8F 9A 00032      MOVZBL #68, R0                 ; 2983
          04 00036      RET                            ;
0000'     CF 9E 00037 2$:    MOVAB  RSX_NAME, NEW_IMAGE_NAME  ; 2935
          31 11 0003C      BRB    7$                      ;
0000'     CF 9E 0003E 3$:    MOVAB  BPA_NAME, NEW_IMAGE_NAME  ; 2942
          2A 11 00043      BRB    7$                      ;
F614     C8 9B 00045 4$:    MOVZBW TYPE_2_IMAGE_NAME, NEW_IMAGE_NAME_DESC ; 2954
F615     C8 9E 0004A      MOVAB  TYPE_2_IMAGE_NAME+1, NEW_IMAGE_NAME_DESC+4 ; 2955
          6E 9E 00050      MOVAB  NEW_IMAGE_NAME_DESC, NEW_IMAGE_NAME ; 2957
          1A 11 00053      BRB    7$                      ;
          F2 A8 E9 00055 5$:    BLBC   OWN_STORAGE, 6$         ; 2970
          0E A6 3C 00059      MOVZWL 14(R6), -(SP)           ; 2974
          01 FB 0005D      CALLS  #1, SYSSDASSGN         ;
0000'     CF 9E 00064      MOVAB  LOGIN_NAME, NEW_IMAGE_NAME ; 2975
          04 11 00069      BRB    7$                      ;
          01 D0 0006B 6$:    MOVL   #1, R0                 ; 2979
          04 0006E      RET                            ;
          10 A6 D4 0006F 7$:    CLRL   16(R6)                 ; 2990
          50 A6 D4 00072      CLRL   80(R6)                 ; 2991
F694     C8 9E 00075      MOVAB  PRIMARY_IHD, 4(R2)      ; 2992
          0E A2 B1 0007B      CMPW   14(R2), #3             ; 2997
          0F 12 0007F      BNEQ   8$                      ;
FD1F     C8 90 00081      MOVB   NAM+59, STORED_NAME     ; 3000
          69 9A 00086      MOVZBL STORED_NAME, R1         ; 3001
FD30     C8 D0 00089      MOVL   NAM+76, R0             ;
          0D 11 0008E      BRB    9$                      ;
FCE7     C8 90 00090 8$:    MOVB   NAM+3, STORED_NAME     ; 3005
          69 9A 00095      MOVZBL STORED_NAME, R1         ; 3006
FCE8     C8 D0 00098      MOVL   NAM+4, R0              ;
          51 28 0009D 9$:    MOVVC3 R1, (R0), STORED_NAME+1 ;
          14 A6 94 000A2      CLRB   20(R6)                 ; 3013
          56 DD 000A5      PUSHL  R6                     ; 3015
FDF4     C8 9F 000A7      PUSHAB RESULT_NAME           ; 3020
FCE4     C8 9F 000AB      PUSHAB PRIMARY_NAM           ; 3019
FC94     C8 9F 000AF      PUSHAB PRIMARY_FAB           ; 3018
0000'     CF 9F 000B3      PUSHAB SYSTEM_NAME           ; 3017
          57 DD 000B7      PUSHL  NEW_IMAGE_NAME         ; 3016
          06 FB 000B9      CALLS  #6, IMG$OPEN_IMAGE     ;
          50 E9 000BE      BLBC   STATUS, 10$           ; 3022
          56 DD 000C1      PUSHL  R6                     ; 3025
          01 FB 000C3      CALLS  #1, IMG$GET_HEADER     ;
          04 000C8 10$:      RET                            ; 3026

```

se: EXEC\$PAGED\_CODE + 0BD2

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-  
32 V4.5-862 Page 74  
X-14 GET\_OTHER\_IMAGE - Open Primary Image File 10-Apr-1989 10:48:52 \_\$25

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

32 V4.5-862 Page 75  
 SYS\$IMGACT SYSIMGACT - Image Activator System Service 10-May-1989 16:11:13 VAX Bliss-

```
X-14          GET_OTHER_IMAGE - Open Primary Image File      10-Apr-1989 10:48:52  _$25
;   3030  1
;   3031  1      END                                     ! End of module SYS$IMGACT
;   3032  1
;   3033  0      ELUDOM
```

PSECT SUMMARY

| Name                 | Bytes                | Attributes                |
|----------------------|----------------------|---------------------------|
| EXEC\$INIT_SSTBL_001 | 16 NOVEC, WRT, RD    | EXE,NOSHR, LCL, REL, CON, |
| EXEC\$PAGED_CODE     | 3227 NOVEC,NOWRT, RD | EXE,NOSHR, LCL, REL, CON, |
| EXEC\$PAGED_DATA     | 96 NOVEC, WRT, RD    | EXE,NOSHR, LCL, REL, CON, |

Library Statistics

| File                            | Symbols |                | Pages Mapped | Proc Time |
|---------------------------------|---------|----------------|--------------|-----------|
|                                 | Total   | Loaded Percent |              |           |
| _\$254\$DUA55:[SYSLIB]LIB.L32;1 | 31401   | 183 0          | 1695         | 00        |

COMMAND QUALIFIERS

```
BLISS/CHECK=(FIELD, INITIAL, OPTIMIZE)/LIS=LIS$:SYSIMGACT/OBJ=OBJ$:SYSIMGACT TMP$:SYSI
; Size:          3227 code + 112 data bytes
; Run Time:      00:15.6
; Elapsed Time:  00:22.8
; Lines/CPU Min: 11702
; Lexemes/CPU-Min:115504
; Memory Used:   384 pages
; Compilation Complete
```

## 5 CHECK\_VERSION.LIS

EXE\$CHECK\_VERSION - Check for SYS.STB Version Mismatch 10-MAY-1989 16:53:02 VAX MACRO V5.0-  
8 Page 0

Table of contents

|     |     |   |
|-----|-----|---|
| (2) | 76  | Declarations  |
| (3) | 108 | EXE\$CHECK_VERSION - Check for SYS.STB Version Mismatch |

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

EXE\$CHECK\_VERSION - Check for SYS.STB Version Mismatch 10-MAY-1989 16:53:02 VAX MACRO V5.0-

8 Page 1

X-7 2-MAR-1989 10:15:30 [SYS.SRC]CHECK\_VERSION.MAR;1 (1)

```
1      .TITLE  EXE$CHECK_VERSION - Check for SYS.STB Version Mismatch
2      .IDENT  'X-7'
3
4 ;*****
5 ;*
6 ;*  COPYRIGHT (c) 1986 BY
7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
8 ;*  ALL RIGHTS RESERVED.
9 ;*
10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
11 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
14 ;*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
15 ;*  TRANSFERRED.
16 ;*
17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
18 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
19 ;*  CORPORATION.
20 ;*
21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
23 ;*
24 ;*
25 ;*****
26
27 ;++
28 ; Facility:
29 ;
30 ;       VAX/VMS Executive
31 ;
32 ;       Image Activator and System-Space Code Loader
33 ;
34 ; Abstract:
35 ;
36 ;       The routine in this module is used by any component that needs to
37 ;       check whether an image linked against the system symbol table
38 ;       (SYS.STB) is compatible with the running system system.
39 ;
40 ; Author:
41 ;
42 ;       Lawrence J. Kenah
43 ;
44 ; Creation Date:
45 ;
46 ;       24 October 1986
47 ;
48 ; Modified by:
49 ;
50 ;       X-6,7  SF00006          Stephen Fiorelli          01-Mar-1989
51 ;             Skip checking of base image subversion because
52 ;             it's screwed up. The silly version is not used
53 ;             for anything anyway.
54 ;
55 ;       X-5    SF00005          Stephen Fiorelli          23-Sep-1988
56 ;             Change the checking for the minor version id.
57 ;             First, the base_image minor id, or the overall
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

EXE\$CHECK\_VERSION - Check for SYS.STB Version Mismatch 10-MAY-1989 16:53:02 VAX MACRO V5.0-  
8 Page 2  
X-7 2-MAR-1989 10:15:30 [SYS.SRC]CHECK\_VERSION.MAR;1 (1)

58 ;                   minor id has had its meaning changed. It contains  
59 ;                   a number corresponding to the release it was  
60 ;                   linked against. This number is relative to  
61 ;                   to other releases under active development.  
62 ;                   For example if both X and Y are releases  
63 ;                   under active development, and X is to be  
64 ;                   released before Y, then X's overall minor  
65 ;                   id is less than Y's.  
66 ;  
67 ;           X-4       SFO4001           Stephen Fiorelli           14-Jan-1987  
68 ;                   Remove temporary condition of checking according to  
69 ;                   the old version format.  
70 ;  
71 ;           X-3       LJK0002           Lawrence J. Kenah           29-Oct-1986  
72 ;                   Make version mismatch a warning, not an alternate success.  
73 ;  
74 ;           V05-001 LJK0001           Lawrence J. Kenah           Original

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

EXEC\$CHECK\_VERSION - Check for SYS.STB Version Mismatch 10-MAY-1989 16:53:02 VAX MACRO V5.0-  
8 Page 3  
X-7 Declarations 2-MAR-1989 10:15:30 [SYS.SRC]CHECK\_VERSION.MAR;1 (2)

```
76      .SUBTITLE      Declarations
77
78 ; Include files:
79
80      $IHDEF          ; Offsets into pieces of image header
81      $IHVDEF         ; Layout of version number array elements
82      $$SDEF          ; System service status codes
83      $$STSDEF        ; Status code internal structure
84
85 ; Local symbol definitions
86
87      IHD_POINTER = 4      ; Offset into argument list
88
89 ; Status codes
90
91      SS__SYSVERDIF_WARNING = SS$__SYSVERDIF & STS$M_CODE
92
93 ; External declarations
94
95      .DISABLE      GLOBAL
96
97 ; Global data cells
98
99      .EXTERNAL      -
100         SYS$GL_VERSION
101
102 ; PSECT Declarations:
103
104      .DEFAULT      DISPLACEMENT , WORD
105
106      DECLARE_PSECT EXEC$NONPAGED_CODE
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

EXE\$CHECK\_VERSION - Check for SYS.STB Version Mismatch 10-MAY-1989 16:53:02 VAX MACRO V5.0-  
8 Page 4

X-7 EXE\$CHECK\_VERSION - Check for SYS.STB Ve 2-MAR-1989 10:15:30 [SYS.SRC]CHECK\_VERSION.MAR;

```
108      .SUBTITLE      EXE$CHECK_VERSION - Check for SYS.STB Version Mismatch
109 ;+
110 ; Functional Description:
111 ;
112 ;      This routine insures that an image (loadable or executable) that is
113 ;      linked against SYS.STB did not use a later set of routines than is
114 ;      currently running.
115 ;
116 ;      These checks are used by the various loading mechanisms and by the
117 ;      image activator when an image linked against SYS.STB is activated into
118 ;      a privileged environment.
119 ;
120 ; Input Parameter:
121 ;
122 ;      4(AP) - Address of buffer containing the image header
123 ;
124 ; Implicit Input:
125 ;
126 ;      SYS$GL_VERSION - Address of a 33-longword array of version numbers
127 ;
128 ;              [0] - contains SYS$K_VERSION
129 ;              [1] - contains SYS$K_VERSION_01
130 ;              [2] - contains SYS$K_VERSION_02
131 ;              .
132 ;              .
133 ;              [32] - contains SYS$K_VERSION_32
134 ;
135 ; Output Parameters:
136 ;
137 ;      none
138 ;
139 ; Status:
140 ;
141 ;      R0 low bit set indicates success. The image can be loaded or
142 ;      activated with no concern for version mismatch.
143 ;
144 ;      R0 low bit clear indicates a failure.
145 ;
146 ;      SS$_VERSION_MISMATCH (spelling?)
147 ;
148 ;      Indicates that the image was linked against a SYS.STB that
149 ;      is incompatible with the running system.
150 ;
151 ;      SS$_BADIMGHDR
152 ;
153 ;      Indicates a problem with the image header (or with the
154 ;      SYS.STB used in the link). This error is returned when
155 ;      IHD$$_VERSION contains a zero but IHD$$_VERSION_ARRAY_OFF
156 ;      is nonzero. In a proper environment, this cannot happen.
157 ;
158 ; Notes:
159 ;
160 ;      There is no match control used by this routine. The semantics are
161 ;      simple. The major ID of the image must exactly match the major ID of
162 ;      the running system. The minor ID of the image must be less than or
163 ;      equal to (LEQU) the ID of the running system.
164 ;
```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

EXE\$CHECK\_VERSION - Check for SYS.STB Version Mismatch 10-MAY-1989 16:53:02 VAX MACRO V5.0-

8 Page 5

X-7 EXE\$CHECK\_VERSION - Check for SYS.STB Ve 2-MAR-1989 10:15:30 [SYS.SRC]CHECK\_VERSION.MAR;

```

165 ;      Thus, the minor ID semantics allow upward compatible growth while the
166 ;      major ID provides for incompatible changes in internal interfaces.
167 ;
168 ;      The overall version number differs from the 32 subsystem version
169 ;      numbers in one small detail. Its major ID is 8 bits wide (and is likely
170 ;      never to change), leaving 24 bits for the minor ID. The subsystem IDs
171 ;      are exist as 16 bits of major ID and 16 bits of minor ID.
172 ; -
173
174      UNIVERSAL_ENTRY EXE$CHECK_VERSION, <^M<R2,R3,R4,R5>>
175
176      MOVL   IHD_POINTER(AP), R4           ; R4 points to image header buffer
177      MOVAL  G^SYS$GL_VERSION, R5         ; R5 locates version array in memory
178
179      MOVZWL IHD$W_VERSION_ARRAY_OFF(R4), R1 ; R1 contains byte offset to version
180
181      MOVL   IHD$L_SYSVER(R4), R3         ; Get overall version number from IH
182      BNEQ   20$                          ; Usual case is nonzero
183
184      TSTL   R1                             ; Is offset also zero?
185      BEQL   10$                          ; Branch if yes
186
187 ; If the array offset is nonzero but the overall version number is zero,
188 ; this indicates an image that is internally inconsistent.
189
190      MOVZWL #SS$_BADIMGHDR, R0
191      RET
192
193 ; We should not even be in this procedure if both version indicators are
194 ; zero but will excuse sloppy callers by simply returning success.
195
196 10$:   MOVZWL #SS$_NORMAL, R0
197      RET
198
199 ; The overall IHD version number is nonzero. Compare this to the
200 ; system version number.
201
202 ;      R2 - scratch
203 ;      R3 - overall version number from image header
204 ;      R5 - address of SYS$GL_VERSION (advanced to point to first subversion number
205 ;
206 ;      R1 and R4 are not used for this step
207
208
209      ASSUME IHVN$V_VERSION_MAJOR_ID EQ 24
210      ASSUME IHVN$$_VERSION_MAJOR_ID EQ 8
211
212 20$:   EXTZV #IHVN$V_VERSION_MAJOR_ID,-
213         #IHVN$$_VERSION_MAJOR_ID,-
214         R3, R2                             ; R2 gets IHD major ID
215      CMPB   R2, 3(R5)                       ; Compare with system major ID
216      BNEQ   90$                             ; Report error if mismatch
217
218
219 ; If the overall version number is acceptable and there is no additional
220 ; version number information, simply return success.
221

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

EXE\$CHECK\_VERSION - Check for SYS.STB Version Mismatch 10-MAY-1989 16:53:02 VAX MACRO V5.0-

8 Page 6

X-7 EXE\$CHECK\_VERSION - Check for SYS.STB Ve 2-MAR-1989 10:15:30 [SYS.SRC]CHECK\_VERSION.MAR;

```

222         TSTL     R1                ; Is offset also zero?
223         BEQL     10$              ; Branch if yes
224
225 ; The overall version numbers are compatible. Now make detailed checks on
226 ; all component version numbers that are relevant to this image.
227
228         ASSUME   IHVN$L_SUBSYSTEM_MASK EQ 0
229         ASSUME   IHVN$L_SUBVERSION_ARRAY EQ 4
230
231         ADDL     R1, R4              ; R4 now points to IHD version array
232         MOVL     (R4)+, R1          ; R1 contains version mask
233
234         ASSUME   IHVN$V_VERSION_MINOR_ID EQ 0
235         ASSUME   IHVN$$_VERSION_MINOR_ID EQ 24
236
237         EXTZV    #IHVN$V_VERSION_MINOR_ID,- ; Store overall minor id from
238                 #IHVN$$_VERSION_MINOR_ID,- ; IHD in R3
239                 R3, R3
240
241 ; Perform a subversion number check for each nonzero entry in the IHD array mask.
242
243 ;      R0 - loop index
244 ;      R1 - version mask
245 ;      R2 - scratch (points to next significant system version number)
246 ;      R3 - overall version number from image header
247 ;      R4 - steps through version array in image header
248 ;      R5 - points to base of system version number array
249
250         ASSUME   IHVN$V_SUBVERSION_MINOR_ID EQ 0
251         ASSUME   IHVN$$_SUBVERSION_MINOR_ID EQ 16
252
253         ASSUME   IHVN$V_SUBVERSION_MAJOR_ID EQ 16
254         ASSUME   IHVN$$_SUBVERSION_MAJOR_ID EQ 16
255
256         CLRL     R0                ; Start with LSB (VERSION_01)
257 50$:     BBC      R0, R1, 60$      ; Skip checks if bit not set
258         TSTL     R0                ; Are we testing the base image vers
259         BEQL     70$              ; Yes, skip checking
260         MOVAL    4(R5)[R0], R2     ; R2 points to next significant vers
261         CMPW     (R2)+, (R4)+      ; Compare minor IDs
262         BLSSU    90$              ; Error if IHD version GTRU system v
263         BEQLU    55$              ; If equal, no need to check overall
264 ;
265 ;      If we get here the minor id from the image header is less than the
266 ;      minor id of the running system. We now check the release numbers
267 ;      which is the overall minor id. If the release the image was linked
268 ;      against is greater than the release of the running system, then
269 ;      there is a version mismatch. We cannot trust the minor ids in this
270 ;      case (unless they are equal). This is a conservative
271 ;      approach. We sometimes issue version mismatch when it might
272 ;      not be necessary, but never issue a success unless we really
273 ;      mean it. This conservative approach gets corrected, and normal
274 ;      version checking for minor ids occurs when the release the image
275 ;      linked against gets rebuilt.
276 ;
277
278         CMPZV    #IHVN$V_VERSION_MINOR_ID,-

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

EXE\$CHECK\_VERSION - Check for SYS.STB Version Mismatch 10-MAY-1989 16:53:02 VAX MACRO V5.0-  
8 Page 7

X-7 EXE\$CHECK\_VERSION - Check for SYS.STB V# 2-MAR-1989 10:15:30 [SYS.SRC]CHECK\_VERSION.MAR;

```
279          #IHVN$$_VERSION_MINOR_ID,-      ; Compare system minor ID with
280          (R5), R3                          ; minor ID from IHD
281      BLSSU  90$                             ; Error if overall minor ID is large
282
283 55$:      CMPW  (R2)+, (R4)+                ; Now compare major IDs
284          BNEQ  90$                          ; Major ID requires strict equality
285 60$:      AOBLEQ #31, R0, 50$              ; Loop through all 32 mask bits
286
287 ; If we drop through the loop, all component specific version numbers
288 ; in the image are compatible with the running system.
289
290          MOVZWL #SS$_NORMAL, R0
291          RET
292 ; If we get here if we were trying to test the base image version number.
293 ; Advance the version number array in image header
294
295 70$:      TSTL  (R4)+
296          BRB   60$
297 ; Some incompatibility was discovered between the image header and
298 ; the running system. Indicate a failure.
299
300 90$:      MOVZWL #SS$_SYSVERDIF_WARNING, R0 ; Return warning status
301          RET
302
303          .END
```

## 6 PERMANENT\_DEVICE\_DATABASE.LIS

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX

8 Page 0

Table of contents

|      |     |  |
|------|-----|--|
| (2)  | 138 | Definition MACROS                                  |
| (3)  | 176 | VAX/VMS SYSTEM PERMANENT DEVICE DATABASE           |
| (4)  | 250 | Initialization of system permanent device database |
| (6)  | 415 | SYSTEM BOOT DEVICE DATABASE                        |
| (7)  | 454 | SYSTEM CONSOLE DEVICE DATABASE                     |
| (8)  | 731 | SYSTEM PERMANENT MAILBOX DATABASE                  |
| (9)  | 879 | NULL DEVICE (NLA) DATABASE                         |
| (10) | 912 | NETWORK DEVICE DATABASE                            |

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX

8 Page 1

X-8 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.MAR;1 (1)

```
1 .TITLE PERMANENT_DEVICE_DATABASE - Miscellaneous system area - base image
2 .IDENT 'X-8'
3
4 ;*****
5 ;*
6 ;* COPYRIGHT (c) 1987 BY
7 ;* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
8 ;* ALL RIGHTS RESERVED.
9 ;*
10 ;* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
11 ;* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
12 ;* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
13 ;* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
14 ;* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
15 ;* TRANSFERRED.
16 ;*
17 ;* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
18 ;* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
19 ;* CORPORATION.
20 ;*
21 ;* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
22 ;* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
23 ;*
24 ;*
25 ;*****
26 ;++
27 ; Facility:
28 ;
29 ; Exec loaded_image SYSTEM_PRIMITIVES
30 ;
31 ; Abstract:
32 ;
33 ; This module contains the system permanent device database
34 ;
35 ; Author:
36 ;
37 ; VMS development team
38 ;
39 ; Creation Date:
40 ;
41 ; April 2, 1987
42 ;
43 ; Modified by:
44 ;
45 ; X-8 DDP0315 Derrell D. Piper 18-Sep-1988
46 ; Add a permanent mailbox for the audit server.
47 ;
48 ; X-7 FAK0002 Forrest A. Kenney 04-Feb-1988
49 ; Make OPA0 UCB use IOLOCK8 forklock, other changes
50 ; allows OPA0 to use this forklock again.
51 ;
52 ; X-6 RNG5006 Rod N. Gamache 7-Sep-1987
53 ; Change IDB$L_CRB to IDB$L_SPL.
54 ;
55 ; X-5 RNG5005 Rod N. Gamache 27-May-1987
56 ; Make OPA0 UCB use IOLOCK9 forklock.
57 ;
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX

8 Page 2

X-8 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.MAR;1 (1)

58 ; X-4 FAK0001 Forrest A. Kenney 22-May-1987  
59 ; Change OPA\$VECTOR to use \$VECINI and \$VEC macro to build  
60 ; port vector table. This will insure that the vector is  
61 ; always the correct size.  
62 ;  
63 ; X-3 MSH0313 Michael S. Harvey 15-May-1987  
64 ; Add RLS\_PC longword to permanent spinlock structure.  
65 ;  
66 ; X-2 SF00002 Stephen Fiorelli 14-Apr-1987  
67 ; Initialize tty\$gl\_jobctlmb with the job controller's  
68 ; mailbox.  
69 ;  
70 ; X-1 SF00001 Stephen Fiorelli 02-Apr-1987  
71 ; Part of system\_misc.mar contain the system device  
72 ; database was moved here. Modification history of  
73 ; the database while it was part of system\_misc is  
74 ; retained.  
75 ;  
76 ; X-OBS MJW0113 Michael J. Worcester 26-Feb-1987  
77 ; Change reference of DYN\$C\_SPL\_DEVICELOCK to  
78 ; SPL\$C\_SPL\_DEVICELOCK to agree with new SPL definition.  
79 ;  
80 ; X-OBS MSH0246 Michael S. Harvey 10-Feb-1987  
81 ; Remove obsolete spinlock counter cell.  
82 ;  
83 ; X-OBS MAS0025 Mary A. Sullivan 26-JAN-1987  
84 ; Fix ASSUMES and add space for VEC\$W\_NUMALT, VEC\$W\_MAPALT  
85 ; in OPA\$CRB  
86 ;  
87 ; X-OBS SF04008 Stephen Fiorelli 22-Jan-1987  
88 ; Remove the label IOC\$GL\_DEVLIST from within the local  
89 ; scs system block. The DDB listhead will still remain,  
90 ; but a second listhead will be created with the movement  
91 ; of IOC\$GL\_DEVLIST to the system\_data\_cells.mar module.  
92 ; Both will point to the next element in the listhead;  
93 ; the system boot device.  
94 ;  
95 ; X-OBS JCK Jonathan C. Kaplan 8-Dec-1986  
96 ; added longword field to OPA\$SPL that is now is the  
97 ; SPLDEF structure  
98 ;  
99 ; X-OBS JCK Jonathan C. Kaplan 7-Dec-1986  
100 ; VECDEF has grown by 16 bytes and assumptions made  
101 ; about CRB are no longer true -- fix assumes no  
102 ; absolute dependence on CRB/VEC size in OPA\$CRB or  
103 ; SYS\_CRB.  
104 ;  
105 ; X-OBS PLL P. Levesque 25-Nov-1986  
106 ; Fix alignment of SYS\$GL\_JOBCTLMB.  
107 ;  
108 ; X-OBS DBM DB Mills 24-Nov-1986  
109 ; Changed ORB, DDB, and UCB macros to align data  
110 ; structures on quadword boundaries. The file system  
111 ; expects the boot UCB to be quadword aligned and uses  
112 ; the low order 3 bits for the cache type.  
113 ;  
114 ; X-OBS BJT0050 Benjamin J. Thomas III 30-Oct-1986

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX  
8 Page 3  
X-8 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.MAR;1 (1)

```
115 ;           Allocate more space at end of boot ucb
116 ;
117 ;   X-OBS   RNG0409           Rod Gamache           24-Jun-1986
118 ;           Add CRB$$_WQFL/WQBL as separate longwords.
119 ;
120 ;   X-OBS   MSH0256           Michael S. Harvey       20-Jun-1986
121 ;           Prevent clobbering the console terminal's device
122 ;           affinity mask.
123 ;
124 ;   X-OBS   MSH0252           Michael S. Harvey       16-Jun-1986
125 ;           Add standard data structure cells to spinlock control
126 ;           block.
127 ;
128 ;   X-OBS   MSH0232           Michael S. Harvey       4-Mar-1986
129 ;           Correct the UCB count in the console IDB to accurately
130 ;           reflect the number of potential UCBs for which there is
131 ;           space.
132 ;
133 ;   X-OBS   RNH0037           Richard N. Holstein     03-Jan-1986
134 ;           Initialize ORB$$_OWNER via DPT_STORE for OP devices.
135 ;
136 ;--
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX

8 Page 4

X-8 Definition MACROS 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.MAR;1 (2)

```
138      .SUBTITLE      Definition MACROS
139
140
141      $ARBDEF          ; ACCESS RIGHTS BLOCK DEFINITIONS
142      $CRBDEF          ; DEFINE CRB
143      $DCDEF           ; DEFINE DEVICE CLASSES
144      $DDBDEF          ; DEFINE DDB
145      $DEVDEF          ; DEFINE DEVICE CHARACTERISTICS
146      $DYNDEF          ; STRUCTURE TYPE CODE DEFINITIONS
147      $FKBDEF          ; DEFINE FORK BLOCK OFFSETS
148      $IDBDEF          ; DEFINE IDB
149      $IPLDEF          ; DEFINE IPL LEVELS
150      $IRPDEF          ; DEFINE IRP OFFSETS
151      $JIBDEF          ; JOB INFORMATION BLOCK DEFINITIONS
152      $ORBDEF          ; OBJECT'S RIGHTS BLOCK OFFSETS
153      $$SPLCODDEF      ; DEFINE SPINLOCK INDICES
154      $$SPLDEF         ; DEFINE SPINLOCK STRUCTURES
155      $TTYMACS         ; THE TERMINAL DRIVER MACRO DEFINITIONS
156      $TTYVECDDEF      ; TERMINAL DRIVER PORT & CLASS VECTOR OFFSETS
157
158 ;      Definiton MACROS need for assembly
159
160      $SBDEF           ;PSTE/GSTE definitions
161      $SECDEF          ;Define state numbers
162      $STATEDEF        ;Define TQE offsets
163      $TQEDDEF         ;Define terminal characteristics
164      $TTDEF           ;Define terminal characteristics
165      $TTYDEF          ;Define terminal characteristics
166      $TTYMACS         ;The terminal driver MACRO definitions
167      $UCBDEF          ;Define UCB
168      $TTYDEF          ;TTY UCB extension (must follow $UCBDEF)
169      $VADEF           ;Define virtual address fields
170      $VECDEF          ;Define CRB vector
171      $WCBDEF          ;Define WCB
172      $WSLDEF          ;Working set list definitions
173
174      INTSTK=1
```



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX  
8 Page 5  
X-8 VAX/VMS SYSTEM PERMANENT DEVICE DATABASE 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.

```

176          .SBTTL  VAX/VMS SYSTEM PERMANENT DEVICE DATABASE
177 ;
178 ; LOCAL MACROS
179 ;
180          .MACRO  ORB      LABEL, ?EN, ?ACL
181          .ALIGN  QUAD
182 ORBASE=.
183 LABEL:
184          .LONG   0
185          .WORD   -1,0                ; ACL MUTEX INITIALIZATION
186          .WORD   EN-LABEL
187          .BYTE   DYN$C_ORB
188          .BYTE   ORB$M_PROT_16
189          .BLKB   ORB$K_LENGTH - <. - LABEL>
190 EN:
191          .ENDM
192 ;
193          .MACRO  STO_ORB OFFSET, SIZE, VALUE
194          X=.
195          .=-ORBASE+ORB$'OFFSET
196          .' SIZE ' VALUE
197          .=-X
198          .ENDM
199 ;
200          .MACRO  UCB      LABEL, EXPAND=0, ORB_ADDR, ?IOL, ?EN, ?ACL
201          .ALIGN  QUAD
202 UCBASE=.
203 LABEL:
204          .LONG   0,0
205          .WORD   EN-LABEL
206          .BYTE   DYN$C_UCB
207          .BYTE   0
208          .BLKB   UCB$L_ORB - <. - LABEL>
209          .ADDRESS ORB_ADDR
210          .BLKB   UCB$L_IOQFL - <. - LABEL>
211 IOL:
212          .ADDRESS IOL, IOL
213          .BLKB   UCB$K_LENGTH - <. - LABEL>
214          .BLKL   EXPAND
215 EN:
216          .ENDM
217          .MACRO  STO_UCB OFFSET, SIZE, VALUE
218          X=.
219          .=-UCBASE+UCB$'OFFSET
220          .' SIZE ' VALUE
221          .=-X
222          .ENDM
223
224          ASSUME  DDB$L_LINK      EQ 0
225          ASSUME  DDB$L_UCB      EQ 4
226          ASSUME  DDB$W_SIZE     EQ 8
227          ASSUME  DDB$B_TYPE     EQ 10
228          ASSUME  DDB$L_DDT      EQ 12
229          ASSUME  DDB$L_ACPD     EQ 16
230
231          .MACRO  DDB NAME, NEXT, FUCB, DDT, ACP, ATYPE, DEVNAM, DRVNAM, ?EN
232          .ALIGN  QUAD

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX  
8 Page 6

X-8 VAX/VMS SYSTEM PERMANENT DEVICE DATABASE 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.

233 NAME: .ADDRESS NEXT  
234 .ADDRESS FUCB  
235 .WORD EN-' NAME  
236 .BYTE DYN\$C\_DDB, 0  
237 .ADDRESS DDT  
238 .LONG ^A/' ACP/+' ATYPE@24>  
239 .ASCIC /' DEVNAM/  
240 .=' NAME+DDB\$T\_DRVNAME  
241 .ASCIC /' DRVNAM/  
242 .=' NAME+DDB\$L\_SB  
243 .ADDRESS SCSS\$GA\_LOCALSB  
244 .=' NAME+DDB\$C\_LENGTH  
245 EN:  
246 .ENDM  
247  
248

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX  
8 Page 7

X-8 Initialization of system permanent devic 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.

```
250      .SBTTL Initialization of system permanent device database
251 ;++
252 ;
253 ; INI$DEVICE_DATABASE
254 ;
255 ; FUNCTIONAL DESCRIPTION:
256 ;
257 ;      This initializes the system permanent device database
258 ;      data structures.
259 ;
260 ; CALLING SEQUENCE:
261 ;
262 ;      JSB, called from INIT.
263 ;
264 ; INPUT PARAMETERS:
265 ;
266 ;      None.
267 ;
268 ; IMPLICIT INPUTS:
269 ;
270 ;      None.
271 ;
272 ; OUTPUT PARAMETERS:
273 ;
274 ;      None.
275 ;
276 ; IMPLICIT OUTPUTS:
277 ;
278 ;      Address of SCS$GA_LOCALSB stuffed in SCS$AR_LOCALSB,
279 ;      also set in the first two long words of SCS$GQ_CONFIG.
280 ;
281 ;      Pointers to the various data structures which live
282 ;      in the base image, sys.
283 ;
284 ; ROUTINE VALUE:
285 ;
286 ;      SS$_NORMAL
287 ;
288 ; SIDE EFFECTS:
289 ;
290 ;      None.
291 ;--
292 ;
293 ;+
294 ;
295 ;+
296 ;
297      DECLARE_PSECT EXEC$INIT_CODE
298 ;
299      INITIALIZATION_ROUTINE INI$DEVICE_DATABASE
300 ;
301 INI$DEVICE_DATABASE::
302 ;
303 ;
304 ;      Initialize the local system block
305 ;
306 ;
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX

8 Page 8

X-8 Initialization of system permanent devic 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.

```
307      MOVAL   SCS$GA_LOCALSB,G^SCS$AR_LOCALSB
308      MOVAL   SCS$GA_LOCALSB,G^SCS$GQ_CONFIG
309      MOVAL   SCS$GA_LOCALSB,G^SCS$GQ_CONFIG+4
310
311 ;
312 ;      Initialize device data structure pointers in the base image
313 ;
314
315      MOVAL   SYS$GL_BOOTDDB,G^SYS$AR_BOOTDDB
316      MOVAL   SYS$GL_BOOTDDB,G^IOC$GL_DEVLIST
317      MOVAL   SYS$GL_BOOTORB,G^SYS$AR_BOOTORB
318      MOVAL   SYS$GL_BOOTUCB,G^SYS$AR_BOOTUCB
319      MOVAL   OPA$GL_DDB,G^OPA$AR_DDB
320      MOVAL   OP$DPT,G^OP$AR_DPT
321      MOVAL   OPA$ORB0,G^OPA$AR_ORB0
322      MOVAL   OPA$UCB0,G^OPA$AR_UCB0
323      MOVAL   OPA$CRB,G^OPA$AR_CRB
324      MOVAL   OPA$IDB,G^OPA$AR_IDB
325      MOVAL   OPA$SPL,G^OPA$AR_SPL
326      MOVAL   MB$GL_DDB,G^MB$AR_DDB
327      MOVAL   MB$ORB0,G^MB$AR_ORB0
328      MOVAL   MB$GL_ORB1,G^MB$AR_ORB1
329      MOVAL   MB$GL_ORB2,G^MB$AR_ORB2
330      MOVAL   MB$GL_ORB3,G^MB$AR_ORB3
331      MOVAL   MB$UCB0,G^MB$AR_UCB0
332      MOVAL   MB$GL_UCB1,G^MB$AR_UCB1
333      MOVAL   MB$GL_UCB2,G^MB$AR_UCB2
334      MOVAL   MB$GL_UCB3,G^MB$AR_UCB3
335      MOVAL   SYS$GL_OPRMBX,G^SYS$AR_OPRMBX
336      MOVAL   SYS$GL_AUDSRVMBX,G^SYS$AR_AUDSRVMBX
337      MOVAL   NL$GL_DDB,G^NL$AR_DDB
338      MOVAL   NL$GL_ORB0,G^NL$AR_ORB0
339      MOVAL   NL$GL_UCB0,G^NL$AR_UCB0
340      MOVAL   OPA$VECTOR,G^OPA$AR_VECTOR
341      MOVAL   SYS$GL_JOBCTLMB,G^SYS$AR_JOBCTLMB
342      MOVAL   SYS$GL_JOBCTLMB,G^TTY$GL_JOBCTLMB
343
344 ;
345 ;      Store clone mailbox ucb size
346 ;
347      MOVW    #SYS$C_MBXUCBSIZ,G^SYS$GW_MBXUCBSIZ
348
349 ;
350 ;      Initialize SCB vectors
351 ;
352
353      MOVL    G^EXE$GL_SCB,R1
354      MOVAL   CON$INTDISI+INTSTK,^XF8(R1)
355      MOVAL   CON$INTDISO+INTSTK,^XFC(R1)
356      MOVL    #SS$NORMAL,R0
357      RSB
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX

8 Page 9

X-8 Initialization of system permanent devic 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.

```

359     DECLARE_PSECT   EXEC$NONPAGED_DATA,ALIGNMENT=QUAD
360 ;
361 ; LOCAL SB
362 ;
363 SCSS$GA_LOCALSB:
364     .LONG   SCS$GQ_CONFIG           ; SB$L_FLINK
365     .LONG   SCS$GQ_CONFIG           ; SB$L_BLINK
366     .WORD   SB$K_LENGTH             ; SB$W_SIZE
367     .BYTE   DYN$C_SCS               ; SB$B_TYPE
368     .BYTE   DYN$C_SCS_SB           ; SB$B_SUBTYP
369     ASSUME  SB$L_PBFL EQ 12
370 10$: .ADDRESS 10$                 ; SB$L_PBFL
371     .ADDRESS 10$                   ; SB$L_PBBL
372     ASSUME  SB$L_PBCONNX EQ 20
373     .LONG   0                       ; SB$L_PBCONNX
374     ASSUME  SB$B_SYSTEMID EQ 24
375     .QUAD   0                       ; SB$B_SYSTEMID
376     ASSUME  SB$W_MAXDG EQ 32
377     .LONG   0                       ; SB$W_MAXDG
378     .LONG   0                       ; SB$W_MAXMSG
379     ASSUME  SB$T_SWTYPE EQ 36
380     .LONG   0                       ; SB$T_SWTYPE
381     ASSUME  SB$T_SWVERS EQ 40
382     .LONG   0                       ; SB$T_SWVERS
383     ASSUME  SB$Q_SWINCARN EQ 44
384     .QUAD   0                       ; SB$Q_SWINCARN
385     ASSUME  SB$T_HWTYPE EQ 52
386     .LONG   0                       ; SB$T_HWTYPE
387     ASSUME  SB$B_HWVERS EQ 56
388     .BLKB   12                       ; SB$B_HWVERS
389     ASSUME  SB$T_NODENAME EQ 68
390     .BLKB   16                       ; SB$T_NODENAME
391     ASSUME  SB$L_DDB EQ 84
392
393 ;
394 ;
395 ; Note: The listhead ioc$gl_devlist use to to be part
396 ; of the local sb, located at the following cell. It
397 ; is now separated out of the local sb, and is located
398 ; in the base image sys.exe.
399 ;
400     .ADDRESS SYS$G_L_BOOTDDB         ; START DEVICE LIST AT BOOT DEVICE
401     .ADDRESS SYS$G_L_BOOTDDB         ; SB$L_DDB
402
403     ASSUME  SB$W_TIMEOUT EQ 88
404     .WORD   -2                       ; LOOKS LIKE TIMEOUT IS IN PROGRESS
405     ASSUME  SB$B_ENBMSK EQ 90
406     .BLKB   SB$S_ENBMSK             ; PROCESS POLL ENABLE MASK
407
408     ASSUME  SB$L_CSB EQ 92
409     .LONG   0                       ; SB$L_CSB
410
411     .BLKB   SB$K_LENGTH-<.-SCSS$GA_LOCALSB> ; SPACE FOR REMAINING FIELDS
412
413     .BLKB   8                       ; Future expansion

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX  
8 Page 10

X-8 SYSTEM BOOT DEVICE DATABASE 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.MAR;1 (6)

```
415          .SBTTL  SYSTEM BOOT DEVICE DATABASE
416 ;
417 ; BOOT DEVICE DDB
418 ;
419          DDB      SYS$GL_BOOTDDB, OPA$GL_DDB, SYS$GL_BOOTUCB, 0, <F11>, 1
420 ;
421 ; UCB FOR SYSTEM BOOT DEVICE
422 ;
423
424 ;
425 ; NOTE - THE UCB FOR THE BOOT DEVICE IS CREATED WITH A REFERENCE COUNT OF 1
426 ; TO AVOID HAVING THE FIRST $ASSIGN TRY TO TAKE OUT A LOCK ON IT BEFORE
427 ; LOCKING IS ENABLED.
428 ;
429 ; THE UCB DATA STRUCTURE MUST BE QUAD WORD ALIGNED.  THE XQP USES THE LOW
430 ; ORDER THREE BITS AS A CACHE TYPE.
431 ;
432          ORB      SYS$GL_BOOTORB
433          STO_ORB  L_OWNER, LONG, <^X010001>
434          UCB      SYS$GL_BOOTUCB, 80, SYS$GL_BOOTORB
435          STO_UCB  B_FLCK, BYTE, SPL$C_IOLOCKS
436          STO_UCB  B_DIPL, BYTE, 21
437          STO_UCB  L_DDB, ADDRESS, SYS$GL_BOOTDDB
438          STO_UCB  L_DEVCHAR, LONG, <<DEV$M_FOD!-
439                  DEV$M_DIR!-
440                  DEV$M_AVL!-
441                  DEV$M_ELG!-
442                  DEV$M_SHR!-
443                  DEV$M_IDV!-
444                  DEV$M_ODV!-
445                  DEV$M_RND>>
446          STO_UCB  B_DEVCLASS, BYTE, DC$_DISK
447          STO_UCB  L_AFFINITY, LONG, -1
448          STO_UCB  W_DEVBUSIZ, WORD, 512
449          STO_UCB  B_ERTCNT, BYTE, 8
450          STO_UCB  B_ERTMAX, BYTE, 8
451          STO_UCB  W_REFC, WORD, 1
452
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX  
8 Page 11

X-8 SYSTEM CONSOLE DEVICE DATABASE 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.MAR;1 (7)

```

454      .SBTTL  SYSTEM CONSOLE DEVICE DATABASE
455 ;
456 ; CONSOLE TERMINAL DDB
457 ;
458      DDB      OPA$GL_DDB,MR$GL_DDB,OPA$UCB0,0,,0,<OPA>,<OPERATOR>
459 ;
460 ; CONSOLE DPT
461 ;
462 ;
463 ; THE UCB SIZE INCLUDES 3 BYTES FOR ROUNDUP, AND 64 BYTES OF EXTRA
464 ; SPACE TO ALLOW INCREASING UCB SIZE WITHOUT NEEDING TO BUILD A NEW
465 ; SYS.
466 ;
467      .PSECT  $$$105_prologue,rd,wrt,noexe,pic,byte
468 OP$DPT:
469      DECLARE_PSECT  EXEC$NONPAGED_DATA
470      DPTAB  -
471          END=OP_DPTEND,-
472          ADAPTER=UBA,-          ;FAKE ADAPTER
473          UCBSIZE=<<<UCB$C_TT_LENGTH+3+64>/4>*4>,-
474          NAME=OPERATOR,-
475          VECTOR=OPA$VECTOR
476
477      DPT_STORE INIT
478      DPT_STORE UCB,UCB$L_TT_DECHAR,@L,TTY$GL_DEFCHAR ; DEFAULT CHARACTERISTICS
479      DPT_STORE UCB,UCB$L_TT_DECHAR1,@L,TTY$GL_DEFCHAR2
480      DPT_STORE UCB,UCB$L_DEVDEPEND,@L,TTY$GL_DEFCHAR
481      DPT_STORE UCB,UCB$L_TT_DEVP1,@L,TTY$GL_DEFCHAR2
482      DPT_STORE UCB,UCB$B_FLCK,B,SPL$C_IOLOCKS          ; FORK LOCK INDEX
483      DPT_STORE UCB,UCB$B_DIPL,B,20                      ; DEVICE IPL
484      DPT_STORE UCB,UCB$L_DEVCHAR,L,<-; CHARACTERISTICS
485          DEV$M_REC!-          ;
486          DEV$M_AVL!-          ;
487          DEV$M_IDV!-          ;
488          DEV$M_ODV!-          ;
489          DEV$M_TRM!-          ;
490          DEV$M_CCL>
491      DPT_STORE UCB,UCB$L_DEVCHAR2,L,-; DEVICE CHARACTERISTICS
492          <DEV$M_NNM>          ; PREFIX WITH NODE$
493      DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$TERM
494      DPT_STORE UCB,UCB$B_DEVTYPE,B,TT$UNKNOWN          ; TYPE
495      DPT_STORE UCB,UCB$W_DEVBUFSIZ,W,132              ; BUFFER SIZE
496      DPT_STORE UCB,UCB$W_STS,W,UCB$M_ONLINE          ; Device comes up online
497      DPT_STORE UCB,UCB$W_TT_DESIZE,W,132            ; BUFFER SIZE
498      DPT_STORE UCB,UCB$W_TT_SPEED,W,TT$C_BAUD_300    ; DEFAULT SPEED
499      DPT_STORE UCB,UCB$B_TT_DETYPE,B,TT$LA36         ; TYPE
500      DPT_STORE UCB,UCB$W_TT_DESPEE,W,TT$C_BAUD_300  ;
501      DPT_STORE ORB,ORB$B_FLAGS,B,-
502          <ORB$M_PROT_16>          ; SOGW protection word
503      DPT_STORE ORB,ORB$W_PROT,@W,TTY$GW_PROT         ; Default protection
504      DPT_STORE ORB,ORB$L_OWNER,@L,TTY$GL_OWNUIC      ; Default owner UIC
505      DPT_STORE UCB,UCB$W_STS,W,UCB$M_ONLINE         ; Device comes up online
506
507      DPT_STORE REINIT          ; Is this needed?
508      DPT_STORE END
509
510 OP_DPTEND:

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX  
8 Page 12  
X-8 SYSTEM CONSOLE DEVICE DATABASE 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.MAR;1 (7)

```

511
512 ;
513 ; CONSOLE UCB
514 ;
515         ORB      OPA$ORBO
516         STO_ORB L_OWNER, LONG, <^X010001>
517         UCB      OPA$UCB0, <<UCB$C_TT_LENGTH - UCB$C_LENGTH + 3 + 64> / 4>, OPA$ORBO
518         STO_UCB B_FLCK, BYTE, SPL$C_IOLOCK8
519         STO_UCB B_DIPL, BYTE, 20
520         STO_UCB L_CRB, ADDRESS, OPA$CRB
521         STO_UCB L_DLCK, ADDRESS, OPA$SPL
522         STO_UCB L_DDB, ADDRESS, OPA$GL_DDB
523         STO_UCB L_DEVCHAR, LONG, <<DEV$M_REC!-
524             DEV$M_AVL!-
525             DEV$M_CCL!-
526             DEV$M_TRM!-
527             DEV$M_IDV!-
528             DEV$M_ODV>>
529         STO_UCB L_DEVCHAR2, LONG, <<DEV$M_NNM>>
530         STO_UCB B_DEVCLASS, BYTE, DC$ _TERM
531         STO_UCB L_AFFINITY, LONG, -1
532         STO_UCB B_DEVTYPE, BYTE, DT$ _LA36
533         STO_UCB W_DEVBUSIZ, WORD, 132
534         STO_UCB W_STS, WORD, UCB$M ONLINE
535         STO_UCB L_DEVDEPEND, LONG, <<TT$M_LOWER!TT$M_TTSYNC!TT$M_WRAP>>
536         STO_UCB L_DEVDEPEND+3, BYTE, 24
537         STO_UCB W_TT_DESIZE, WORD, 132
538         STO_UCB L_TT_DECHAR, LONG, <<TT$M_LOWER!TT$M_TTSYNC!TT$M_WRAP>>
539         STO_UCB L_TT_DECHAR+3, BYTE, 24
540         STO_UCB W_TT_SPEED, WORD, TT$C_BAUD_300
541         STO_UCB B_TT_DETYPE, BYTE, TT$ _LA36
542         STO_UCB W_TT_DESPEE, WORD, TT$C_BAUD_300
543 ;
544 ; CONSOLE CRB
545 ;
546 OPA$CRB::
547         ASSUME  CRB$L_FQFL  EQ  0
548         ASSUME  CRB$L_FQBL  EQ  4
549         .LONG   0, 0                ; FLINK, BLINK
550         .WORD   CD-OPA$CRB          ; SIZE
551         .BYTE   DYN$C_CRB           ; TYPE IS CRB
552
553         ASSUME  CRB$B_FLCK  EQ 11
554         .BYTE   0                    ; FLCK (FORK LOCK INDEX)
555
556         ASSUME  CRB$L_FPC   EQ 12
557         ASSUME  CRB$L_FR3  EQ 16
558         ASSUME  CRB$L_FR4  EQ 20
559         .LONG   0, 0, 0              ; FPC, FR3, FR4
560
561         ASSUME  CRB$L_WQFL  EQ 24
562         ASSUME  CRB$L_WQBL  EQ 28
563         .LONG   0, 0
564
565         ASSUME  CRB$B_TT_TYPE EQ 32
566         .LONG   0                    ; TT DEVICE TYPE + 3 UNUSED BYTES
567

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX  
8 Page 13  
X-8 SYSTEM CONSOLE DEVICE DATABASE 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.MAR;1 (7)

```

568     ASSUME  CRB$W_REFC  EQ  36
569     ASSUME  CRB$B_MASK  EQ  38
570     ASSUME  CRB$B_UNIT_BRK  EQ  39
571
572     .LONG    1                ; REF COUNT=1 AND NEVER BUSY
573     ASSUME  CRB$L_AUXSTRUC  EQ  40
574     .LONG    0                ; Auxiliary structure ptr.
575
576     ASSUME  CRB$L_TIMELINK  EQ  44
577     .LONG    0                ; CRB thread for periodic wakeups.
578
579     ASSUME  CRB$L_DUETIME   EQ  48
580     .LONG    0                ; Time when to periodically awaken
581
582     ASSUME  CRB$L_TOUTROUT  EQ  52
583     .LONG    0                ; Routine to call at periodic awakening
584
585     ASSUME  CRB$L_LINK      EQ  56
586     .LONG    0                ; NO NEXT CRB
587
588     ASSUME  CRB$L_DLCK      EQ  60
589     .ADDRESS OPA$SPL                ; CONSOLE SPINLOCK ADDRESS
590
591     ASSUME  <CRB$L_INTD+VEC$L_BUGCHECK> EQ  64
592     .BLKB   16                ; room for multi-level dispatch code
593
594 CON$INTDISI::                ;
595     ASSUME  CON$INTDISI-OPA$CRB  EQ  CRB$L_INTD
596
597     ASSUME  VEC$Q_DISPATCH  EQ  0
598     PUSHR  #^M<R0,R1,R2,R3,R4,R5> ; SAVE REGISTERS
599     JSB    G^CON$INTINP      ; INPUT INTERRUPT SERVICE
600
601     ASSUME  VEC$L_IDB       EQ  8
602     .ADDRESS OPA$IDB        ; POINTER TO IDB
603
604     ASSUME  VEC$L_INITIAL   EQ  12
605     .LONG    CON$INITIAL    ; INITIALIZE CONTROLLER ENTRY POINT
606
607     ASSUME  VEC$W_MAPREG    EQ  16
608     ASSUME  VEC$B_NUMREG    EQ  18
609     ASSUME  VEC$B_DATAPATH  EQ  19
610     .LONG    0                ; MAP AND DATA PATH ALLOCATION CONTROL
611
612     ASSUME  VEC$L_ADP       EQ  20
613     .LONG    0                ; ADDRESS OF ADP
614
615     ASSUME  VEC$L_UNITINIT  EQ  24
616     .LONG    CON$INITLINE   ; INITIALIZE UNIT
617
618     ASSUME  VEC$L_START     EQ  28
619     .LONG    0                ; UNUSED LONGWORD
620
621     ASSUME  VEC$L_UNITDISC  EQ  32
622     .LONG    0                ; UNUSED LONGWORD
623
624     ASSUME  VEC$W_MAPALT    EQ  36

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX  
8 Page 14

X-8 SYSTEM CONSOLE DEVICE DATABASE 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.MAR;1 (7)

```

625     ASSUME  VEC$W_NUMALT   EQ  38
626     .LONG   0                ; ALTERNATE MAP ALLOCATION CONTROL
627     ASSUME  VEC$K_LENGTH   EQ  56
628
629     .BLKB   16                ; room for multi-level dispatch code
630 CON$INTDISO::
631     ASSUME  CON$INTDISO-OPA$CRB EQ  CRB$L_INTD2
632
633     ASSUME  VEC$Q_DISPATCH EQ  0
634     PUSHR  #^M<R0,R1,R2,R3,R4,R5> ; SAVE REGISTERS
635     JSB    G^CON$INTOUT      ; OUTPUT INTERRUPT SERVICE
636     .ADDRESS OPA$IDB         ; POINTER TO IDB
637
638     ASSUME  VEC$L_IDB       EQ  8
639 CD:
640
641 ;
642 ; CONSOLE IDB
643 ;
644 OPA$IDB::
645     ASSUME  IDB$L_CSR       EQ  0
646     .LONG   0                ; CSR ADDRESS
647
648     ASSUME  IDB$L_OWNER     EQ  4
649     .LONG   0                ; OWNER UCB ADDRESS
650     .WORD   ID-OPA$IDB      ; SIZE OF IDB
651     .BYTE   DYN$C_IDB      ; TYPE OF STRUCTURE
652     .BYTE   0                ; UNUSED
653
654     ASSUME  IDB$W_UNITS     EQ  12
655     .WORD   6                ; NUMBER OF UNITS
656     .BYTE   0                ; TT ENABLE
657     .BYTE   0                ; CSR OFFSET TO MAIN CSR FOR COMBO STYLE DEV
658
659     ASSUME  IDB$B_COMBO_VECTOR_OFFSET EQ  16
660     .BYTE   0                ; VECTOR OFFSET TO MAIN VECTOR FOR COMBO STY
661     .BYTE   0                ; UNUSED
662     .WORD   0                ; UNUSED
663
664     ASSUME  IDB$L_SPL       EQ  20
665     .ADDRESS OPA$SPL
666
667     ASSUME  IDB$L_ADP       EQ  24
668     .LONG   0                ; ADAPTER ADDRESS
669
670     ASSUME  IDB$L_UCBLST    EQ  28
671     .ADDRESS OPA$UCB0       ; UNIT 0 UCB ADDRESS
672     .LONG   0                ; UNIT 1 UCB ADDRESS (FLOPPY)
673     .LONG   0                ; UNIT 2 INPUT UCB ADDRESS (FLOPPY)
674     .LONG   0                ; UNIT 3 USED BY VENUS ONLY
675     .LONG   0                ; UNIT 4 (RESERVED)
676     .LONG   0                ; UNIT 5 (RESERVED)
677 ID:
678
679 ;
680 ; CONSOLE DEVICE SPINLOCK
681 ;

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX  
8 Page 15

X-8 SYSTEM CONSOLE DEVICE DATABASE 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.MAR;1 (7)

```

682 OPA$SPL:
683     ASSUME  SPL$B_SPINLOCK  EQ  0
684     ASSUME  SPL$B_IPL       EQ  1
685     ASSUME  SPL$B_RANK      EQ  2
686     ASSUME  SPL$B_VEC_INX   EQ  3
687     .BYTE   0                ; STRUCTURE AVAILABLE
688     .BYTE   20               ; IPL = 20
689     .BYTE   -1              ; LOWEST RANK
690     .BYTE   0                ; OWNER VECTOR INDEX
691
692     ASSUME  SPL$W_OWN_CNT    EQ  4
693     ASSUME  SPL$W_WAIT_CPUS EQ  6
694     .WORD   -1              ; NO OWNERS
695     .WORD   0                ; NO WAITERS
696
697     ASSUME  SPL$W_SIZE       EQ  8
698     ASSUME  SPL$B_TYPE       EQ 10
699     ASSUME  SPL$B_SUBTYPE    EQ 11
700     .WORD   SPL$C_LENGTH     ; STRUCTURE LENGTH
701     .BYTE   DYN$C_SPL        ; SPINLOCK CONTROL BLOCK
702     .BYTE   SPL$C_SPL_DEVICELOCK ; DYNAMIC SPINLOCK TYPE
703
704     ASSUME  SPL$S_L_OWN_CPU  EQ 12
705     .LONG   0                ; NO OWNER
706
707     ASSUME  SPL$S_L_OWN_PC_VEC EQ 16
708     .BLKL  SPL$K_PC_VEC_CNT  ; NO PC'S YET
709
710     ASSUME  SPL$S_L_WAIT_PC  EQ 48
711     .LONG   0
712
713     ASSUME  SPL$Q_ACQ_COUNT  EQ 52
714     .QUAD   0
715
716     ASSUME  SPL$S_L_BUSY_WAITS EQ 60
717     .LONG   0
718
719     ASSUME  SPL$Q_SPINS       EQ 64
720     .QUAD   0
721
722     ASSUME  SPL$S_L_TIMO_INT  EQ 72
723     .LONG   0
724
725     ASSUME  SPL$S_L_RLS_PC    EQ 76
726     .LONG   0
727
728     ASSUME  <.-OPA$SPL>      EQ  SPL$K_LENGTH
729

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX

8 Page 16

X-8 SYSTEM PERMANENT MAILBOX DATABASE 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.MAR;1 (

```
731      .SBTTL  SYSTEM PERMANENT MAILBOX DATABASE
732
733 ;
734 ; MAILBOX DDB
735 ;
736      DDB      MB$GL_DDB,NL$GL_DDB,MB$GL_UCB1,0,,0,<MBA>,<MBDRIVER>
737 ;
738 ; CLONE MAILBOX UCB
739 ;
740 ;
741 ; NOTE THAT THIS UCB IS NOT IN THE DDB'S UCB LIST
742 ;
743      ORB      MB$ORBO
744      STO_ORB L_OWNER, LONG, <^X010001>
745      UCB      MB$UCB0, 0, MB$ORBO
746      STO_UCB W_MB_SEED, WORD, 0
747      STO_UCB B_FLCK, BYTE, SPL$C_MAILBOX
748      STO_UCB B_DIPL, BYTE, IPL$ _MAILBOX
749      STO_UCB W_MSGMAX, WORD, 20
750      STO_UCB W_BUFQUO, WORD, -1
751      STO_UCB L_CRB, ADDRESS, SYS_CRB
752 ;
753      STO_UCB L_DLCK, ADDRESS, SMP$GL_MAILBOX
754      STO_UCB L_DDB, ADDRESS, MB$GL_DDB
755      STO_UCB L_LINK, ADDRESS, MB$GL_UCB1
756      STO_UCB L_DEVCHAR, LONG, <<DEV$M_REC!-
757          DEV$M_AVL!-
758          DEV$M_IDV!-
759          DEV$M_MBX!-
760          DEV$M_ODV!-
761          DEV$M_SHR>>
761      STO_UCB L_DEVCHAR2, LONG, <<DEV$M_NNM>>
762      STO_UCB B_DEVCLASS, BYTE, DC$ _MAILBOX
763      STO_UCB L_AFFINITY, LONG, -1
764      STO_UCB B_DEVTYPE, BYTE, DT$ _MBX
765      STO_UCB W_DEVBUFSIZ, WORD, 256
766      STO_UCB W_REFC, WORD, 1
767      STO_UCB W_UNIT, WORD, 0
768      STO_UCB W_STS, WORD, UCB$M_ONLINE
769      STO_UCB W_DEVSTS, WORD, UCB$M_PRMMBX
770      STO_UCB L_DDT, LONG, 0
771 MB$MBO END:
772 SYS$C_MBXUCBSIZ ==      <MB$MBO_END - MB$UCB0>
773
774 ;
775 ; SYSTEM JOB CONTROLLER MAILBOX
776 ;
777      .ALIGN  QUAD
778 SYS$GL_JOBCTLMB::
779 SYS$C_JOBCTLMB==^A/MBA1/
780      UCB      MB$GL_UCB1, 0, MB$GL_ORB1
781      STO_UCB L_FQFL, ADDRESS, MB$GL_UCB1
782      STO_UCB L_FQFL+4, ADDRESS, MB$GL_UCB1
783      STO_UCB B_FLCK, BYTE, SPL$C_MAILBOX
784      STO_UCB B_DIPL, BYTE, IPL$ _MAILBOX
785      STO_UCB W_MSGMAX, WORD, 60
786      STO_UCB W_BUFQUO, WORD, -1
787      STO_UCB L_CRB, ADDRESS, SYS_CRB
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX

8 Page 17

X-8 SYSTEM PERMANENT MAILBOX DATABASE 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.MAR;1 (

```

788 ;          STO_UCB L_DLCK, ADDRESS, SMP$GL_MAILBOX
789          STO_UCB L_DDB, ADDRESS, MB$GL_DDB
790          STO_UCB L_LINK, ADDRESS, MB$GL_UCB2
791          STO_UCB L_DEVCHAR, LONG, <<DEV$M_REC!-
792              DEV$M_AVL!-
793              DEV$M_MBX!-
794              DEV$M_IDV!-
795              DEV$M_ODV!-
796              DEV$M_SHR>>
797          STO_UCB L_DEVCHAR2, LONG, <<DEV$M_NNM>>
798          STO_UCB B_DEVCLASS, BYTE, DC$_MAILBOX
799          STO_UCB L_AFFINITY, LONG, -1
800          STO_UCB W_DEVBUFSIZ, WORD, 1024
801          STO_UCB W_REFC, WORD, 1
802          STO_UCB W_UNIT, WORD, 1
803          STO_UCB W_STS, WORD, UCB$M_ONLINE
804          STO_UCB W_DEVSTS, WORD, <UCB$M_PRMBX+^X08000>
805          STO_UCB L_DDT, LONG, 0
806          ORB          MB$GL_ORB1
807          STO_ORB L_OWNER, LONG, <^X010004>
808          STO_ORB W_PROT, WORD, <^XOFFOF>
809 ;
810 ; SYSTEM OPERATOR MAILBOX
811 ;
812          .ALIGN QUAD
813 SYS$GL_OPRMBX::
814 SYS$C_OPRMBX==^A/MBA2/
815          UCB          MB$GL_UCB2, 0, MB$GL_ORB2
816          STO_UCB L_FQFL, ADDRESS, MB$GL_UCB2
817          STO_UCB L_FQFL+4, ADDRESS, MB$GL_UCB2
818          STO_UCB B_FLCK, BYTE, SPL$C_MAILBOX
819          STO_UCB B_DIPL, BYTE, IPL$C_MAILBOX
820          STO_UCB W_MSGMAX, WORD, 20
821          STO_UCB W_BUFQUO, WORD, -1
822          STO_UCB L_CRB, ADDRESS, SYS_CRB
823 ;          STO_UCB L_DLCK, ADDRESS, SMP$GL_MAILBOX
824          STO_UCB L_DDB, ADDRESS, MB$GL_DDB
825          STO_UCB L_LINK, ADDRESS, MB$GL_UCB3
826          STO_UCB L_DEVCHAR, LONG, <<DEV$M_REC!-
827              DEV$M_AVL!-
828              DEV$M_MBX!-
829              DEV$M_IDV!-
830              DEV$M_ODV!-
831              DEV$M_SHR>>
832          STO_UCB L_DEVCHAR2, LONG, <<DEV$M_NNM>>
833          STO_UCB B_DEVCLASS, BYTE, DC$_MAILBOX
834          STO_UCB L_AFFINITY, LONG, -1
835          STO_UCB W_DEVBUFSIZ, WORD, 2560
836          STO_UCB W_REFC, WORD, 1
837          STO_UCB W_UNIT, WORD, 2
838          STO_UCB W_STS, WORD, UCB$M_ONLINE
839          STO_UCB W_DEVSTS, WORD, UCB$M_PRMBX
840          STO_UCB L_DDT, LONG, 0
841          ORB          MB$GL_ORB2
842          STO_ORB L_OWNER, LONG, <^X010004>
843          STO_ORB W_PROT, WORD, <^XOFFOF>
844 ;

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX  
8 Page 18  
X-8 SYSTEM PERMANENT MAILBOX DATABASE 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.MAR;1 (

```
845 ; AUDIT SERVER MAILBOX
846 ;
847     .ALIGN QUAD
848 SYS$GL_AUDSRVMBX::
849 SYS$C_AUDSRVMBX==^A/MBA3/
850     UCB      MB$GL_UCB3,0, MB$GL_ORB3
851     STO_UCB L_FQFL, ADDRESS, MB$GL_UCB3
852     STO_UCB L_FQFL+4, ADDRESS, MB$GL_UCB3
853     STO_UCB B_FLCK, BYTE, SPL$C_MAILBOX
854     STO_UCB B_DIPL, BYTE, IPL$ MAILBOX
855     STO_UCB W_MSGMAX, WORD, 20
856     STO_UCB W_BUFQUO, WORD, -1
857     STO_UCB L_CRB, ADDRESS, SYS_CRB
858 ;     STO_UCB L_DLCK, ADDRESS, SMP$GL_MAILBOX
859     STO_UCB L_DDB, ADDRESS, MB$GL_DDB
860     STO_UCB L_DEVCHAR, LONG, <<DEV$M_REC!-
861         DEV$M_AVL!-
862         DEV$M_MBX!-
863         DEV$M_IDV!-
864         DEV$M_ODV!-
865         DEV$M_SHR>>
866     STO_UCB L_DEVCHAR2, LONG, <<DEV$M_NNM>>
867     STO_UCB B_DEVCLASS, BYTE, DC$ MAILBOX
868     STO_UCB L_AFFINITY, LONG, -1
869     STO_UCB W_DEVBUFSIZ, WORD, 2560
870     STO_UCB W_REFC, WORD, 1
871     STO_UCB W_UNIT, WORD, 3
872     STO_UCB W_STS, WORD, UCB$M ONLINE
873     STO_UCB W_DEVSTS, WORD, UCB$M_PRMMBX
874     STO_UCB L_DDT, LONG, 0
875     ORB      MB$GL_ORB3
876     STO_ORB L_OWNER, LONG, <^X010004>
877     STO_ORB W_PROT, WORD, <^XOFFOF>
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX  
8 Page 19

X-8 NULL DEVICE (NLA) DATABASE 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.MAR;1 (9)

```
879      .SBTTL  NULL DEVICE (NLA) DATABASE
880 ;
881 ; NLA DDB
882 ;
883      DDB      NL$GL_DDB,0,NL$GL_UCB0,0,,0,<NLA>,<NLDRIVER>
884
885 ;
886 ; NL UCB FOR UNIT 0
887 ;
888      ORB      NL$GL_ORB0
889      STO_ORB L_OWNER, LONG, <^X010001>
890      UCB      NL$GL_UCB0, 0, NL$GL_ORB0
891      STO_UCB B_FLCK, BYTE, SPL$C_IOLOCK8
892      STO_UCB B_DIPL, BYTE, 8
893      STO_UCB L_CRB, ADDRESS, SYS_CRB
894 ;
895 ;
896      STO_UCB L_DLCK, ADDRESS, SMP$GL_IOLOCK8????
897      STO_UCB L_DDB, ADDRESS, NL$GL_DDB
898      STO_UCB L_DEVCHAR, LONG, <<DEV$M_REC!-
899          DEV$M_AVL!-
900          DEV$M_MBX!-
901          DEV$M_IDV!-
902          DEV$M_ODV!-
903          DEV$M_SHR>>
903      STO_UCB B_DEVCLASS, BYTE, DC$_MAILBOX
904      STO_UCB L_AFFINITY, LONG, -1
905      STO_UCB B_DEVTYPE, BYTE, DT$_NULL
906      STO_UCB W_DEVBUFSIZ, WORD, 512
907      STO_UCB W_REFC, WORD, 1
908      STO_UCB W_UNIT, WORD, 0
909      STO_UCB W_STS, WORD, UCB$M_ONLINE
910      STO_UCB L_DDT, LONG, 0
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX  
8 Page 20

X-8 NETWORK DEVICE DATABASE 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.MAR;1 (10)

```

912      .SBTTL  NETWORK DEVICE DATABASE
913
914 ;
915 ; COMMON CRB FOR MAILBOX TYPE DEVICES
916 ;
917 SYS_CRB:
918     ASSUME  CRB$$_FQFL  EQ  0
919     ASSUME  CRB$$_FQBL  EQ  4
920     .LONG   0,0
921     .WORD   10$-SYS_CRB
922     .BYTE   DYN$C_CRB
923     ASSUME  CRB$$_FLCK  EQ  11
924     .BYTE   0
925     ASSUME  CRB$$_FPC   EQ  12
926     ASSUME  CRB$$_FR3  EQ  16
927     ASSUME  CRB$$_FR4  EQ  20
928     .LONG   0, 0, 0
929     ASSUME  CRB$$_WQFL  EQ  24
930     ASSUME  CRB$$_WQBL  EQ  28
931     .LONG   0,0
932     ASSUME  CRB$$_TT_TYPE EQ  32
933     .LONG   0
934     ASSUME  CRB$$_REFC  EQ  36
935     ASSUME  CRB$$_MASK  EQ  38
936     ASSUME  CRB$$_UNIT_BRK EQ  39
937     .LONG   0
938     ASSUME  CRB$$_AUXSTRUC EQ  40
939     .LONG   0
940     ASSUME  CRB$$_TIMELINK EQ  44
941     .LONG   0
942
943     ASSUME  CRB$$_DUETIME  EQ  48
944     .LONG   0
945
946     ASSUME  CRB$$_TOUTROUT EQ  52
947     .LONG   0
948
949     ASSUME  CRB$$_LINK     EQ  56
950     .LONG   0
951
952     ASSUME  CRB$$_DLCK     EQ  60
953 ; .ADDRESS SMP$$_GL_MAILBOX
954     .LONG   0
955
956     ASSUME  <CRB$$_INTD+VEC$$_BUGCHECK> EQ  64
957     .BLKB   16
958
959     ASSUME  .-SYS_CRB      EQ  CRB$$_INTD
960     ASSUME  VEC$$_DISPATCH EQ  0
961     .LONG   0,0
962
963     ASSUME  VEC$$_IDB      EQ  8
964     .ADDRESS SYS_CRB
965
966     ASSUME  VEC$$_INITIAL  EQ  12
967     ASSUME  VEC$$_MAPREG   EQ  16
968     ASSUME  VEC$$_NUMREG   EQ  18

```



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PERMANENT\_DEVICE\_DATABASE - Miscellaneous system area - base image 10-MAY-1989 16:57:36 VAX  
8 Page 21

X-8 NETWORK DEVICE DATABASE 15-SEP-1988 10:32:38 PERMANENT\_DEVICE\_DATABASE.MAR;1 (10)

```

969      ASSUME  VEC$B_DATAPATH  EQ  19
970      ASSUME  VEC$L_ADP       EQ  20
971      ASSUME  VEC$L_UNITINIT  EQ  24
972      ASSUME  VEC$L_START     EQ  28
973      ASSUME  VEC$L_UNITDISC  EQ  32
974      ASSUME  VEC$W_MAPALT    EQ  36
975      ASSUME  VEC$W_NUMALT    EQ  38
976      .LONG   0,0,0,0,0,0    ; NO INITIALIZATION FOR CONTROLLER OR UNIT

977
978      ASSUME  VEC$K_LENGTH     EQ  56
979      .BLKB  16                ; room for multi-level dispatch code
980 10$:
981      ASSUME  10$-SYS_CRB      EQ  CRB$L_INTD2
982
983
984
985 OPA$VECTOR:
986      $VECINI OPA, CON$NULL
987
988      $VEC    STARTIO, CON$STARTIO
989      $VEC    DISCONNECT, CON$DISCONNECT
990      $VEC    SET_LINE, CON$SET_LINE
991      $VEC    DS_SET, CON$DS_SET
992      $VEC    XON, CON$XON
993      $VEC    XOFF, CON$XOFF
994      $VEC    STOP, CON$STOP
995      $VEC    STOP2, CON$STOP2
996      $VEC    ABORT, CON$ABORT
997      $VEC    RESUME, CON$RESUME
998      $VEC    SET_MODEM, CON$SET_MODEM
999
1000     $VECEND
1001     .END

```

## **7 LDAT.LIS**

**LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 0**  
**Table of contents**

**(2) 350 SPINLOCK DATA STRUCTURES**

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 1  
X-35 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (1)

```

1      .TITLE  LDAT - SPIN LOCK DATA BASES
2      .IDENT  'X-35'
3 ;
4 ;*****
5 ;*
6 ;*  COPYRIGHT (c) 1987, 1988 BY
7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
8 ;*  ALL RIGHTS RESERVED.
9 ;*
10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
11 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
14 ;*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
15 ;*  TRANSFERRED.
16 ;*
17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
18 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
19 ;*  CORPORATION.
20 ;*
21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
23 ;*
24 ;*
25 ;*****
26 ;
27 ;++
28 ; SYSTEM SPIN LOCKS
29 ;--
30 ;
31 ;
32 ; AUTHOR: Rod Gamache   16-Jul-1985
33 ;
34 ; MODIFIED BY:
35 ;
36 ;      X-35      EMB0369      Ellen M. Batbouta      21-Nov-1988
37 ;              The spinlocks, EMB and MCHECK, are now one spinlock.
38 ;              The spinlock vector table for EMB and MCHECK contains
39 ;              the address of SMP$GL_EMB for both vectors.
40 ;
41 ;      X-34      MSH0350      Michael S. Harvey      17-Oct-1988
42 ;              Replace references to IPL$_SYNCH on originally defined locks
43 ;              with the lock specific IPL symbols, as was done for more
44 ;              recently added locks. Also, replace hardcoded IPL constants
45 ;              with symbols.
46 ;
47 ;      X-33      RNG5033      Rod Gamache      23-Jul-1987
48 ;              Change IPL of INVALIDATE spinlock back to IPL 19.
49 ;              Make OWN_CPU and RLS_PC fields of SPL structure init to 0.
50 ;
51 ;      X-32      MSH0313      Michael S. Harvey      15-May-1987
52 ;              Add longword for PC of releaser of multiple acquisitions.
53 ;
54 ;      X-30      SJF          Stu Farnham      11-May-1987
55 ;              Change IPL of INVALIDATE and VIRTCONS to 20.
56 ;
57 ;      X-29      RNG5029      Rod Gamache      24-Mar-1987

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 2  
X-35 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (1)

58 ; Compress spinlock indices into range 20-3F (HEX).  
59 ;  
60 ; X-28 RNG5028 Rod Gamache 12-Mar-1987  
61 ; Move JIB to below MMG in ranking. MMG should always  
62 ; be adjacent to SCHED to allow pool extension to work  
63 ; correctly.  
64 ;  
65 ; X-27 MSH0307 Michael S. Harvey 10-Mar-1987  
66 ; Add JIB and ASTDEL spinlocks. Also, simplify the way  
67 ; that forklocks are treated.  
68 ;  
69 ; X-26 RNG5026 Rod Gamache 3-Mar-1987  
70 ; Add MEGA synchronization spinlock.  
71 ;  
72 ; X-25 MJW0113 Michael J. Worcester 26-Feb-1987  
73 ; Change reference to DYN\$C\_SPL\_'LOCKTYPE' in SPINLOCK  
74 ; MACRO to SPL\$C\_SPL\_'LOCKTYPE'.  
75 ;  
76 ; X-24 MSH0246 Michael S. Harvey 10-Feb-1987  
77 ; Remove obsolete spin counter in spinlock structures.  
78 ;  
79 ; X-23 RNG5023 Rod Gamache 30-Dec-1986  
80 ; Create an IPL vector which corresponds to the spinlock  
81 ; vector, but contains only IPLs. To be used for uniprocessor  
82 ; forklocking. Remove all vestiges of SMP\$AR\_RANK\_MAP vector.  
83 ; Check that all forklocks have a rank higher than SPL\$C\_FORKLOCK.  
84 ; Create a spinlock vector that contains SPL\$C\_MAXLOCK entries.  
85 ;  
86 ; X-22 MSH0285 Michael S. Harvey 8-Dec-1986  
87 ; Remove BUSERR spinlock.  
88 ;  
89 ; X-21 RNG0021 Rod Gamache 8-Dec-1986  
90 ; Move SPL\$L\_TIMO\_INT after the reserved longword.  
91 ;  
92 ; X-20 SJF Stu Farnham 8-Dec-1986  
93 ; Reorder FILSYS, IOLLOCK8, and PR\_LK8. Place VIRTCONS  
94 ; spinlock correctly with respect to IPL.  
95 ;  
96 ; X-19 SJF Stu Farnham 18-Nov-1986  
97 ; Initialize SPL\$L\_TIMO\_INT.  
98 ;  
99 ; X-18 MSH0285 Michael S. Harvey 4-Nov-1986  
100 ; Move MCHECK below EMB.  
101 ;  
102 ; X-17 MSH0285 Michael S. Harvey 29-Oct-1986  
103 ; Add machine check spinlock.  
104 ;  
105 ; X-16 RNG0016 Rod Gamache 28-Oct-1986  
106 ; Add check for IPL's associated with each spinlock database.  
107 ;  
108 ; X-15 RNG0015 Rod Gamache 24-Oct-1986  
109 ; Remove XDELTA spinlocks, add BUSERR spinlock.  
110 ;  
111 ; X-14 SJF Stu Farnham 13-Oct-1986  
112 ; Add virtual console spinlock. Do absolutely nothing to  
113 ; the ranking of MMG, SCHED, and FILSYS.  
114 ;

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 3  
X-35 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (1)

|       |        |  |                   |             |
|-------|--------|--|-------------------|-------------|
| 115 ; | X-13   | RNG0013  | Rod N. Gamache    | 24-Sep-1986 |
| 116 ; |        | Move MMG spinlock up next to SCHED again! Also move            |                   |             |
| 117 ; |        | FILSYS spinlock above both IOLOCK8 and PR_LOCK8.               |                   |             |
| 118 ; |        |  |                   |             |
| 119 ; | X-12   | WCT001   | Ward C. Travis    | 21-Aug-1986 |
| 120 ; |        | Add a PSECT to contain the label for the spinlock              |                   |             |
| 121 ; |        | rank mapping vector, a new database to facilitate              |                   |             |
| 122 ; |        | dynamic reordering of lock IPLs. Alter SPINLOCK                |                   |             |
| 123 ; |        | macro to fill the rank map in at link.                         |                   |             |
| 124 ; |        |  |                   |             |
| 125 ; | X-11   | MSH0274  | Michael S. Harvey | 10-Aug-1986 |
| 126 ; |        | Move MMG back to where it was. PAGEFAULT and WRTMFYPAG         |                   |             |
| 127 ; |        | still need MMG across build-pkt routines, which will           |                   |             |
| 128 ; |        | acquire IOLOCK8.   |                   |             |
| 129 ; |        |  |                   |             |
| 130 ; | X-10   | RNG0010  | Rod Gamache       | 30-Jul-1986 |
| 131 ; |        | Shuffle spinlocks again! Try moving MMG spinlock up            |                   |             |
| 132 ; |        | next to SCHED.   |                   |             |
| 133 ; |        |  |                   |             |
| 134 ; | X-9    | SJF  | Stu Farnham       | 28-Jul-1986 |
| 135 ; |        | Rename IPINT spinlock to INVALIDATE to better reflect          |                   |             |
| 136 ; |        | its function under the latest implementation of                |                   |             |
| 137 ; |        | interprocessor interrupts.                                     |                   |             |
| 138 ; |        |  |                   |             |
| 139 ; | X-8    | MSH0266  | Michael S. Harvey | 23-Jul-1986 |
| 140 ; |        | Force IPINT spinlock's IPL to the lower of two legal           |                   |             |
| 141 ; |        | values, the value which is valid for existing multiprocessors. |                   |             |
| 142 ; |        | This will be restored later when I write a generalized         |                   |             |
| 143 ; |        | routine for modifying a given spinlock's IPL value and         |                   |             |
| 144 ; |        | reordering the spinlock rank assignments accordingly. Such     |                   |             |
| 145 ; |        | rank reordering is necessary to prevent rank or IPL violations |                   |             |
| 146 ; |        | when attempting to acquire/release adjacently ranked spinlocks |                   |             |
| 147 ; |        | whose IPLs have been modified.                                 |                   |             |
| 148 ; |        |  |                   |             |
| 149 ; | X-6H10 | RNG6010  | Rod Gamache       | 24-Jun-1986 |
| 150 ; |        | Switch the order of TIMER and SCHED spinlocks...               |                   |             |
| 151 ; |        | EXE\$RMVTIMQ deallocates pool, which may need the SCHED        |                   |             |
| 152 ; |        | SPINLOCK.  |                   |             |
| 153 ; |        |  |                   |             |
| 154 ; | X-6H9  | SJF  | Stu Farnham       | 17-Jun-1986 |
| 155 ; |        | Change IPL of IPINT spinlock to 21                             |                   |             |
| 156 ; |        |  |                   |             |
| 157 ; | X-6H8  | MSH0252  | Michael S. Harvey | 16-Jun-1986 |
| 158 ; |        | Add spinlock data structure longword to the control block.     |                   |             |
| 159 ; |        |  |                   |             |
| 160 ; | X-6H7  | RNG6007  | Rod Gamache       | 13-Jun-1986 |
| 161 ; |        | IOLOCK8 can't be lowest ranked spinlock at IPL 8 - PAGEFAULT   |                   |             |
| 162 ; |        | calls the driver with MMG held! Therefore MMG must be below    |                   |             |
| 163 ; |        | IOLOCK8!   |                   |             |
| 164 ; |        |  |                   |             |
| 165 ; | X-6H6  | MSH0250  | Michael S. Harvey | 12-Jun-1986 |
| 166 ; |        | Remove initial IPL control since a new mechanism renders       |                   |             |
| 167 ; |        | this feature obsolete.   |                   |             |
| 168 ; |        |  |                   |             |
| 169 ; | X-6H3  | RNG6003  | Rod Gamache       | 10-Jun-1986 |
| 170 ; |        | Shuffle spinlocks - move IOLOCK8 to lowest ranked spinlock     |                   |             |
| 171 ; |        | at IPL 8.  |                   |             |

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 4  
X-35 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (1)

172 ;  
173 ; X-6H2 RNG6002 Rod Gamache 5-Jun-1986  
174 ; Make sure that all SPINLOCKS have an IPL above 3. Also,  
175 ; make sure that SPINLOCK database match constants defined  
176 ; in SPLCODDEF.  
177 ;  
178 ; X-6F8 MSH0243 Michael S. Harvey 20-May-1986  
179 ; Close off structure interlock vs. ownership by allowing  
180 ; the ownership counter to served as the indicator of  
181 ; a spinlock being owned.  
182 ;  
183 ; X-6F7 MSH0246 Michael S. Harvey 15-May-1986  
184 ; Add acquisition counts to the list of assumed offsets  
185 ; for spinlock data structures in the spinlock declaration  
186 ; macro.  
187 ;  
188 ; X-6F6 MSH0243 Michael S. Harvey 15-May-1986  
189 ; Snug up the allocation of rank values from highest  
190 ; possible value down, rather than from lowest possible  
191 ; value up.  
192 ;  
193 ; X-6F5 RNG0605 Rod N. Gamache 14-May-1986  
194 ; Add name to SPINLOCK structure.  
195 ;  
196 ; X-6F4 MSH0243 Michael S. Harvey 13-May-1986  
197 ; Reverse the specification of rank. A higher rank  
198 ; has a lower valued rank number assigned.  
199 ;  
200 ; X-1A5 SJF Stu Farnham 9-May-1986  
201 ; Add IPL 22 interprocessor interrupt spinlock. This  
202 ; is because of the INCREDIBLE HACK which requires that  
203 ; CPUs which have their IP interrupts at IPL 22 LOWER  
204 ; IPL to 21.  
205 ;  
206 ; X-1A4 SJF Stu Farnham 23-Apr-1986  
207 ; Add primary cpu boot fork locks  
208 ;  
209 ; X-1A3 RNG4003 Rod Gamache 6-Nov-1985  
210 ; Place IOLOCK8 spinlock after MMG spinlock (ie make IOLOCK8  
211 ; have a higher rank than MMG). This allows SWAPPER/PAGEFAULT  
212 ; to initiate I/O while holding the MMG spinlock.  
213 ;  
214 ; X-1A2 RNG4002 Rod Gamache 8-Oct-1985  
215 ; Add new SPINLOCKS for second SMP baselevel, new LOCKS  
216 ; are for each IPL for queue, QUEUEAST, IOLOCK8, IOLOCK9,  
217 ; IOLOCK10, IOLOCK11. Remove KERNEL lock. Make the init  
218 ; IPL = real IPL for spinlocks intended for forking.  
219 ;

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 5  
X-35 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (1)

```

221 ;
222 ; MACRO LIBRARY CALLS
223 ;
224     $DYNDEF                ; DEFINE DATA STRUCTURE TYPE CODES
225     $IPLDEF                ; DEFINE INTERRUPT PRIORITIES
226     $PRDEF                 ; DEFINE PROCESSOR REGISTERS
227     $$SPLDEF               ; DEFINE SPIN LOCK STRUCTURES
228     $$SPLCODDEF            ; DEFINE SPINLOCK INDEX CODES
229
230     LOWEST_RANK = 30        ; DEFINED RANK RANGE IS CURRENTLY
231                               ; FROM 0-31. RANK 31 IS RESERVED FOR
232                               ; ALL DEVICELOCKS, SO THE VALID RANGE
233                               ; FOR SYSTEM SPINLOCKS IS 0-30.
234                               ;
235                               ; EXTENDING THE RANGE TO GREATER
236                               ; THAN A LONGWORD OF BITS REQUIRES
237                               ; MODIFICATIONS IN THIS MODULE AS
238                               ; WELL AS IN THE CODE THAT ACQUIRES
239                               ; AND RELEASES SPINLOCKS.
240
241 ;
242 ; Add some ASSUMES which the SPINLOCK macro below depends upon.
243 ;
244     ASSUME    SPL$B_SPINLOCK    EQ    0
245     ASSUME    SPL$B_IPL         EQ    1
246     ASSUME    SPL$B_RANK        EQ    2
247     ASSUME    SPL$B_VEC_INX     EQ    3
248     ASSUME    SPL$W_OWN_CNT     EQ    4
249     ASSUME    SPL$W_WAIT_CPUS   EQ    6
250     ASSUME    SPL$W_SIZE        EQ    8
251     ASSUME    SPL$B_TYPE        EQ   10
252     ASSUME    SPL$B_SUBTYPE     EQ   11
253     ASSUME    SPL$L_OWN_CPU     EQ   12
254     ASSUME    SPL$L_OWN_PC_VEC  EQ   16
255     ASSUME    SPL$L_WAIT_PC     EQ   48
256     ASSUME    SPL$Q_ACQ_COUNT   EQ   52
257     ASSUME    SPL$L_BUSY_WAITS  EQ   60
258     ASSUME    SPL$Q_SPINS       EQ   64
259     ASSUME    SPL$L_TIMO_INT    EQ   72
260     ASSUME    SPL$L_RLS_PC      EQ   76
261     ASSUME    SPL$K_LENGTH      EQ   80
262
263 ;
264 ; MACRO DEFINITIONS:
265 ;
266 ;
267 ;
268 ; SPINLOCK, a macro to create a spinlock database entry
269 ;
270 ;     NAME      = The name of the spinlock. This string is equivalent
271 ;                to the name of the lock as specified in the various
272 ;                spinlock acquisition and release macros.
273 ;
274 ;     IPL       = Specifies the synchronization IPL that is to be associated
275 ;                with the spinlock. When this lock is acquired, the acquiring
276 ;                CPU's IPL will be set to this value.
277 ;

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 6  
X-35 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (1)

```

278 ;           Many spinlocks with an IPL that is in the range of software
279 ;           interrupts are potentially usable as forklocks, whether
280 ;           that is their primary use or not. Verify that the indices
281 ;           of these spinlocks are in the range required for forklocking.
282 ;
283 ; LOCKTYPE= Generates the subtype field of the spinlock data structure
284 ;           to indicate the expected primary use of the spinlock.
285 ;
286
287 .MACRO SPINLOCK NAME,IPL,LOCKTYPE=SPINLOCK,EQUATE
288
289 ASSUME IPL EQ IPL$ 'NAME
290 .IIF NDF $$$RANK, $$$RANK = 0
291 ASSUME $$$RANK EQ <SPL$C 'NAME-SPL$ _MIN_INDEX>
292 ...RANK 'NAME = $$$RANK
293 .IIF LT LOWEST_RANK-$$$RANK, .ERROR ; OVERRAN RESERVED DEVICELock RANK
294 .IIF NDF $$$IPL, $$$IPL = 31
295 .IIF GT IPL-$$$IPL, .ERROR ; OUT OF ORDER RANKING DETECTED
296 $$$IPL = IPL
297
298 .IF DIF <NAME>, <ASTDEL>
299 .IIF GE 3-IPL, .ERROR ; IPL_VALUE DOESN'T PREVENT RESCHEDULING ON SMP
300 .ENDC
301
302 .IF B EQUATE
303 .PSECT EXEC$SMP_DDD,QUAD,WRT,NOEXE,PIC
304 .ALIGN QUAD
305 SMP$GL_ 'NAME:: ; Name of spinlock
306 .BYTE 0 ; Structure not currently interlocked
307 .BYTE IPL ; Synchronization level for lock
308 .BYTE $$$RANK ; RANK for lock
309 .BYTE 0 ; PC vector index
310 .WORD -1 ; Owned if positive
311 .WORD 0 ; Waiting CPUs
312 .WORD SPL$C_LENGTH ; Length of spinlock control block
313 .BYTE DYN$C_SPL ; Data structure type code
314 .BYTE SPL$C_SPL_ 'LOCKTYPE' ; Spinlock subtype code
315 .LONG 0 ; Physical ID of owning CPU
316 .BLKL SPL$K_PC_VEC_CNT ; PC history of owning CPU
317 .LONG 0 ; PC of waiting CPU
318 .QUAD 0 ; Number of successful acquisitions
319 .LONG 0 ; Number of failed acquisitions
320 .QUAD 0 ; Number of spins
321 .LONG ^X7FFFFFF ; Longest possible timeout interval
322 .LONG 0 ; PC of last releaser of multiple acquisitio
323 .ENDC
324 $$$RANK = $$$RANK+1
325 .PSECT EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
326 . =4*<SPL$C 'NAME-SPL$ _MIN_INDEX>
327 .IF B EQUATE
328 .ADDRESS SMP$GL_ 'NAME ; Enter spinlock address in vector
329 .IFF
330 .ADDRESS SMP$GL_ 'EQUATE ; Enter address of equivalent spinlock
331 .ENDC
332 .PSECT EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
333 . =4*<SPL$C 'NAME-SPL$ _MIN_INDEX>
334 .LONG IPL

```



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 7  
X-35 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (1)

```
335      .ENDM   SPINLOCK
336
337      .LIST   MEB
338
339 ;
340 ; VERIFY IPL SYMBOLS:
341 ;
342
343      ASSUME  IPL$ _SYNCH  EQ  IPL$ _SCHED
344      ASSUME  IPL$ _SYNCH  EQ  IPL$ _MMG
345      ASSUME  IPL$ _SYNCH  EQ  IPL$ _FILSYS
346      ASSUME  IPL$ _SYNCH  EQ  IPL$ _TIMER
347      ASSUME  IPL$ _SYNCH  EQ  IPL$ _SCS
348      ASSUME  IPL$ _SYNCH  EQ  IPL$ _JIB
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 8  
X-35 SPINLOCK DATA STRUCTURES 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (2)

```

350      .SBTTL  SPINLOCK DATA STRUCTURES
351
352      .PSECT  EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
353
354 SMP$GL_IPL_VEC==.-<4*SPL$ _MIN_INDEX>      ; BIAS IPL vector from starting index
355                                              ; IPL vector (only used within loadable
356                                              ; image - so not universal).
357      .LONG   0[SPL$ _NUM_LOCKS]              ; Create vector of maximum IPLs

358
359      .PSECT  EXEC$SMP_AAA, LONG, WRT, NOEXE, PIC
360
361 ;+
362 ; SPINLOCK DATA STRUCTURE ADDRESS TABLE
363 ;--
364
365      .PSECT  EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
366
367 SMP$GL_SPNLKVEC==.-<4*SPL$ _MIN_INDEX>      ; BIAS SPINLOCK vector from starting
368      .LONG   0[SPL$ _NUM_LOCKS]              ; Create vector of maximum locks

369
370 ; Vector of SPINLOCK addresses follow here
371
372 ;++
373 ; SPINLOCK STRUCTURES, ORDERED BY DECREASING RANK.
374 ;
375 ; RANK VALUES ARE ASSIGNED IN REVERSE ORDER. HIGHER NUMBERED
376 ; VALUES INDICATE LOWER SPINLOCK RANK.
377 ;--
378
379
380 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
381 ;          ERROR LOG MESSAGE BUFFER DATABASE LOCK
382 ;
383      SPINLOCK      NAME=<EMB>, IPL=IPL$ _EMB
      .PSECT  EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_EMB::
      .BYTE  0              ; EMB of spinlock
      .BYTE  IPL$ _EMB      ; Structure not currently interlocked
      .BYTE  $$$RANK        ; Synchronization level for lock
      .BYTE  0              ; RANK for lock
      .WORD  -1             ; PC vector index
      .WORD  0              ; Owned if positive
      .WORD  0              ; Waiting CPUs
      .WORD  SPL$C _LENGTH  ; Length of spinlock control block

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 9  
X-35 SPINLOCK DATA STRUCTURES 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (2)

```
.BYTE DYN$C_SPL ; Data structure type code
.BYTE SPL$C_SPL_SPINLOCK ; Spinlock subtype code
.LONG 0 ; Physical ID of owning CPU
.BLKL SPL$K_PC_VEC_CNT ; PC history of owning CPU
.LONG 0 ; PC of waiting CPU
.QUAD 0 ; Number of successful acquisitions
.LONG 0 ; Number of failed acquisitions
.QUAD 0 ; Number of spins
.LONG ^X7FFFFFF ; Longest possible timeout interval
.LONG 0 ; PC of last releaser of multiple acquisitio
.PSECT EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_EMB-SPL$ _MIN_INDEX>
.ADDRESS SMP$GL_EMB ; Enter spinlock address in vector
.PSECT EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_EMB-SPL$ _MIN_INDEX>
.LONG IPL$ _EMB
```

384

```
385 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
386 ; THIS SPINLOCK VECTOR ENTRY CONTAINS THE ADDRESS OF THE EMB SPINLOCK.
387 ; THIS ENTRY POINT IS USED BY MACHINE CHECK AND OTHER HIGH IPL
388 ; ERROR CODE.
389 ;
```

```
390 SPINLOCK NAME=<MCHECK>, IPL=IPL$ _MCHECK, EQUATE=<EMB>
.PSECT EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_MCHECK-SPL$ _MIN_INDEX>
.ADDRESS SMP$GL_EMB ; Enter address of equivalent spinlock
.PSECT EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_MCHECK-SPL$ _MIN_INDEX>
.LONG IPL$ _MCHECK
```

391

392

```
393 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
394 ; KITCHEN SYNCH HEAVY DUTY SYNCHRONIZATION SPINLOCK
395 ;
```

```
396 SPINLOCK NAME=<MEGA>, IPL=IPL$ _MEGA
.PSECT EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_MEGA:: ; MEGA of spinlock
.BYTE 0 ; Structure not currently interlocked
.BYTE IPL$ _MEGA ; Synchronization level for lock
.BYTE $$$RANK ; RANK for lock
.BYTE 0 ; PC vector index
.WORD -1 ; Owned if positive
.WORD 0 ; Waiting CPUs
.WORD SPL$C_LENGTH ; Length of spinlock control block
.BYTE DYN$C_SPL ; Data structure type code
.BYTE SPL$C_SPL_SPINLOCK ; Spinlock subtype code
.LONG 0 ; Physical ID of owning CPU
.BLKL SPL$K_PC_VEC_CNT ; PC history of owning CPU
.LONG 0 ; PC of waiting CPU
.QUAD 0 ; Number of successful acquisitions
.LONG 0 ; Number of failed acquisitions
.QUAD 0 ; Number of spins
.LONG ^X7FFFFFF ; Longest possible timeout interval
.LONG 0 ; PC of last releaser of multiple acquisitio
.PSECT EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_MEGA-SPL$ _MIN_INDEX>
.ADDRESS SMP$GL_MEGA ; Enter spinlock address in vector
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 10  
X-35 SPINLOCK DATA STRUCTURES 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (2)

```
.PSECT EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
.=4*<SPL$C MEGA-SPL$ _MIN_INDEX>
.LONG IPL$ MEGA

397 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
398 ;          HARDWARE CLOCK LOCK - FOR MODIFYING THE HARDWARE TIMER INTERVAL
399 ;
400      SPINLOCK      NAME=<HWCLK>, IPL=IPL$ HWCLK
      .PSECT EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_HWCLK::          ; HWCLK of spinlock
      .BYTE 0          ; Structure not currently interlocked
      .BYTE IPL$ HWCLK          ; Synchronization level for lock
      .BYTE $$$RANK          ; RANK for lock
      .BYTE 0          ; PC vector index
      .WORD -1          ; Owned if positive
      .WORD 0          ; Waiting CPUs
      .WORD SPL$C_LENGTH          ; Length of spinlock control block
      .BYTE DYN$C_SPL          ; Data structure type code
      .BYTE SPL$C_SPL_SPINLOCK          ; Spinlock subtype code
      .LONG 0          ; Physical ID of owning CPU
      .BLKL SPL$K_PC_VEC_CNT          ; PC history of owning CPU
      .LONG 0          ; PC of waiting CPU
      .QUAD 0          ; Number of successful acquisitions
      .LONG 0          ; Number of failed acquisitions
      .QUAD 0          ; Number of spins
      .LONG ^X7FFFF          ; Longest possible timeout interval
      .LONG 0          ; PC of last releaser of multiple acquisitio
      .PSECT EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
      .=4*<SPL$C HWCLK-SPL$ _MIN_INDEX>
      .ADDRESS SMP$GL_HWCLK          ; Enter spinlock address in vector
      .PSECT EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
      .=4*<SPL$C HWCLK-SPL$ _MIN_INDEX>
      .LONG IPL$ HWCLK

401 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
402 ;          VIRTUAL CONSOLE SPINLOCK - OWNERSHIP OF VIRTUAL CONSOLE
403 ;
404      SPINLOCK      NAME=<VIRTCONS>, IPL=IPL$ VIRTCONS
      .PSECT EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_VIRTCONS::          ; VIRTCONS of spinlock
      .BYTE 0          ; Structure not currently interlocked
      .BYTE IPL$ VIRTCONS          ; Synchronization level for lock
      .BYTE $$$RANK          ; RANK for lock
      .BYTE 0          ; PC vector index
      .WORD -1          ; Owned if positive
      .WORD 0          ; Waiting CPUs
      .WORD SPL$C_LENGTH          ; Length of spinlock control block
      .BYTE DYN$C_SPL          ; Data structure type code
      .BYTE SPL$C_SPL_SPINLOCK          ; Spinlock subtype code
      .LONG 0          ; Physical ID of owning CPU
      .BLKL SPL$K_PC_VEC_CNT          ; PC history of owning CPU
      .LONG 0          ; PC of waiting CPU
      .QUAD 0          ; Number of successful acquisitions
      .LONG 0          ; Number of failed acquisitions
      .QUAD 0          ; Number of spins
      .LONG ^X7FFFF          ; Longest possible timeout interval
      .LONG 0          ; PC of last releaser of multiple acquisitio
      .PSECT EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
      .=4*<SPL$C VIRTCONS-SPL$ _MIN_INDEX>
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 11  
 X-35 SPINLOCK DATA STRUCTURES 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (2)

```

    .ADDRESS      SMP$GL_VIRTCONS ; Enter spinlock address in vector
    .PSECT EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
    .=4*<SPL$C_VIRTCONS-SPL$ _MIN_ INDEX>
    .LONG IPL$ _VIRTCONS
405 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
406 ;          TB INVALIDATE SINGLE SPINLOCK
407 ;
408 SPINLOCK      NAME=<INVALIDATE>, IPL=IPL$ _INVALIDATE
    .PSECT EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_INVALIDATE::
    ; INVALIDATE of spinlock
    .BYTE 0 ; Structure not currently interlocked
    .BYTE IPL$ _INVALIDATE ; Synchronization level for lock
    .BYTE $$$RANK ; RANK for lock
    .BYTE 0 ; PC vector index
    .WORD -1 ; Owned if positive
    .WORD 0 ; Waiting CPUs
    .WORD SPL$C_LENGTH ; Length of spinlock control block
    .BYTE DYN$C_SPL ; Data structure type code
    .BYTE SPL$C_SPL_SPINLOCK ; Spinlock subtype code
    .LONG 0 ; Physical ID of owning CPU
    .BLKL SPL$K_PC_VEC_CNT ; PC history of owning CPU
    .LONG 0 ; PC of waiting CPU
    .QUAD 0 ; Number of successful acquisitions
    .LONG 0 ; Number of failed acquisitions
    .QUAD 0 ; Number of spins
    .LONG ^X7FFFFFF ; Longest possible timeout interval
    .LONG 0 ; PC of last releaser of multiple acquisitio
    .PSECT EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
    .=4*<SPL$C_INVALIDATE-SPL$ _MIN_ INDEX>
    .ADDRESS      SMP$GL_INVALIDATE ; Enter spinlock address in vector
    .PSECT EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
    .=4*<SPL$C_INVALIDATE-SPL$ _MIN_ INDEX>
    .LONG IPL$ _INVALIDATE
409 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
410 ;          PERFORMANCE MONITOR LOCK
411 ;
412 SPINLOCK      NAME=<PERFMON>, IPL=IPL$ _PERFMON
    .PSECT EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_PERFMON::
    ; PERFMON of spinlock
    .BYTE 0 ; Structure not currently interlocked
    .BYTE IPL$ _PERFMON ; Synchronization level for lock
    .BYTE $$$RANK ; RANK for lock
    .BYTE 0 ; PC vector index
    .WORD -1 ; Owned if positive
    .WORD 0 ; Waiting CPUs
    .WORD SPL$C_LENGTH ; Length of spinlock control block
    .BYTE DYN$C_SPL ; Data structure type code
    .BYTE SPL$C_SPL_SPINLOCK ; Spinlock subtype code
    .LONG 0 ; Physical ID of owning CPU
    .BLKL SPL$K_PC_VEC_CNT ; PC history of owning CPU
    .LONG 0 ; PC of waiting CPU
    .QUAD 0 ; Number of successful acquisitions
    .LONG 0 ; Number of failed acquisitions
    .QUAD 0 ; Number of spins
    .LONG ^X7FFFFFF ; Longest possible timeout interval
    .LONG 0 ; PC of last releaser of multiple acquisitio
    .PSECT EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
  
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 12  
X-35 SPINLOCK DATA STRUCTURES 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (2)

```

.=4*<SPL$C_PERFMON-SPL$ _MIN_INDEX>
.ADDRESS      SMP$GL_PERFMON ; Enter spinlock address in vector
.PSECT EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_PERFMON-SPL$ _MIN_INDEX>
.LONG IPL$ _PERFMON
413 ///////////////////////////////////////////////////
414 ;          NON-PAGED POOL DATABASE LOCK
415 ;
416          SPINLOCK          NAME=<POOL>, IPL=IPL$ _POOL
          .PSECT EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_POOL::
          .BYTE 0 ; POOL of spinlock
          .BYTE IPL$ _POOL ; Structure not currently interlocked
          .BYTE $$$RANK ; Synchronization level for lock
          .BYTE 0 ; RANK for lock
          .WORD -1 ; PC vector index
          .WORD 0 ; Owned if positive
          .WORD 0 ; Waiting CPUs
          .WORD SPL$C_LENGTH ; Length of spinlock control block
          .BYTE DYN$C_SPL ; Data structure type code
          .BYTE SPL$C_SPL_SPINLOCK ; Spinlock subtype code
          .LONG 0 ; Physical ID of owning CPU
          .BLKL SPL$K_PC_VEC_CNT ; PC history of owning CPU
          .LONG 0 ; PC of waiting CPU
          .QUAD 0 ; Number of successful acquisitions
          .LONG 0 ; Number of failed acquisitions
          .QUAD 0 ; Number of spins
          .LONG ^X7FFFF ; Longest possible timeout interval
          .LONG 0 ; PC of last releaser of multiple acquisitio
          .PSECT EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
          .=4*<SPL$C_POOL-SPL$ _MIN_INDEX>
          .ADDRESS      SMP$GL_POOL ; Enter spinlock address in vector
          .PSECT EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
          .=4*<SPL$C_POOL-SPL$ _MIN_INDEX>
          .LONG IPL$ _POOL
417 ///////////////////////////////////////////////////
418 ;          MAILBOX LOCK
419 ;
420          SPINLOCK          NAME=<MAILBOX>, IPL=IPL$ _MAILBOX
          .PSECT EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_MAILBOX::
          .BYTE 0 ; MAILBOX of spinlock
          .BYTE IPL$ _MAILBOX ; Structure not currently interlocked
          .BYTE $$$RANK ; Synchronization level for lock
          .BYTE 0 ; RANK for lock
          .WORD -1 ; PC vector index
          .WORD 0 ; Owned if positive
          .WORD 0 ; Waiting CPUs
          .WORD SPL$C_LENGTH ; Length of spinlock control block
          .BYTE DYN$C_SPL ; Data structure type code
          .BYTE SPL$C_SPL_SPINLOCK ; Spinlock subtype code
          .LONG 0 ; Physical ID of owning CPU
          .BLKL SPL$K_PC_VEC_CNT ; PC history of owning CPU
          .LONG 0 ; PC of waiting CPU
          .QUAD 0 ; Number of successful acquisitions
          .LONG 0 ; Number of failed acquisitions
          .QUAD 0 ; Number of spins
          .LONG ^X7FFFF ; Longest possible timeout interval
          .LONG 0 ; PC of last releaser of multiple acquisitio

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 13  
X-35 SPINLOCK DATA STRUCTURES 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (2)

```
.PSECT EXEC$$SMP_BBB, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_MAILBOX-SPL$ MIN_INDEX>
.ADDRESS SMP$GL_MAILBOX ; Enter spinlock address in vector
.PSECT EXEC$$SMP_CCC, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_MAILBOX-SPL$ MIN_INDEX>
.LONG IPL$_MAILBOX

421 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
422 ; IPL 11 PRIMARY CPU FORK LOCK
423 ;
424 SPINLOCK NAME=<PR_LK11>, IPL=IPL$ PR_LK11, LOCKTYPE=FORKLOCK
.PSECT EXEC$$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_PR_LK11:: ; PR_LK11 of spinlock
.BYTE 0 ; Structure not currently interlocked
.BYTE IPL$ PR_LK11 ; Synchronization level for lock
.BYTE $$$RANK ; RANK for lock
.BYTE 0 ; PC vector index
.WORD -1 ; Owned if positive
.WORD 0 ; Waiting CPUs
.WORD SPL$C_LENGTH ; Length of spinlock control block
.BYTE DYN$C_SPL ; Data structure type code
.BYTE SPL$C_SPL_FORKLOCK ; Spinlock subtype code
.LONG 0 ; Physical ID of owning CPU
.BLKL SPL$K_PC_VEC_CNT ; PC history of owning CPU
.LONG 0 ; PC of waiting CPU
.QUAD 0 ; Number of successful acquisitions
.LONG 0 ; Number of failed acquisitions
.QUAD 0 ; Number of spins
.LONG ^X7FFFFFF ; Longest possible timeout interval
.LONG 0 ; PC of last releaser of multiple acquisitio
.PSECT EXEC$$SMP_BBB, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_PR_LK11-SPL$ MIN_INDEX>
.ADDRESS SMP$GL_PR_LK11 ; Enter spinlock address in vector
.PSECT EXEC$$SMP_CCC, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_PR_LK11-SPL$ MIN_INDEX>
.LONG IPL$ PR_LK11

425 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
426 ; IPL 11 I/O FORK QUEUE LOCK
427 ;
428 SPINLOCK NAME=<IOLOCK11>, IPL=IPL$ IOLOCK11, LOCKTYPE=FORKLOCK
.PSECT EXEC$$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_IOLOCK11:: ; IOLOCK11 of spinlock
.BYTE 0 ; Structure not currently interlocked
.BYTE IPL$ IOLOCK11 ; Synchronization level for lock
.BYTE $$$RANK ; RANK for lock
.BYTE 0 ; PC vector index
.WORD -1 ; Owned if positive
.WORD 0 ; Waiting CPUs
.WORD SPL$C_LENGTH ; Length of spinlock control block
.BYTE DYN$C_SPL ; Data structure type code
.BYTE SPL$C_SPL_FORKLOCK ; Spinlock subtype code
.LONG 0 ; Physical ID of owning CPU
.BLKL SPL$K_PC_VEC_CNT ; PC history of owning CPU
.LONG 0 ; PC of waiting CPU
.QUAD 0 ; Number of successful acquisitions
.LONG 0 ; Number of failed acquisitions
.QUAD 0 ; Number of spins
.LONG ^X7FFFFFF ; Longest possible timeout interval
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

LDAT - SPIN LOCK DATA BASES /10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 14  
X-35 SPINLOCK DATA STRUCTURES 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (2)

```

.LONG 0 ; PC of last releaser of multiple acquisitio
.PSECT EXEC$SMP BBB, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_IOLOCK11-SPL$ MIN_INDEX>
.ADDRESS SMP$GL_IOLOCK11 ; Enter spinlock address in vector
.PSECT EXEC$SMP CCC, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_IOLOCK11-SPL$ MIN_INDEX>
.LONG IPL$ IOLOCK11
429 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
430 ; IPL 10 PRIMARY CPU FORK LOCK
431 ;
432 SPINLOCK NAME=<PR LK10>, IPL=IPL$ PR_LK10, LOCKTYPE=FORKLOCK
.PSECT EXEC$SMP DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_PR_LK10:: ; PR_LK10 of spinlock
.BYTE 0 ; Structure not currently interlocked
.BYTE IPL$ PR_LK10 ; Synchronization level for lock
.BYTE $$$RANK ; RANK for lock
.BYTE 0 ; PC vector index
.WORD -1 ; Owned if positive
.WORD 0 ; Waiting CPUs
.WORD SPL$C_LENGTH ; Length of spinlock control block
.BYTE DYN$C_SPL ; Data structure type code
.BYTE SPL$C_SPL_FORKLOCK ; Spinlock subtype code
.LONG 0 ; Physical ID of owning CPU
.BLKL SPL$K_PC_VEC_CNT ; PC history of owning CPU
.LONG 0 ; PC of waiting CPU
.QUAD 0 ; Number of successful acquisitions
.LONG 0 ; Number of failed acquisitions
.QUAD 0 ; Number of spins
.LONG ^X7FFFFF ; Longest possible timeout interval
.LONG 0 ; PC of last releaser of multiple acquisitio
.PSECT EXEC$SMP BBB, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_PR_LK10-SPL$ MIN_INDEX>
.ADDRESS SMP$GL_PR_LK10 ; Enter spinlock address in vector
.PSECT EXEC$SMP CCC, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_PR_LK10-SPL$ MIN_INDEX>
.LONG IPL$ PR_LK10
433 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
434 ; IPL 10 I/O FORK QUEUE LOCK
435 ;
436 SPINLOCK NAME=<IOLOCK10>, IPL=IPL$ IOLOCK10, LOCKTYPE=FORKLOCK
.PSECT EXEC$SMP DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_IOLOCK10:: ; IOLOCK10 of spinlock
.BYTE 0 ; Structure not currently interlocked
.BYTE IPL$ IOLOCK10 ; Synchronization level for lock
.BYTE $$$RANK ; RANK for lock
.BYTE 0 ; PC vector index
.WORD -1 ; Owned if positive
.WORD 0 ; Waiting CPUs
.WORD SPL$C_LENGTH ; Length of spinlock control block
.BYTE DYN$C_SPL ; Data structure type code
.BYTE SPL$C_SPL_FORKLOCK ; Spinlock subtype code
.LONG 0 ; Physical ID of owning CPU
.BLKL SPL$K_PC_VEC_CNT ; PC history of owning CPU
.LONG 0 ; PC of waiting CPU
.QUAD 0 ; Number of successful acquisitions
.LONG 0 ; Number of failed acquisitions
.QUAD 0 ; Number of spins

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 15  
X-35 SPINLOCK DATA STRUCTURES 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (2)

```

.LONG ^X7FFFF ; Longest possible timeout interval
.LONG 0 ; PC of last releaser of multiple acquisitio
.PSECT EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_IOLOCK10-SPL$ MIN_INDEX>
.ADDRESS SMP$GL_IOLOCK10 ; Enter spinlock address in vector
.PSECT EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_IOLOCK10-SPL$ MIN_INDEX>
.LONG IPL$ IOLOCK10
437 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
438 ; IPL 9 PRIMARY CPU FORK LOCK
439 ;
440 SPINLOCK NAME=<PR_LK9>, IPL=IPL$ PR_LK9, LOCKTYPE=FORKLOCK
.PSECT EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_PR_LK9:: ; PR_LK9 of spinlock
.BYTE 0 ; Structure not currently interlocked
.BYTE IPL$ PR_LK9 ; Synchronization level for lock
.BYTE $$$RANK ; RANK for lock
.BYTE 0 ; PC vector index
.WORD -1 ; Owned if positive
.WORD 0 ; Waiting CPUs
.WORD SPL$C_LENGTH ; Length of spinlock control block
.BYTE DYN$C_SPL ; Data structure type code
.BYTE SPL$C_SPL_FORKLOCK ; Spinlock subtype code
.LONG 0 ; Physical ID of owning CPU
.BLKL SPL$K_PC_VEC_CNT ; PC history of owning CPU
.LONG 0 ; PC of waiting CPU
.QUAD 0 ; Number of successful acquisitions
.LONG 0 ; Number of failed acquisitions
.QUAD 0 ; Number of spins
.LONG ^X7FFFF ; Longest possible timeout interval
.LONG 0 ; PC of last releaser of multiple acquisitio
.PSECT EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_PR_LK9-SPL$ MIN_INDEX>
.ADDRESS SMP$GL_PR_LK9 ; Enter spinlock address in vector
.PSECT EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
.=4*<SPL$C PR_LK9-SPL$ MIN_INDEX>
.LONG IPL$ PR_LK9
441 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
442 ; IPL 9 I/O FORK QUEUE LOCK
443 ;
444 SPINLOCK NAME=<IOLOCK9>, IPL=IPL$ IOLOCK9, LOCKTYPE=FORKLOCK
.PSECT EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_IOLOCK9:: ; IOLOCK9 of spinlock
.BYTE 0 ; Structure not currently interlocked
.BYTE IPL$ IOLOCK9 ; Synchronization level for lock
.BYTE $$$RANK ; RANK for lock
.BYTE 0 ; PC vector index
.WORD -1 ; Owned if positive
.WORD 0 ; Waiting CPUs
.WORD SPL$C_LENGTH ; Length of spinlock control block
.BYTE DYN$C_SPL ; Data structure type code
.BYTE SPL$C_SPL_FORKLOCK ; Spinlock subtype code
.LONG 0 ; Physical ID of owning CPU
.BLKL SPL$K_PC_VEC_CNT ; PC history of owning CPU
.LONG 0 ; PC of waiting CPU
.QUAD 0 ; Number of successful acquisitions
.LONG 0 ; Number of failed acquisitions

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 16  
X-35 SPINLOCK DATA STRUCTURES 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (2)

```

        .QUAD      0                ; Number of spins
        .LONG      ^X7FFFFFFF      ; Longest possible timeout interval
        .LONG      0                ; PC of last releaser of multiple acquisitio
        .PSECT    EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
        .=4*<SPL$C_IOLOCK9-SPL$ _MIN_INDEX>
        .ADDRESS   SMP$GL_IOLOCK9  ; Enter spinlock address in vector
        .PSECT    EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
        .=4*<SPL$C_IOLOCK9-SPL$ _MIN_INDEX>
        .LONG      IPL$ _IOLOCK9

445 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
446 ;                SCHEDULING DATABASE AND MUTEX LOCK
447 ;
448      SPINLOCK      NAME=<SCHED>, IPL=IPL$ _SCHED
        .PSECT    EXEC$SMP _DDD, QUAD, WRT, NOEXE, PIC
SMP$GL _SCHED::      ; SCHED of spinlock
        .BYTE     0                ; Structure not currently interlocked
        .BYTE     IPL$ _SCHED      ; Synchronization level for lock
        .BYTE     $$$RANK          ; RANK for lock
        .BYTE     0                ; PC vector index
        .WORD     -1              ; Owned if positive
        .WORD     0                ; Waiting CPUs
        .WORD     SPL$C _LENGTH    ; Length of spinlock control block
        .BYTE     DYN$C _SPL       ; Data structure type code
        .BYTE     SPL$C _SPL _SPINLOCK ; Spinlock subtype code
        .LONG     0                ; Physical ID of owning CPU
        .BLKL     SPL$K _PC _VEC _CNT ; PC history of owning CPU
        .LONG     0                ; PC of waiting CPU
        .QUAD     0                ; Number of successful acquisitions
        .LONG     0                ; Number of failed acquisitions
        .QUAD     0                ; Number of spins
        .LONG     ^X7FFFFFFF      ; Longest possible timeout interval
        .LONG     0                ; PC of last releaser of multiple acquisitio
        .PSECT    EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
        .=4*<SPL$C _SCHED-SPL$ _MIN_INDEX>
        .ADDRESS   SMP$GL _SCHED  ; Enter spinlock address in vector
        .PSECT    EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
        .=4*<SPL$C _SCHED-SPL$ _MIN_INDEX>
        .LONG      IPL$ _SCHED

449 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
450 ;                MEMORY MANAGEMENT, PFN DATABASE, SWAPPER,
451 ;                MODIFIED PAGE WRITER, SYSWS
452 ;
453      SPINLOCK      NAME=<MMG>, IPL=IPL$ _MMG
        .PSECT    EXEC$SMP _DDD, QUAD, WRT, NOEXE, PIC
SMP$GL _MMG::      ; MMG of spinlock
        .BYTE     0                ; Structure not currently interlocked
        .BYTE     IPL$ _MMG       ; Synchronization level for lock
        .BYTE     $$$RANK          ; RANK for lock
        .BYTE     0                ; PC vector index
        .WORD     -1              ; Owned if positive
        .WORD     0                ; Waiting CPUs
        .WORD     SPL$C _LENGTH    ; Length of spinlock control block
        .BYTE     DYN$C _SPL       ; Data structure type code
        .BYTE     SPL$C _SPL _SPINLOCK ; Spinlock subtype code
        .LONG     0                ; Physical ID of owning CPU
        .BLKL     SPL$K _PC _VEC _CNT ; PC history of owning CPU
        .LONG     0                ; PC of waiting CPU

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 17  
X-35 SPINLOCK DATA STRUCTURES 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (2)

```
.QUAD 0 ; Number of successful acquisitions
.LONG 0 ; Number of failed acquisitions
.QUAD 0 ; Number of spins
.LONG ^X7FFFFFF ; Longest possible timeout interval
.LONG 0 ; PC of last releaser of multiple acquisitio
.PSECT EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_MMG-SPL$ _MIN_INDEX>
.ADDRESS SMP$GL_MMG ; Enter spinlock address in vector
.PSECT EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_MMG-SPL$ _MIN_INDEX>
.LONG IPL$ _MMG
```

```
454 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
455 ; JIB LOCK - LOCKS ACCESS TO BYTCNT AND BYTLM (RELATED
456 ; TO NON-PAGED POOL) IN THE JIB
457 ;
458 SPINLOCK NAME=<JIB>, IPL=IPL$ _JIB
```

```
.PSECT EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL _JIB:: ; JIB of spinlock
.BYTE 0 ; Structure not currently interlocked
.BYTE IPL$ _JIB ; Synchronization level for lock
.BYTE $$RANK ; RANK for lock
.BYTE 0 ; PC vector index
.WORD -1 ; Owned if positive
.WORD 0 ; Waiting CPUs
.WORD SPL$C_LENGTH ; Length of spinlock control block
.BYTE DYN$C_SPL ; Data structure type code
.BYTE SPL$C_SPL_SPINLOCK ; Spinlock subtype code
.LONG 0 ; Physical ID of owning CPU
.BKLN SPL$K_PC_VEC_CNT ; PC history of owning CPU
.LONG 0 ; PC of waiting CPU
.QUAD 0 ; Number of successful acquisitions
.LONG 0 ; Number of failed acquisitions
.QUAD 0 ; Number of spins
.LONG ^X7FFFFFF ; Longest possible timeout interval
.LONG 0 ; PC of last releaser of multiple acquisitio
.PSECT EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
.=4*<SPL$C _JIB-SPL$ _MIN_INDEX>
.ADDRESS SMP$GL _JIB ; Enter spinlock address in vector
.PSECT EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
.=4*<SPL$C _JIB-SPL$ _MIN_INDEX>
.LONG IPL$ _JIB
```

```
459 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
460 ; TIMER QUEUE LOCK - FOR ADDING/DELETING/SEARCHING TIMER QUEUE
461 ;
462 SPINLOCK NAME=<TIMER>, IPL=IPL$ _TIMER
```

```
.PSECT EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL _TIMER:: ; TIMER of spinlock
.BYTE 0 ; Structure not currently interlocked
.BYTE IPL$ _TIMER ; Synchronization level for lock
.BYTE $$RANK ; RANK for lock
.BYTE 0 ; PC vector index
.WORD -1 ; Owned if positive
.WORD 0 ; Waiting CPUs
.WORD SPL$C_LENGTH ; Length of spinlock control block
.BYTE DYN$C_SPL ; Data structure type code
.BYTE SPL$C_SPL_SPINLOCK ; Spinlock subtype code
.LONG 0 ; Physical ID of owning CPU
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 18  
X-35 SPINLOCK DATA STRUCTURES 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (2)

```

        .BLKL   SPL$K_PC_VEC_CNT           ; PC history of owning CPU
        .LONG   0                         ; PC of waiting CPU
        .QUAD   0                         ; Number of successful acquisitions
        .LONG   0                         ; Number of failed acquisitions
        .QUAD   0                         ; Number of spins
        .LONG   ^X7FFFFFFF                ; Longest possible timeout interval
        .LONG   0                         ; PC of last releaser of multiple acquisitio
        .PSECT  EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
        .=4*<SPL$C_TIMER-SPL$ _MIN_INDEX>
        .ADDRESS SMP$GL_TIMER             ; Enter spinlock address in vector
        .PSECT  EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
        .=4*<SPL$C_TIMER-SPL$ _MIN_INDEX>
        .LONG   IPL$ _TIMER

463 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
464 ;                               IPL 8 PRIMARY CPU FORK LOCK
465 ;
466      SPINLOCK      NAME=<PR_LK8>, IPL=IPL$ _PR_LK8, LOCKTYPE=FORKLOCK
        .PSECT  EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_PR_LK8::                               ; PR_LK8 of spinlock
        .BYTE   0                         ; Structure not currently interlocked
        .BYTE   IPL$ _PR_LK8              ; Synchronization level for lock
        .BYTE   $$$RANK                   ; RANK for lock
        .BYTE   0                         ; PC vector index
        .WORD   -1                        ; Owned if positive
        .WORD   0                         ; Waiting CPUs
        .WORD   SPL$C_LENGTH               ; Length of spinlock control block
        .BYTE   DYN$C_SPL                  ; Data structure type code
        .BYTE   SPL$C_SPL_FORKLOCK        ; Spinlock subtype code
        .LONG   0                         ; Physical ID of owning CPU
        .BLKL   SPL$K_PC_VEC_CNT           ; PC history of owning CPU
        .LONG   0                         ; PC of waiting CPU
        .QUAD   0                         ; Number of successful acquisitions
        .LONG   0                         ; Number of failed acquisitions
        .QUAD   0                         ; Number of spins
        .LONG   ^X7FFFFFFF                ; Longest possible timeout interval
        .LONG   0                         ; PC of last releaser of multiple acquisitio
        .PSECT  EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
        .=4*<SPL$C_PR_LK8-SPL$ _MIN_INDEX>
        .ADDRESS SMP$GL_PR_LK8           ; Enter spinlock address in vector
        .PSECT  EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
        .=4*<SPL$C_PR_LK8-SPL$ _MIN_INDEX>
        .LONG   IPL$ _PR_LK8

467 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
468 ;                               IPL 8 I/O FORK QUEUE LOCK & SCS LOCK
469 ;
470      SPINLOCK      NAME=<IOLOCKS>, IPL=IPL$ _IOLOCKS, LOCKTYPE=FORKLOCK
        .PSECT  EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_IOLOCKS::                               ; IOLOCKS of spinlock
        .BYTE   0                         ; Structure not currently interlocked
        .BYTE   IPL$ _IOLOCKS             ; Synchronization level for lock
        .BYTE   $$$RANK                   ; RANK for lock
        .BYTE   0                         ; PC vector index
        .WORD   -1                        ; Owned if positive
        .WORD   0                         ; Waiting CPUs
        .WORD   SPL$C_LENGTH               ; Length of spinlock control block
        .BYTE   DYN$C_SPL                  ; Data structure type code
        .BYTE   SPL$C_SPL_FORKLOCK        ; Spinlock subtype code

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 19  
X-35 SPINLOCK DATA STRUCTURES 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (2)

```

.LONG 0 ; Physical ID of owning CPU
.BKLN SPL$K_PC_VEC_CNT ; PC history of owning CPU
.LONG 0 ; PC of waiting CPU
.QUAD 0 ; Number of successful acquisitions
.LONG 0 ; Number of failed acquisitions
.QUAD 0 ; Number of spins
.LONG ^X7FFFFFF ; Longest possible timeout interval
.LONG 0 ; PC of last releaser of multiple acquisitio
.PSECT EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_IOLOCK8-SPL$ MIN_INDEX>
.ADDRESS SMP$GL_IOLOCK8 ; Enter spinlock address in vector
.PSECT EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_IOLOCK8-SPL$ MIN_INDEX>
.LONG IPL$ _IOLOCK8

```

471 ;  
472 ; THE FILE SYSTEM LOCK  
473 ;

```

474 SPINLOCK NAME=<FILSYS>, IPL=IPL$ _FILSYS
.PSECT EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_FILSYS:: ; FILSYS of spinlock
.BYTE 0 ; Structure not currently interlocked
.BYTE IPL$ _FILSYS ; Synchronization level for lock
.BYTE $$$RANK ; RANK for lock
.BYTE 0 ; PC vector index
.WORD -1 ; Owned if positive
.WORD 0 ; Waiting CPUs
.WORD SPL$C_LENGTH ; Length of spinlock control block
.BYTE DYN$C_SPL ; Data structure type code
.BYTE SPL$C_SPL_SPINLOCK ; Spinlock subtype code
.LONG 0 ; Physical ID of owning CPU
.BKLN SPL$K_PC_VEC_CNT ; PC history of owning CPU
.LONG 0 ; PC of waiting CPU
.QUAD 0 ; Number of successful acquisitions
.LONG 0 ; Number of failed acquisitions
.QUAD 0 ; Number of spins
.LONG ^X7FFFFFF ; Longest possible timeout interval
.LONG 0 ; PC of last releaser of multiple acquisitio
.PSECT EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_FILSYS-SPL$ MIN_INDEX>
.ADDRESS SMP$GL_FILSYS ; Enter spinlock address in vector
.PSECT EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
.=4*<SPL$C_FILSYS-SPL$ MIN_INDEX>
.LONG IPL$ _FILSYS

```

475 ;  
476 ; QUEUEAST FORK QUEUE LOCK  
477 ;

```

478 SPINLOCK NAME=<QUEUEAST>, IPL=IPL$ _QUEUEAST
.PSECT EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_QUEUEAST:: ; QUEUEAST of spinlock
.BYTE 0 ; Structure not currently interlocked
.BYTE IPL$ _QUEUEAST ; Synchronization level for lock
.BYTE $$$RANK ; RANK for lock
.BYTE 0 ; PC vector index
.WORD -1 ; Owned if positive
.WORD 0 ; Waiting CPUs
.WORD SPL$C_LENGTH ; Length of spinlock control block
.BYTE DYN$C_SPL ; Data structure type code

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 20  
X-35 SPINLOCK DATA STRUCTURES 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (2)

```

        .BYTE    SPL$C_SPL_SPINLOCK      ; Spinlock subtype code
        .LONG    0                       ; Physical ID of owning CPU
        .BLKL   SPL$K_PC_VEC_CNT        ; PC history of owning CPU
        .LONG    0                       ; PC of waiting CPU
        .QUAD    0                       ; Number of successful acquisitions
        .LONG    0                       ; Number of failed acquisitions
        .QUAD    0                       ; Number of spins
        .LONG    ^X7FFFFFF              ; Longest possible timeout interval
        .LONG    0                       ; PC of last releaser of multiple acquisitio
        .PSECT  EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
        .=4*<SPL$C_QUEUEAST-SPL$ MIN_INDEX>
        .ADDRESS SMP$GL_QUEUEAST ; Enter spinlock address in vector
        .PSECT  EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
        .=4*<SPL$C_QUEUEAST-SPL$ MIN_INDEX>
        .LONG    IPL$_QUEUEAST

479 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
480 ; temporary lock - this lock DOES NOT PROVIDE MULTIPROCESSOR SYNCHRONIZATION
481 ; because it has an IPL that doesn't prevent rescheduling. This lock
482 ; is defined solely for early versions of UIS which need something to
483 ; work in the first version of VMS that supports SMP.
484 ;
485 ; It is expected that this lock can be removed when UIS doesn't
486 ; need it anymore.
487 ;
488 SPINLOCK          NAME=<ASTDEL>, IPL=2
        .PSECT  EXEC$SMP_DDD, QUAD, WRT, NOEXE, PIC
SMP$GL_ASTDEL::    ; ASTDEL of spinlock
        .BYTE    0                       ; Structure not currently interlocked
        .BYTE    2                       ; Synchronization level for lock
        .BYTE    $$$RANK                 ; RANK for lock
        .BYTE    0                       ; PC vector index
        .WORD    -1                      ; Owned if positive
        .WORD    0                       ; Waiting CPUs
        .WORD    SPL$C_LENGTH            ; Length of spinlock control block
        .BYTE    DYN$C_SPL               ; Data structure type code
        .BYTE    SPL$C_SPL_SPINLOCK      ; Spinlock subtype code
        .LONG    0                       ; Physical ID of owning CPU
        .BLKL   SPL$K_PC_VEC_CNT        ; PC history of owning CPU
        .LONG    0                       ; PC of waiting CPU
        .QUAD    0                       ; Number of successful acquisitions
        .LONG    0                       ; Number of failed acquisitions
        .QUAD    0                       ; Number of spins
        .LONG    ^X7FFFFFF              ; Longest possible timeout interval
        .LONG    0                       ; PC of last releaser of multiple acquisitio
        .PSECT  EXEC$SMP_BBB, LONG, WRT, NOEXE, PIC
        .=4*<SPL$C_ASTDEL-SPL$ MIN_INDEX>
        .ADDRESS SMP$GL_ASTDEL ; Enter spinlock address in vector
        .PSECT  EXEC$SMP_CCC, LONG, WRT, NOEXE, PIC
        .=4*<SPL$C_ASTDEL-SPL$ MIN_INDEX>
        .LONG    2

```

489

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

LDAT - SPIN LOCK DATA BASES 10-MAY-1989 16:57:58 VAX MACRO V5.0-8 Page 21  
X-35 SPINLOCK DATA STRUCTURES 23-NOV-1988 14:12:56 \_\$254\$DUA55:[SYS.SRC]LDAT.MAR;1 (3)

```
491 ;+
492 ; ASSUMPTIONS
493 ;
494 ;-
495
496 ; Make sure that MMG and SCHED are adjacent to match checks done
497 ; in MEMORYALC to allow for pool expansion.
498
499 ...RANK_MMG = ...RANK_SCHED+1
500
501
502 ; Make sure that all locks fall in the range 20 to 3E (hex). (Remember
503 ; to reserve one index for dynamic spinlocks). This makes code paths
504 ; like FORKCNTRL be able to test bit #5 to see if it is a FORKLOCK index
505 ; in the FKBSB_FLCK or an IPL (range 0-1F). The test of bit #5 is
506 ; also done in the FORKLOCK and FORKUNLOCK macros in certain instances.
507
508 ASSUME SPL$ _MIN_INDEX GE 32
509 ASSUME SPL$ _MAX_INDEX LE 62
510
511
512 ; Make sure that there aren't more spinlocks than table can handle.
513 ; NUM_LOCKS accounts for the 1 reservation for the dynamic spinlocks -
514
515 ; and is only equal to 31.
516
517 ASSUME <SPL$ _MAX_INDEX-SPL$ _MIN_INDEX> LE SPL$ _NUM_LOCKS
518 ASSUME SPL$ _NUM_LOCKS LE 31
519
520 .END
```

## 8 MMDAT.LIS

MMDAT 10-MAY-1989 17:04:31 VAX MACRO V5.0-8 Page 0  
Table of contents

(3)        70        MMDAT\_INIT



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

MMDAT 10-MAY-1989 17:04:31 VAX MACRO V5.0-8 Page 1  
X-9 28-AUG-1986 10:34:47 \_\$254\$DUA55:[SYS.SRC]MMDAT.MAR;1 (1)

```
1 .TITLE MMDAT
2 .IDENT 'X-9'
3 ;
4 ;*****
5 ;*
6 ;* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
7 ;* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
8 ;* ALL RIGHTS RESERVED.
9 ;*
10 ;* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
11 ;* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
12 ;* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
13 ;* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
14 ;* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
15 ;* TRANSFERRED.
16 ;*
17 ;* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
18 ;* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
19 ;* CORPORATION.
20 ;*
21 ;* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
22 ;* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
23 ;*
24 ;*
25 ;*****
26 ;
27 ; AUTHOR: Trudy C. Matthews          CREATION DATE: 25-MAY-1985
28 ;
29 ; FUNCTIONAL DESCRIPTION:
30 ;     This module contains read-only cells that are referenced by time-
31 ;     critical code in the system. All cells defined in this module can
32 ;     be referenced locally by any image which links this module into
33 ;     itself.
34 ;
35 ; ENVIRONMENT:
36 ;
37 ; MODIFIED BY:
38 ;
39 ;     X-9      SSA0001      Stan Amway          28-Aug-1986
40 ;     Remove definitions of and references to MPW$AL_PTE
41 ;     and MPW$AW_PHVINDEX.
42 ;
43 ;     X-8      WMC0003      Wayne Cardoza        29-Jul-1986
44 ;     get rid of re-executeable flag.
45 ;
46 ;     X-7      WMC0002      Wayne Cardoza        18-Jun-1986
47 ;     Initialization routines should return status.
48 ;
49 ;     X-6      WMC0001      Wayne Cardoza        12-Feb-1985
50 ;     remove writeable modified page writer cells.
51 ;
52 ;     X-5      TCM0003      Trudy C. Matthews    21-Nov-1985
53 ;     Remove LNM$AL_HASHTBL, as it is the address of a TABLE and
54 ;     the entire table would have to be moved, not just the
55 ;     first longword.
56 ;
57 ;     V04-002 TCM0002      Trudy C. Matthews    3-Oct-1985
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

MMDAT 10-MAY-1989 17:04:31 VAX MACRO V5.0-8 Page 2  
X-9 28-AUG-1986 10:34:47 \_\$254\$DUA55:[SYS.SRC]MMDAT.MAR;1 (1)

```
58 ;           Remove PFNSAL_HEAD, PFNSAL_TAIL, PFNSAL_COUNT, and
59 ;           MPW$GL_BADPAGTOTAL cells as they are not read-only.
60 ;           LNM_AL_HASHTBL is more than a word offset, so do some
61 ;           arithmetic to get the address calculation right.
62 ;
```

MMDAT 10-MAY-1989 17:04:31 VAX MACRO V5.0-8 Page 3  
X-9 28-AUG-1986 10:34:47 \_\$254\$DUA55:[SYS.SRC]MMDAT.MAR;1 (2)

64 ;  
65 ; MACRO LIBRARY CALLS  
66 ;  
67       \$BOOSTATEDEF  
68 ;

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

MMDAT 10-MAY-1989 17:04:31 VAX MACRO V5.0-8 Page 4  
X-9 MMDAT\_INIT 28-AUG-1986 10:34:47 \_\$254\$DUA55:[SYS.SRC]MMDAT.MAR;1 (3)

```
70      .SBTTL  MMDAT_INIT
71 ;
72 ; When this image is loaded, copy data from the cells in the exec into
73 ; these local copies.
74 ;
75 ; Inputs:
76 ;     None
77 ;
78 ; Implicit inputs:
79 ;     R0-R3 available as scratch registers.
80 ;
81 ; Outputs:
82 ;     System data cells copied into local image.
83 ;
84
85      DECLARE_PSECT  EXEC$INIT_CODE
86
87  INITIALIZATION ROUTINE -
88      MMDAT_INIT
89
90  MMDAT_INIT::
91      BBS      #BOOSTATE$V_SWAPPER,-      ; Some cells aren't initialized until
92              G^EXE$GL_STATE,10$        ; INIT is done running
93      BICL    #INIRTN$M_NO_RECALL,(R5); Keep on trying
94
95  10$:      MOVAB  G^SGN$A_COMPVALUES,R0      ; Get base of SYSGEN computed values.
96      MOVL    PFN_AB_STATE(R0),W^PFN$AB_STATE
97      MOVL    PFN_AB_TYPE(R0),W^PFN$AB_TYPE
98      MOVL    PFN_AL_BAK(R0),W^PFN$AL_BAK
99      MOVL    PFN_AL_PTE(R0),W^PFN$AL_PTE
100     MOVL    PFN_AW_REFCNT(R0),W^PFN$AW_REFCNT
101     MOVL    PFN_AW_SWPVBN(R0),W^PFN$AW_SWPVBN
102     MOVL    PFN_AX_BLINK(R0),W^PFN$AX_BLINK
103     MOVL    PFN_AX_FLINK(R0),W^PFN$AX_FLINK
104     MOVL    PFN_AX_SHRCNT(R0),W^PFN$AX_SHRCNT
105     MOVL    PFN_AX_WSLX(R0),W^PFN$AX_WSLX
106     MOVL    MMG_GL_GPTBASE(R0),W^MMG$GL_GPTBASE
107     MOVL    MMG_GL_SPTBASE(R0),W^MMG$GL_SPTBASE
108     MOVL    SWP_GL_BALSPT(R0),W^SWP$GL_BALSPT
109     MOVL    SWP_GL_BALBASE(R0),W^SWP$GL_BALBASE
110
111     MOVAB  G^EXEC$SYSDATA_CELLS,R0      ; Get base of SYSDATA cells.
112     MOVL    PHV_GL_REFCBAS(R0),W^PHV$GL_REFCBAS
113     MOVL    PHV_GL_PIXBAS(R0),W^PHV$GL_PIXBAS
114     MOVL    MMG_GL_PAGSWPVC(R0),W^MMG$GL_PAGSWPVC
115     MOVL    SCH_GL_PCBVEC(R0),W^SCH$GL_PCBVEC
116     MOVL    SCH_GL_SEQVEC(R0),W^SCH$GL_SEQVEC
117
118  20$:      MOVL    #SS$ _NORMAL,R0
119     RSB
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

MMDAT 17-MAY-1988 17:04:31 VAX MACRO V5.0-8 Page 5  
X-9 MMDAT\_INIT 28-AUG-1986 10:34:47 \_\$254\$DUA55:[SYS.SRC]MMDAT.MAR:1 (4)

```
121      DECLARE_PSECT   EXEC$NONPAGED_DATA ;
122 ;
123 ; Define vectors for cells commonly used in an indirect data reference.
124 ;
125 PFN$AB_STATE::
126      .LONG      0
127 PFN$AB_TYPE::
128      .LONG      0
129 PFN$AL_BAK::
130      .LONG      0
131 PFN$AL_PTE::
132      .LONG      0
133 PFN$AW_REFCNT::
134      .LONG      0
135 PFN$AW_SWPVEN::
136      .LONG      0
137 PFN$AX_BLINK::
138      .LONG      0
139 PFN$AX_FLINK::
140      .LONG      0
141 PFN$AX_SHRCNT::
142      .LONG      0
143 PFN$AX_WSLX::
144      .LONG      0
145 MMG$GL_GPTBASE::
146      .LONG      0
147 MMG$GL_SPTBASE::
148      .LONG      0
149 SWP$GL_BALSPT::
150      .LONG      0
151 SWP$GL_BALBASE::
152      .LONG      0
153 PHV$GL_REFCBAS::
154      .LONG      0
155 PHV$GL_PIXBAS::
156      .LONG      0
157 MMG$GL_PAGSWPVC::
158      .LONG      0
159 SCH$GL_PCBVEC::
160      .LONG      0
161 SCH$GL_SEQVEC::
162      .LONG      0
163
164      .END
```

## 9 DOINIT.LIS

SYSS\$DOINIT 10-MAY-1989 17:04:14 VAX MACRO V5.0-8 Page 0

### Table of contents

|      |     |  |
|------|-----|--|
| (3)  | 136 | INITIALIZATION VECTOR TABLES                                     |
| (4)  | 153 | INI\$DOINIT - Call all initialization routines                   |
| (6)  | 290 | IMG\$PRVSHRIMG Fixup Routine for Privileged Shareable Images     |
| (7)  | 331 | FIXUP_ADDRESS Fixup .ADDRESS entries throughout the image        |
| (8)  | 434 | PFN FIXUP ARRAYS   |
| (9)  | 458 | INI\$PFN_FIXUP - Alter PFN references if large PFN configuration |
| (10) | 512 | Build Table Vector   |
| (11) | 530 | INI\$SYSTEM_SERVICE - Load system service vectors                |

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSSDOINIT 10-MAY-1989 17:04:14 VAX MACRO V5.0-8 Page 1  
X-21 3-JUN-1987 12:33:40 [SYS.SRC]DOINIT.MAR;1 (1)

```
1      .TITLE  SYSSDOINIT
2      .IDENT  'X-21'
3 ;
4 ;*****
5 ;*
6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
8 ;*  ALL RIGHTS RESERVED.
9 ;*
10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
11 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
14 ;*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
15 ;*  TRANSFERRED.
16 ;*
17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
18 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
19 ;*  CORPORATION.
20 ;*
21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
23 ;*
24 ;*
25 ;*****
26 ;
27 ;  AUTHOR: Trudy C. Matthews                CREATION DATE: 1-Jul-1985
28 ;
29 ;  FUNCTIONAL DESCRIPTION:
30 ;
31 ;  This module is linked into each VMS executive sharable image. It provides
32 ;  the mechanism to execute initialization routines on a module-by-module
33 ;  basis, and allows each module to independently declare its own initialization
34 ;  routine.
35 ;
36 ;  Each module, if it has an initialization routine, will use the
37 ;  INITIALIZATION_ROUTINE macro to declare its entry point. This macro will
38 ;  enter a self-relative offset to the initialization routine into the
39 ;  initialization routine vector table. At initialization time, DOINIT will
40 ;  JSB through each vector in the table.
41 ;
42 ;
43 ;  ENVIRONMENT:
44 ;
45 ;  MODIFIED BY:
46 ;
47 ;      X-21      WMC0005      Wayne Cardoza      03-Jun-1987
48 ;              Check MAPEN before non-paged fix-ups.
49 ;
50 ;      X-20      SF04006      Stephen Fiorelli      09-Mar-1987
51 ;              Test initialization flags at the address in r5, not
52 ;              r5.
53 ;
54 ;      X-19      WMC0004      Wayne Cardoza      15-Oct-1986
55 ;              INIT psects must all be EXE.
56 ;
57 ;      X-14      SF04005      Stephen Fiorelli      10-Sep-1986
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYS\$DOINIT 10-MAY-1989 17:04:14 VAX MACRO V5.0-8 Page 2  
X-21 3-JUN-1987 12:33:40 [SYS.SRC]DOINIT.MAR;1 (1)

58 ;                   Change the handling of errors, no longer issue a  
59 ;                   halt, bugcheck instead.  
60 ;  
61 ;           X-13    HH215           Hai Huang                   02-Sep-1986  
62 ;                   This module is now inserted into STARLET.OLB, change  
63 ;                   module name to SYS\$DOINIT.  
64 ;  
65 ;           X-12    WMC0003       Wayne Cardoza                25-Jul-1986  
66 ;                   Get rid of re\_executeable switches.  
67 ;  
68 ;           X-11    WMC0002       Wayne Cardoza                21-May-1986  
69 ;                   Rewrite most of it.  
70 ;  
71 ;           X-10    WMC0001       Wayne Cardoza                11-Mar-1986  
72 ;                   PFNTBL psects mut be byte aligned.  
73 ;  
74 ;           X-8,9   SF04004       Stephen Fiorelli             17-Feb-1986  
75 ;                   Don't need to convert to byte count in the size of  
76 ;                   image sections when doing .address fixups. Byte count  
77 ;                   is already stored. Also moved computation of ending  
78 ;                   addresses of image sections out of a loop.  
79 ;  
80 ;           X-6,7   SF04003       Stephen Fiorelli             03-Feb-1986  
81 ;                   Change references to system service connection routines  
82 ;                   use exe\$ prefix instead of exec\$.  
83 ;  
84 ;           V04-005 SF04002       Stephen Fiorelli             03-Dec-1985  
85 ;                   Added error messages for system service initialization  
86 ;                   routine.  
87 ;  
88 ;           V04-004 SF04001       Stephen Fiorelli             16-Oct-1985  
89 ;                   Added initialization routine that will connect  
90 ;                   each system service within an EXEC shareable image  
91 ;                   to their vectors in the base image. Also added  
92 ;                   PSECTS to define the beginning and an of the build  
93 ;                   table (used in system service connecting).  
94 ;  
95 ;           V04-003 TCM0003       Trudy C. Matthews            14-Oct-1985  
96 ;                   More bug fixes in .ADDRESS fixup code.  
97 ;  
98 ;           V04-002 TCM0002       Trudy C. Matthews            1-Oct-1985  
99 ;                   Add conditional dispatching of iniialization routines based  
100 ;                   on \$INIRTN flags. Also, enhance LOADERS\$FIXUP\_DOT\_ADDRESS  
101 ;                   so that it will fixup .ADDRESS references in either non-paged  
102 ;                   only, paged only, or both paged and non-paged image sections,  
103 ;                   depending on what environment it is called from. Also,  
104 ;                   return status of success always from INI\$DOINIT.  
105 ;  
106 ;           V04-001 TCM0001       Trudy C. Matthews            29-Sep-1985  
107 ;                   Add .ADDRESS fixup routines.  
108 ;



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYS\$DOINIT 10-MAY-1989 17:04:14 VAX MACRO V5.0-8 Page 3  
X-21 3-JUN-1987 12:33:40 [SYS.SRC]DOINIT.MAR;1 (2)

```
110 ;
111 ; MACRO LIBRARY CALLS
112 ;
113
114     $BOOSTATEDEF           ; Define bootstrap stage flags
115     $IAFDEF                ; Offsets into image activator fixup
116                             ; area within image file
117     $INIRTNDEF             ; Define initialization routine flags
118     $LDRIMGDEF             ; Define SYSLDR image data structure
119     $PRDEF
120     $SHLDEF                ; Offsets into shareable image list element
121     $SSDESCRDEF           ; System service descriptor block
122
123
124 ;
125 ; LOCAL MACROS
126 ;
127
128 ;
129 ; LOCAL SYMBOLS
130 ;
131
132 ;
133 ; LOCAL DATA
134
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$DOINIT 10-MAY-1989 17:04:14 VAX MACRO V5.0-8 Page 4  
X-21 INITIALIZATION VECTOR TABLES 3-JUN-1987 12:33:40 [SYS.SRC]DOINIT.MAR;1 (3)

```
136      .SBTTL  INITIALIZATION VECTOR TABLES
137
138      .PSECT  EXEC$INIT_000, LONG, RD, WRT, EXE, PIC
139  INI$A_VECTOR_TABLE::          ; Beginning of vector table.
140
141      DECLARE_PSECT  EXEC$INIT_001
142 ;
143 ; This psect is declared here for informational purposes only.  Individual
144 ; modules contribute the address or a self-relative offset to an initialization
145 ; routine, plus a flags longword, to this table.
146 ;
147
148      .PSECT  EXEC$INIT_002, LONG, RD, WRT, EXE, PIC
149      .LONG  0          ; This is the signal for the end of the
150                      ; table.
151
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$DOINIT 10-MAY-1989 17:04:14 VAX MACRO V5.0-8 Page 5  
X-21 INI\$DOINIT - Call all initialization rou 3-JUN-1987 12:33:40 [SYS.SRC]DOINIT.MAR;1 (4)

```

153      .SBTTL  INI$DOINIT - Call all initialization routines
154      DECLARE_PSECT  EXEC$INIT_CODE
155 ;
156 ; Call each initialization routine in the vector table.
157 ;
158 ; Inputs:
159 ;     4(AP) - The address of this image data block (as created by SYSLDR)
160 ;
161 ; Outputs:
162 ;     None
163 ;
164 ; Implicit outputs:
165 ;     All routines in the vector table are called, based conditionally
166 ;     on the flags declared by that routine.  On entry to each routine,
167 ;     the following inputs are available:
168 ;
169 ;           R4 - address of image data block
170 ;           R5 - address of initialization routine flags longword
171 ;           R0,R1,R2,R3 - available as scratch registers
172 ;
173      .ENTRY  INI$DOINIT, ^M<R2,R3,R4,R5,R6,R7> ; Common initialization routine.
174      MOVAB  INI$A_VECTOR_TABLE,R6      ; Get address of vector table.
175      MOVL   4(AP),R4                    ; Get address of image data block.
176      BICL   #LDRIMG$M_DELAY_INIT,-     ; Assume no delayed init
177      LDRIMG$L_FLAGS(R4)
178 10$:
179      MOVL   (R6),R0                      ; Get address of init routine
180      BEQL   50$                          ; Zero - all done.
181      MOVAL  4(R6),R5                      ; Get initialization routine flags.
182      BBS    #INIRTN$V_NO_RECALL,-       ; Branch if routine not to be recalled
183      (R5),40$
184 20$:
185      BBS    #INIRTN$V_SYSRTN,(R5),-     ; Branch if routine is external to
186      30$                                  ; this image.
187      ADDL   R6,R0                          ; Convert self-relative offset to
188      ; actual address of routine.
189
190 30$:   JSB    (R0)                          ; Call initialization routine.
191      BISL   #INIRTN$M_CALLED,(R5)        ; Record that it was called
192      BLBC   R0,60$                          ; Error
193      BBS    #INIRTN$V_NO_RECALL,-       ; Branch if routine not to be recalled
194      (R5),40$
195      BISL   #LDRIMG$M_DELAY_INIT,-     ; Request delayed init
196      LDRIMG$L_FLAGS(R4)
197
198 40$:   ADDL   #8,R6                          ; Step to next vector.
199      BRB    10$                              ; See if there are any more.
200
201 50$:   MOVL   #SS$ _NORMAL,R0              ; Signal success always.
202 60$:   RET

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYSSDOINIT 10-MAY-1989 17:04:14 VAX MACRO V5.0-8 Page 6  
X-21 INI\$DOINIT - Call all initialization rou 3-JUN-1987 12:33:40 [SYS.SRC]DOINIT.MAR;1 (5)

```

204 ;+
205 ; LOADER$FIXUP_DOT_ADDRESS
206 ;
207 ; Functional Description:
208 ;
209 ;     This routine fixes up .ADDRESS references by adding the base address
210 ;     of the image to all .ADDRESS cells.
211 ;
212 ; Input Parameters:
213 ;
214 ;     R4 - Address of loadable image block
215 ;     R5 - address of initialization routine flags
216 ;     R0,R1,R2,R3 - available as scratch registers
217 ;
218 ; Output Parameters:
219 ;
220 ;     none
221 ;
222 ; Implicit Output:
223 ;
224 ;     All cells within the image originally generated with either .ADDRESS
225 ;     directives or .ASCID directives are modified (FIXED UP) by adding
226 ;     the base address of the image to each cell.
227 ;
228 ; Final Status:
229 ;
230 ;     Routines called by this routine may fail. These failure codes are
231 ;     passed back to the caller.
232 ;-
233
234     INITIALIZATION_ROUTINE -
235         LOADER$FIXUP_DOT_ADDRESS
236
237 LOADER$FIXUP_DOT_ADDRESS:
238     MOVL     $SS$ NORMAL,R0           ; Assume nothing to do
239     TSTL     LDRIMG$L_FIXUP_LEN(R4)  ; Any work to do?
240     BEQL     10$                      ; Simply return if nothing to do
241     BSBW     IMG$PRVSHRIMG           ; Is it legal set of fix-ups
242     BLBS     R0,20$                  ; Yes
243 10$:      RSB
244 ;
245 20$:      MOVL     LDRIMG$L_BASE(R4),R1 ; Get base address
246           MOVL     LDRIMG$L_FIXUP_BASE(R4),R2 ; Base of fix-up data
247           ADDL     IAF$L_DOTADROFF(R2),R2 ; .ADDRESS area
248           MOVL     (R2)+,R3           ; Count of fix-ups
249           BNEQ     30$
250           BISL     $LDRIMG$M_FIX_UPS_DONE,-; Done with all fix-ups
251           LDRIMG$L_FLAGS(R4)
252           RSB
253 30$:      ADDL     #4,R2              ; Skip image number
254
255 ;
256 ; Now determine what environment we're operating in, to see which (if any)
257 ; types of fixups we should do.
258 ;
259           MOVL     G^EXE$GL_STATE,R0 ; Get flags that tell us what stage
260           ; of bootstrap we're currently in.

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYS\$DOINIT 10-MAY-1989 17:04:14 VAX MACRO V5.0-8 Page 7  
X-21 INI\$DOINIT - Call all initialization rou 3-JUN-1987 12:33:40 [SYS.SRC]DOINIT.MAR;1 (5)

```
261 ;
262 ; If we're at the SWAPPER stage (which implies paging is now possible), which
263 ; fixups we do depends on whether this initialization routine has been called
264 ; before: if it has been called, the non-paged fixups are already done, so
265 ; just do the paged; if it hasn't, we must do them all.
266 ;
267     BBC     #BOOSTATE$V_SWAPPER,-      ; Branch if we haven't reached the
268           RO,50$                        ; SWAPPER stage yet.
269     BBS     #LDRIMG$V_NONPAGED_FIXUP,-
270           LDRIMG$L_FLAGS(R4),40$      ; We have already done the non-paged
271     BRW     FIXUP_ADDRESS_ALL
272 40$:     BRW     FIXUP_ADDRESS_PAGED
273 ;
274 ; If we're at the INIT stage, we can only do non-paged fixups. The only check
275 ; is to see if this routine was already called, and so the non-paged fixups
276 ; have already been done.
277 ;
278 50$:
279     MFPR    #PR$MAPEN,R0
280     BLBC    RO,55$                      ; Wait until memory management enabled
281     BBS     #LDRIMG$V_NONPAGED_FIXUP,-
282           LDRIMG$L_FLAGS(R4),55$      ; We have already done the non-paged
283     BRW     FIXUP_ADDRESS_NONPAGED
284
285 55$:     BICL    #INIRTN$M_NO_RECALL,-  ; We need to be called again
286           (R5)
287 60$:     MOVL    #SS$NORMAL,R0
288     RSB
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYSSDOINIT 10-MAY-1989 17:04:14 VAX MACRO V5.0-8 Page 8  
X-21 IMG\$PRVSHRIMG Fixup Routine for Privileg 3-JUN-1987 12:33:40 [SYS.SRC]DOINIT.MAR;1 (6)

```
290      .SBTTL  IMG$PRVSHRIMG  Fixup Routine for Privileged Shareable Images
291 ;+
292 ; Functional Description:
293 ;
294 ;      This routine checks that a privileged shareable image has no
295 ;      outbound calls.
296 ;
297 ;      This routine began as a copy of the routine found in SYSIMGFIX.MAR,
298 ;      which is part of the image activator.
299 ;
300 ; Calling Sequence:
301 ;
302 ;      JSB      IMG$PRVSHRIMG
303 ;
304 ; Input Parameters:
305 ;
306 ;      R4      Address of LDRIMG image data block
307 ;
308 ;
309 ; Side Effects:
310 ;
311 ;      R0 and R1 are destroyed
312 ;
313 ; Completion Codes:
314 ;
315 ;      SSS_NORMAL
316 ;
317 ;      SSS_NOSHRIMG  Shareable image has outbound calls
318 ;-
319
320
321 IMG$PRVSHRIMG:
322      MOVL     #SS$ NORMAL,R0          ; Assume success
323      MOVL     LDRIMG$L_FIXUP_BASE(R4),R1 ; Get base of data structures
324      CML     IAF$L_SHRIMGCNT(R1),#1   ; precisely one image?
325      BNEQ    10$,          ; No - error
326      TSTL    IAF$L_G_FIXOFF(R1)      ; Any G^
327      BEQL    20$,          ; No - no problem
328 10$:      MOVZWL #SS$ NOSHRIMG,R0     ; No outbound calls allowed
329 20$:      RSB
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$DOINIT 10-MAY-1989 17:04:14 VAX MACRO V5.0-8 Page 9  
X-21 FIXUP\_ADDRESS Fixup .ADDRESS entries thr 3-JUN-1987 12:33:40 [SYS.SRC]DOINIT.MAR;1 (7)

```

331      .SBTTL  FIXUP_ADDRESS  Fixup .ADDRESS entries throughout the image
332 ;+
333 ; Functional Description:
334 ;
335 ;     This routine performs the .ADDRESS fixup for a specific exit vector.
336 ;     Specifically, the base address of the appropriate shareable image
337 ;     is added to each .ADDRESS entry in this shareable image.
338 ;
339 ;     This routine has three entry points:
340 ;     FIXUP_ADDRESS_ALL      - fix up all .ADDRESS references in the image
341 ;     FIXUP_ADDRESS_PAGED    - only fix up those .ADDRESS references in the
342 ;                             pagable portion of the image
343 ;     FIXUP_ADDRESS_NONPAGED - only fix up those .ADDRESS references in the
344 ;                             non-pagable portion of the image
345 ;
346 ; Calling Sequence:
347 ;
348 ;     BSBW  FIXUP_ADDRESS_ALL
349 ;     BSBW  FIXUP_ADDRESS_PAGED
350 ;     BSBW  FIXUP_ADDRESS_NONPAGED
351 ;
352
353 ; Input Parameters:
354 ;
355 ;     R1 = Base of image
356 ;     R2 = Address of .ADDRESS area
357 ;     R3 = Count of fix-ups
358 ;     R4 = Address of LDRIMG image data block
359 ;     R5 = Address of initialization flags
360 ;
361 ; Implicit Input:
362 ;
363 ;     Contents of .ADDRESS fixup area
364 ;
365 ; Implicit Output:
366 ;
367 ;     .ADDRESS directives within this shareable image have the base addresses
368 ;     of the appropriate shareable images added to them.
369 ;-
370
371 FIXUP_ADDRESS_ALL:
372     BISL  #LDRIMG$M_FIX_UPS_DONE- ; Done with all fix-ups
373         !LDRIMG$M_NONPAGED_FIXUP,-
374         LDRIMG$L_FLAGS(R4)
375
376 10$:   ADDL3  R1,(R2)+,R0          ; Get address of .ADDRESS directive
377       ADDL2  R1,(R0)             ; Bias by base address of shareable image
378       SOBGT  R3,10$             ; Do next entry
379
380 20$:   MOVZWL #SS$ _NORMAL,R0    ; Indicate success
381       RSB                    ; Return
382
383
384 FIXUP_ADDRESS_NONPAGED:
385     PUSHR  #^M<R6,R7>
386     BISL  #LDRIMG$M_NONPAGED_FIXUP,- ; Record what we did
387         LDRIMG$L_FLAGS(R4)

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$DOINIT 10-MAY-1989 17:04:14 VAX MACRO V5.0-8 Page 10

X-21 FIXUP\_ADDRESS Fixup .ADDRESS entries thr 3-JUN-1987 12:33:40 [SYS.SRC]DOINIT.MAR;1 (7)

```

388      ADDL3  LDRIMG$L_PAG_W_BASE(R4),-    ; Compute the ending address
389      LDRIMG$L_PAG_W_LEN(R4),R7 ; of the pagable portion.
390      CLRL   R6                          ; Count of paged fix-ups
391 10$:  ADDL3  R1,(R2)+,R0                 ; Get address of .ADDRESS directive
392      CMPL  RO,LDRIMG$L_PAG_W_BASE(R4)    ; Compare against start of paged
393      BLSSU 15$                          ; Not paged - go fix it up.
394      CMPL  RO,R7                        ; Compare against end of paged region.
395      BGEQU 15$                          ; Not paged - go fix it up.
396      INCL  R6                          ; Count a paged one
397      BRB   20$
398 15$:  ADDL2  R1,(R0)                    ; Bias by base address of shareable image
399 20$:  SOBGTR R3,10$                     ; Do next entry
400
401      TSTL  R6                          ; Any paged ones
402      BNEQ  30$                          ; Yes
403      BISL  #LDRIMG$M_FIX_UPS_DONE,-; Done with all fix-ups
404      LDRIMG$L_FLAGS(R4)
405      BRB   40$
406 30$:  BICL  #INIRTN$M_NO_RECALL,-    ; We need to be called again
407      (R5)
408 40$:  MOVZWL #SS$ NORMAL,R0            ; Indicate success
409      POPR  #^M<R6,R7>
410      RSB                               ; Return
411
412
413 FIXUP_ADDRESS_PAGED:
414      BISL  #LDRIMG$M_FIX_UPS_DONE,-; Done with all fix-ups
415      LDRIMG$L_FLAGS(R4)
416
417      PUSHR #^M<R7>
418      ADDL3  LDRIMG$L_PAG_W_BASE(R4),-    ; Compute the ending address
419      LDRIMG$L_PAG_W_LEN(R4),-    ; of the pagable portion.
420      R7
421 10$:  ADDL3  R1,(R2)+,R0                 ; Get address of .ADDRESS directive
422      CMPL  RO,LDRIMG$L_PAG_W_BASE(R4)    ; Compare against start of paged
423      BLSSU 20$                          ; Less than start - don't fix it up
424      CMPL  RO,R7                        ; Compare against end of paged region.
425      BGEQU 20$                          ; Not in the paged part - don't fix it.
426
427 15$:  ADDL2  R1,(R0)                    ; Bias by base address of shareable image
428 20$:  SOBGTR R3,10$                     ; Do next entry
429
430      MOVZWL #SS$ NORMAL,R0            ; Indicate success
431      POPR  #^M<R7>
432      RSB                               ; Return

```



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYS\$DOINIT 10-MAY-1989 17:04:14 VAX MACRO V5.0-8 Page 11  
X-21 PFN FIXUP ARRAYS 3-JUN-1987 12:33:40 [SYS.SRC]DOINIT.MAR;1 (8)

```
434      .SBTTL  PFN FIXUP ARRAYS
435 ;+
436 ;
437 ; Define global label for opcode/address table used by code that 'fixes up'
438 ; all pfn references if more than 32 Mbytes of memory is present on the system.
439 ;-
440      .PSECT  EXEC$INIT_PFN_TBL_000, LONG, RD, WRT, EXE, PIC
441 MMG$AL_FIXUP_TBL::
442
443      DECLARE_PSECT  EXEC$INIT_PFN_TBL_001, BYTE
444 ;
445 ; This psect is included here for information only.  The table entries go
446 ; in this psect.
447 ;
448 ; Each six byte entry in this table consists of a location whose contents
449 ; are to be altered, a byte containing the current contents of that location
450 ; (to be used as a sanity check), and a byte containing the new opcode.  The
451 ; table is terminated with a longword of zero.
452 ;
453
454      .PSECT  EXEC$INIT_PFN_TBL_002, BYTE, RD, WRT, EXE, PIC
455      .LONG  0 ; End of PFN fixup table
456
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$DOINIT 10-MAY-1989 17:04:14 VAX MACRO V5.0-8 Page 12

X-21 INI\$PFN\_FIXUP - Alter PFN references if 3-JUN-1987 12:33:40 [SYS.SRC]DOINIT.MAR;1 (9)

```

458      .SBTTL  INI$PFN_FIXUP - Alter PFN references if large PFN configuration
459      DECLARE_PSECT  EXEC$INIT_CODE
460 ;+
461 ; INI$PFN_FIXUP
462 ; This initialization routine is common to many exec images.
463 ;
464 ; If there is less than 32 Mbytes of memory described in the PFN data base
465 ; (MMG$GW_BIGPFN contains zero), this next block of code does nothing.
466 ; Otherwise, an address table is scanned.
467 ;
468 ;     1. Each address must be in the nonpaged system image.
469 ;
470 ;     2. The current contents are verified as a consistency check.
471 ;
472 ;     3. A new (longword context) opcode is stored at that location.
473 ;
474 ; Failure of either test prevents the system from being bootstrapped with
475 ; more than 32 Mbytes of physical memory. (That is, the PHYSICALPAGES
476 ; parameter must be used to allow the system to come up using less than
477 ; its total amount of physical memory.)
478 ;-
479
480      INITIALIZATION_ROUTINE  INI$PFN_FIXUP
481 INI$PFN_FIXUP::
482      PFN_DISP_IF_BIGPFN_THEN          END_BIGPFN_CODE=100$

;This code executes if the PFN link arrays are longword arrays.

483
484      MOVAL  G^MMG$AL_FIXUPTBL,R1      ; Address of opcode/address table
485 10$:    TSTL   (R1)                   ; Self-relative offset of next fixup
486      BEQL   50$                       ; Zero indicates end of list
487      MOVAB  @ (R1) [R1],R0            ; Calculate address from self-relative
488                                         ; offset
489      CMPB   4 (R1), (R0)              ; Perform sanity check
490      BNEQU  30$                       ; Quit with error if different
491      MOVB   5 (R1), (R0)              ; Finally, alter the opcode
492      ADDL   #6,R1                     ; R1 now points to old opcode byte
493      BRB    10$                       ; and go back for the next one
494
495 30$:    BUG_CHECK PFNFIXUP,FATAL      ; Opcode mismatch, bugcheck
496
497 ;
498 ; This is the successful exit path after all opcodes have been altered. It
499 ; is necessary to execute an REI instruction in order that any instruction
500 ; lookahead be invalidated and the new opcodes used.
501 ;
502 50$:    MOVPSL -(SP)                   ; Store PSL for REI
503      PUSHAB B^100$                   ; Push PC also
504      REI                                ; Drop through to next instruction
505
506      PFN_DISP_ENDIF                  COMMON_CODE=100$

;End of code that depends on size of PFN link arrays
507      MOVL   #SS$_NORMAL,R0
508      RSB
509
510

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SYS\$DOINIT 10-MAY-1989 17:04:14 VAX MACRO V5.0-8 Page 13  
X-21 Build Table Vector 3-JUN-1987 12:33:40 [SYS.SRC]DOINIT.MAR;1 (10)

```
512      .SBTTL  Build Table Vector
513
514      .PSECT  EXEC$INIT_SSTBL_000, LONG, RD, WRT, EXE, PIC
515  INI$A_BUILD_TABLE::          ; Beginning of build table
516
517      DECLARE_PSECT  EXEC$INIT_SSTBL_001
518 ;
519 ; This psect is declared here for informational purposes only.
520 ; The SYSTEM_SERVICE macro is responsible for contributing
521 ; build table entries.
522 ;
523
524      .PSECT  EXEC$INIT_SSTBL_002, LONG, RD, WRT, EXE, PIC
525      .LONG  0          ; This is the signal for the end of the
526                  ; table.
527
528
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SYS\$DOINIT 10-MAY-1989 17:04:14 VAX MACRO V5.0-8 Page 14

X-21 INI\$SYSTEM\_SERVICE - Load system service 3-JUN-1987 12:33:40 [SYS.SRC]DOINIT.MAR;1 (11)

```
530      .SBTTL  INI$SYSTEM_SERVICE - Load system service vectors
531
532      DECLARE_PSECT  EXEC$INIT_CODE
533
534 ;--
535 ;      This initialization routine is common to many EXEC
536 ;      shareable images.
537 ;
538 ;      The build table for an EXEC shareable image is
539 ;      traversed to connect system services with their
540 ;      vectors in the base image. Each build table
541 ;      entry is passed to the system service connector.
542 ;
543 ;      The build table is built at assemble time with
544 ;      the SYSTEM_SERVICE macro.
545 ;--
546
547      INITIALIZATION ROUTINE -
548      INI$SYSTEM_SERVICE
549
550 INI$SYSTEM_SERVICE::
551      MOVL    G^EXE$GL_STATE,R0          ; Get flags that tell us what stage
552                                          ; of bootstrap we're currently in.
553      BBC     #BOOSTATE$V_SWAPPER,-     ; Branch if we haven't reached the
554      R0,40$                               ; SWAPPER stage yet.
555      MOVAL  INI$A_BUILD_TABLE,R2      ; Get the beginning of the build table
556      MOVL   #SSDESCR K_LENGTH,R3     ; Size of a build table entry
557      CALLS  #0,G^EXE$SET_PAGES_WRITABLE ; Allow modifications to system serv
558                                          ; vectors in base image
559 10$:
560      TSTL   (R2)                        ; Test for end of build table
561      BEQL  20$                          ; If end, then done
562      PUSHL R2                            ; Pass address of a build table entry
563      CALLS #1,G^EXE$CONNECT_SERVICES ; Connect the system service
564      MOVAB (R2)[R3],R2                  ; Move to next build table entry
565      BRB   10$                          ;
566 20$:
567      CALLS #0,G^EXE$SET_PAGES_READ_ONLY; All done.  Protect pages again.
568 30$:
569      MOVL  #SS$_NORMAL,R0              ; Always success
570      RSB
571
572 40$:  BICL  #INIRTN$M_NO_RECALL,-      ; We need to be called again
573      (R5)
574      BRB  30$
575
576      .END  INI$DOINIT                  ; This is the initialization routine.
```

## 10 PTALLOC.LIS

PTALLOC - System page table allocation routines 10-MAY-1989 15:58:36 VAX MACRO V5.0-  
8 Page 0

### Table of contents

|     |     |   |
|-----|-----|---|
| (2) | 95  | DECLARATIONS  |
| (3) | 107 | LDR\$ALLOC_PT - Allocate page table entries         |
| (4) | 201 | SYNCHRONIZE - Synchronize allocation of system PTEs |
| (5) | 275 | LDR\$DEALLOC_PT - Deallocate page table entries     |

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PTALOC - System page table allocation routines 10-MAY-1989 15:58:36 VAX MACRO V5.0-  
8 Page 1  
X-11 25-JAN-1988 12:22:29 [SYS.SRC]PTALOC.MAR;1 (1)

```

1      .TITLE  PTALOC - System page table allocation routines
2      .IDENT  'X-11'
3 ;
4 ;*****
5 ;*
6 ;*  COPYRIGHT (c) 1985 BY
7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
8 ;*  ALL RIGHTS RESERVED.
9 ;*
10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
11 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
14 ;*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
15 ;*  TRANSFERRED.
16 ;*
17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
18 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
19 ;*  CORPORATION.
20 ;*
21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
23 ;*
24 ;*
25 ;*****
26 ;
27
28 ;++
29 ; Facility:
30 ;
31 ;     System Code Loader
32 ;
33 ; Abstract:
34 ;
35 ;     Allocation of system page table entries.
36 ;
37 ; Environment:
38 ;
39 ;     Kernel Mode.
40 ;
41 ; Author:
42 ;
43 ;     Wayne Cardoza
44 ;
45 ; Creation Date:
46 ;
47 ;     10-Apr-1985
48 ;
49 ; Modified by:
50 ;
51 ;     X-11      WMC0008      Wayne Cardoza      25-Jan-1988
52 ;             Fix insertions to empty list and at end of list.
53 ;
54 ;     X-10      WMC0007      Wayne Cardoza      14-Nov-1986
55 ;             Base synchronization technique on swapper run yet.
56 ;
57 ;     X-9       WMC0006      Wayne Cardoza      15-Oct-1986

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

PTALLOCC - System page table allocation routines 10-MAY-1989 15:58:36 VAX MACRO V5.0-  
8 Page 2

X-11 25-JAN-1988 12:22:29 [SYS.SRC]PTALLOCC.MAR;1 (1)

|        |         |         |   |              |
|--------|---------|---------|---|--------------|
| 58 ;   |         |         | Yet another problem with combining.                       |              |
| 59 ;   |         |         |   |              |
| 60 ;   | X-8     | SF04002 | Stephen Fiorelli  | 24-Sep-1986  |
| 61 ;   |         |         | Fix deallocation when the block of ptes to be             |              |
| 62 ;   |         |         | deallocated abuts another free block above it.            |              |
| 63 ;   |         |         |   |              |
| 64 ;   | X-7     | WMC0005 | Wayne Cardoza   | 01-Aug-1986  |
| 65 ;   |         |         | Handle empty list properly.                               |              |
| 66 ;   |         |         |   |              |
| 67 ;   | X-6     | WMC0004 | Wayne Cardoza   | 08-Jul-1986  |
| 68 ;   |         |         | Link free list in reverse order.                          |              |
| 69 ;   |         |         |   |              |
| 70 ;   | X-5H3   | RNG5003 | Rod Gamache   | 10-Jun-1986  |
| 71 ;   |         |         | Conditionally release MMG spinlocks.                      |              |
| 72 ;   |         |         |   |              |
| 73 ;   | X-5H2   | MSH0248 | Michael S. Harvey   | 2-Jun-1986   |
| 74 ;   |         |         | Add spinlock synchronization for SMP operation. Also,     |              |
| 75 ;   |         |         | fix problems that would be caused if the length parameter |              |
| 76 ;   |         |         | is passed to these routines with a value of zero.         |              |
| 77 ;   |         |         |   |              |
| 78 ;   | X-5     | SF04001 | Stephen Fiorelli  | 12-May-1986  |
| 79 ;   |         |         | When searching through the pte links when deallocating    |              |
| 80 ;   |         |         | a pte, add ldr\$gl_sptbase to the next pte offset.        |              |
| 81 ;   |         |         |   |              |
| 82 ;   | X-4     | WMC0003 | Wayne Cardoza   | 07-Jan-1986  |
| 83 ;   |         |         | Really preserve R2.                                       |              |
| 84 ;   |         |         |   |              |
| 85 ;   | X-3     | WMC0002 | Wayne Cardoza   | 14-Nov-1985  |
| 86 ;   |         |         | Make routine vectored symbol.                             |              |
| 87 ;   |         |         |   |              |
| 88 ;   | X-2     | WMC0001 | Wayne Cardoza   | 14-Nov-1985  |
| 89 ;   |         |         | Add comment that R2 is preserved.                         |              |
| 90 ;   |         |         |   |              |
| 91 ;   | V04-001 | LJK4011 | Lawrence J. Kenah   | 18 July 1985 |
| 92 ;   |         |         | Perform minor cleanup.                                    |              |
| 93 ;-- |         |         |   |              |

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PTALLOC - System page table allocation routines 10-MAY-1989 15:58:36 VAX MACRO V5.0-  
8 Page 3

X-11 DECLARATIONS 25-JAN-1988 12:22:29 [SYS.SRC]PTALLOC.MAR;1 (2)

```
95      .SUBTITLE      DECLARATIONS
96
97      $BOOSTATEDEF
98      $IPLDEF
99      $LDRPTEDEF
100     $PRDEF
101     $SSDEF
102
103 ; Program section directives
104
105     DECLARE_PSECT   EXEC$NONPAGED_CODE
```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PTALLOC - System page table allocation routines 10-MAY-1989 15:58:36 VAX MACRO V5.0-

8 Page 4

X-11 LDR\$ALLOC\_PT - Allocate page table entr 25-JAN-1988 12:22:29 [SYS.SRC]PTALLOC.MAR;1 (3)

```

107      .SUBTITLE      LDR$ALLOC_PT - Allocate page table entries
108 ;++;
109 ; LDR$ALLOC_PT - Allocate page table entries
110 ;
111 ;      This subroutine allocates system page table entries.
112 ;
113 ; Calling Sequence:
114 ;
115 ;      JSB LDR$ALLOC_PT
116 ;
117 ; Input Parameters:
118 ;
119 ;      R2 -> Requested PTE count
120 ;
121 ; Implicit Input:
122 ;
123 ;      Linked list formats for free PTE entries:
124
125 ;          For 2 or more PTEs
126 ;              21 20                                0
127 ;          +-----+-----+-----+-----+
128 ;          |           | 0 | link (PTE offset) |
129 ;          +-----+-----+-----+-----+
130 ;          |           |           | total count |
131 ;          +-----+-----+-----+-----+
132 ;
133 ;          For a single free PTE
134 ;              21 20                                0
135 ;          +-----+-----+-----+-----+
136 ;          |           | 1 | link (PTE offset) |
137 ;          +-----+-----+-----+-----+
138 ;
139 ;      List header: LDR$GL_FREE_PT
140 ;
141 ; Output Parameters:
142 ;
143 ;      R0 -> Completion Status
144 ;      R1 -> Starting PTE address
145 ;      R2 -> Preserved
146 ;
147 ; Implicit Output:
148 ;
149 ;      The pool of available PTEs is altered to reflect the allocation.
150 ;--
151
152 UNIVERSAL_SYMBOL LDR$ALLOC_PT
153
154      BSBB      SYNCHRONIZE                ; Synchronize with all CPUs
155      PUSHR    #^M<R3>
156      MOVL     R2,R1                      ; If length zero, clear PTE address
157      BNEQ     5$
158      MOVL     #SS$_BADPARAM,R0           ; If NEQ, continue with allocation
159      BRB      95$
160
161 5$:      MOVAB  G^LDR$GL_FREE_PT,R3      ; No previous free ones
162      TSTL     (R3)                       ; Empty?
163      BEQL     100$

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PTALLOC - System page table allocation routines 10-MAY-1989 15:58:36 VAX MACRO V5.0-  
8 Page 5

X-11 LDR\$ALLOC\_PT - Allocate page table entr 25-JAN-1988 12:22:29 [SYS.SRC]PTALLOC.MAR;1 (3)

```

164      MOVL      G^LDR$GL_SPTBASE,R1          ; Get base of system page table
165      ADDL      (R3),R1                      ; Get to first free PTE
166 10$:  MOVL      #1,R0                       ; Assume only one
167      BBS       #LDRPTE$V_FLAG,(R1),20$     ; Branch if really is only one
168      MOVL      LDRPTE$L_COUNT(R1),R0       ; Real count
169 20$:  CMPL     R2,R0                       ; Enough?
170      BLEQL    40$                          ; Yes
171      MOVL      R1,R3                       ; Save previous pointer
172      EXTZV     #LDRPTE$V_LINK,#LDRPTE$S_LINK,- ; Get link to next free one
173      (R1),R1
174      BEQL     100$                          ; No luck
175      ADDL      G^LDR$GL_SPTBASE,R1         ; Next
176      BRB      10$                          ; Try again
177
178 40$:  BNEQ     60$                          ; Not an exact match
179      INSV     (R1),#LDRPTE$V_LINK,-        ; Link around this one
180      #LDRPTE$S_LINK,(R3)
181      BRB      90$
182
183 60$:  SUBL3    R2,LDRPTE$L_COUNT(R1),R0     ; Shrink the current free space
184      MOVL      R0,LDRPTE$L_COUNT(R1)       ; Save new size
185      CMPL     R0,#1                         ; Do we need to set flag
186      BNEQ     70$                          ; OK - more than one left
187      BBSS     #LDRPTE$V_FLAG,(R1),70$     ; Indicate only 1
188 70$:  MOVAL    (R1)[R0],R1                 ; Address of first free one
189 90$:  MOVL      R1,R3                       ; Address of first one
190      MOVL      R2,R0                       ; Count
191 91$:  CLRL     (R3)+                       ; Make sure they are all clear
192      SOBGTR   R0,91$
193
194      MOVL      #SS$_NORMAL,R0
195 95$:  POPR     #^M<R3>
196      RSB
197
198 100$: MOVZWL   #SS$_INSFPTS,R0
199      BRB      95$

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PTALOC - System page table allocation routines 10-MAY-1989 15:58:36 VAX MACRO V5.0-

8 Page 6

X-11 SYNCHRONIZE - Synchronize allocation of 25-JAN-1988 12:22:29 [SYS.SRC]PTALOC.MAR;1 (4)

```

201      .SUBTITLE      SYNCHRONIZE - Synchronize allocation of system PTEs
202 ;+
203 ; SYNCHRONIZE - Synchronize allocation/deallocation of system page table entries
204 ;
205 ;     This subroutine ensures that access to the system page table
206 ;     for the purposes of allocation and deallocation of the SPTEs
207 ;     is properly synchronized in all environments in which the caller
208 ;     may be operating.
209 ;
210 ;     This routine will operate as a coroutine most of the time. A caller
211 ;     of this routine can assume upon return that synchronization has been
212 ;     achieved, and can simply execute an RSB instruction when it's done.
213 ;     That RSB returns control to this coroutine which desynchronizes
214 ;     prior to returning control to the caller's caller.
215 ;
216 ; Calling Sequence:
217 ;
218 ;     BSBx/JSB SYNCHRONIZE
219 ;
220 ; Input Parameters:
221 ;
222 ;     None.
223 ;
224 ; Implicit Input:
225 ;
226 ;     Contents of the MAPEN IPR
227 ;
228 ; Output Parameters:
229 ;
230 ;     None
231 ;
232 ; Implicit Output:
233 ;
234 ;     If swapper initialization has not executed, this routine simply
235 ;     returns to the caller, assuming that synchronization is already
236 ;     achieved. This assumption is based on the following facts:
237 ;
238 ;         PRIMARY CPU only calls this during early stages of system
239 ;         initial booting. During this time, no other CPUs
240 ;         are running that need to be synchronized with.
241 ;
242 ;         non-PRIMARY CPUs never call this code during early stages of
243 ;         booting.
244 ;
245 ;     After swapper initialization, this routine synchronizes across
246 ;     all CPUs and then calls the caller as a coroutine. Desynchronization
247 ;     occurs when the caller executes an RSB back to this coroutine.
248 ;
249 ;-
250 SYNCHRONIZE:                ; Synchronize the caller with all CPUs
251     BBC      #BOOSTATE$V_SWAPPER,-
252             G^EXE$GGL_STATE,-      ; If not set, assume that this is the
253             BOOTING                 ; PRIMARY CPU going through the
254                                     ; system initialization sequence
255                                     ; and that there are no other CPUs
256                                     ; with which to synchronize
257     POPL     R0                    ; Pop return address into scratch register

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PTALOC - System page table allocation routines 10-MAY-1989 15:58:36 VAX MACRO V5.0-

8 Page 7

X-11 SYNCHRONIZE - Synchronize allocation of 25-JAN-1988 12:22:29 [SYS.SRC]PTALOC.MAR;1 (4)

```
258         LOCK   LOCKNAME=MMG,-           ; Synchronize with all CPUs
259         LOCKIPL=IPL$_SYNCH,-           ;
260         SAVIPL=-(SP)                   ;
261         JSB     (R0)                   ; Call caller as a coroutine
262
263 ;
264 ; When the caller finishes, it will issue an RSB instruction which returns
265 ; control to here. Desynchronize across all CPUs, restore IPL as appropriate,
266 ; and return to the caller's caller.
267 ;
268         UNLOCK  LOCKNAME=MMG,-           ; Desynchronize system structures
269         NEWIPL=(SP)+,-                   ;
270         CONDITION=RESTORE              ; Conditionally release spinlock
271
272 BOOTING:
273         RSB
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PTALLOC - System page table allocation routines 10-MAY-1989 15:58:36 VAX MACRO V5.0-

8 Page 8

X-11 LDR\$DEALLOC\_PT - Deallocate page table 25-JAN-1988 12:22:29 [SYS.SRC]PTALLOC.MAR;1 (5)

```

275      .SUBTITLE      LDR$DEALLOC_PT - Deallocate page table entries
276 ;+
277 ; LDR$DEALLOC_PT - Deallocate page table entries
278 ;
279 ;      This subroutine deallocates system page table entries.
280 ;
281 ;      Note that entries are linked in reverse address order. This ensures
282 ;      that freed entries are checked before the large initial block on
283 ;      allocation.
284 ;
285 ; Calling Sequence:
286 ;
287 ;      JSB LDR$DEALLOC_PT
288 ;
289 ; Input Parameters:
290 ;
291 ;      R1 -> Starting PTE address
292 ;      R2 -> PTE count
293 ;
294 ; Implicit Input:
295 ;
296 ;      Linked list for free PTE entries:
297 ;
298 ; Output Parameters:
299 ;
300 ;      R0 -> Completion Status
301 ;      R2 -> Not Preserved
302 ;
303 ; Implicit Output:
304 ;
305 ;      The pool of free PTEs is altered to reflect the addition of PTEs that
306 ;      were just deallocated.
307 ;-
308
309 UNIVERSAL_SYMBOL LDR$DEALLOC_PT
310
311      BSBB      SYNCHRONIZE      ; Synchronize across all CPUs
312      PUSHR    #^M<R3,R4,R5,R6>
313      TSTL     R2      ; Non-zero length specified?
314      BEQL     110$     ; If EQL yes, return an error
315      CLRL     R0
316 20$:      TSTL     (R1)[R0]     ; Is PTE zero?
317      BNEQ     100$     ; No - potential disaster
318      AOBLSS   R2,R0,20$     ; Go check the next one
319
320      MOVL     G^LDR$GL_SPTBASE,R4
321      ADDL     G^LDR$GL_FREE_PT,R4      ; Address of first free PTE
322      MOVAL    G^LDR$GL_FREE_PT,R3     ; No previous one yet
323
324      TSTL     (R3)      ; Empty list?
325      BEQL     220$     ; Yes
326
327 30$:      CMPL     R4,R1      ; Is this the place to insert it
328      BLEQU    200$     ; Yes
329      MOVL     R3,R6      ; Save 2nd predecessor
330      MOVL     R4,R3      ; New predecessor
331      EXTZV    #LDRPTE$V_LINK,#LDRPTE$S_LINK,- ; Get link to next free one

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

PTALLOC - System page table allocation routines 10-MAY-1989 15:58:36 VAX MACRO V5.0-  
8 Page 9

X-11 LDR\$DEALLOC\_PT - Deallocate page table 25-JAN-1988 12:22:29 [SYS.SRC]PTALLOC.MAR;1 (5)

```

332                (R3),R4
333      BEQL      220$                ; End of list
334      ADDL      G^LDR$GL_SPTBASE,R4 ; Address of free PTE
335      BRB       30$                ; Go try next one
336
337 100$:  MOVL     #LOADER$_PTE_NOT_EMPTY,R0 ; PTEs aren't all zero
338      BRB       310$
339
340 110$:  MOVL     #SS$_BADPARAM,R0        ; Bad deallocation length
341      BRB       310$
342
343 200$:  MOVL     #1,R0                  ; Assume preceeding is a single
344      BBS       #LDRPTE$V_FLAG,(R4),210$ ; Is it?
345      MOVL     LDRPTE$L_COUNT(R4),R0    ; Get the correct count
346 210$:  MOVAL    (R4)[R0],R5           ; First PTE past preceeding
347      CML     R5,R1                    ; Can they be combined
348      BNEQ     220$                     ; No
349      MOVL     R4,R1                    ; New starting address
350      ADDL     R0,R2                     ; New count
351      MOVL     R2,LDRPTE$L_COUNT(R4)    ; Make it one big one
352      BBCC     #LDRPTE$V_FLAG,(R4),240$ ; Make sure it is clear
353      BRB       240$                     ; Skip all the linking
354
355 220$:  EXTZV    #LDRPTE$V_LINK,#LDRPTE$$_LINK,- ; Get link to next one
356                (R3),(R1)
357      SUBL3    G^LDR$GL_SPTBASE,R1,R0    ; Get offset to the new one
358      INSV     R0,#LDRPTE$V_LINK,-      ; Link in the new one
359                #LDRPTE$$_LINK,(R3)
360      CML     R2,#1                       ; Is there only one
361      BEQL     230$                       ; Yes
362      MOVL     R2,LDRPTE$L_COUNT(R1)    ; Add the count
363      BRB       240$
364
365 230$:  BBSS     #LDRPTE$V_FLAG,(R1),240$ ; Indicate only one PTE
366
367 240$:  MOVAL    (R1)[R2],R0             ; One PTE past the new ones
368      CML     R0,R3                       ; Can we combine with the next
369      BNEQ     300$                       ; No
370      EXTZV    #LDRPTE$V_LINK,#LDRPTE$$_LINK,- ; Get link to next one
371                (R3),R0
372      INSV     R0,#LDRPTE$V_LINK,#LDRPTE$$_LINK,(R6)
373      MOVL     #1,R0                       ; Assume next is a single
374      BBS       #LDRPTE$V_FLAG,(R3),250$ ; Is it?
375      MOVL     LDRPTE$L_COUNT(R3),R0    ; Get real count
376 250$:  ADDL3    R0,R2,LDRPTE$L_COUNT(R1) ; Combined count
377      BBCC     #LDRPTE$V_FLAG,(R1),300$ ; Make sure it is clear
378 300$:  MOVL     #SS$_NORMAL,R0
379 310$:  POPR     #^M<R3,R4,R5,R6>
380      RSB                                ; Desynchronize and return
381
382      .END

```

# 11 SMPROUT.LIS

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 0  
Table of contents

|      |      |   |
|------|------|---|
| (1)  | 35   | HISTORY   |
| (2)  | 350  | DECLARATIONS  |
| (3)  | 383  | GET_CURPCB - Return current PCB address                           |
| (4)  | 426  | SMP\$SWITCH_CPU - Switch to another CPU base on device affinity   |
| (5)  | 526  | IOPOST_IRP - Place IRP on Per-processor IOPOST Q.                 |
| (6)  | 590  | SMP\$INVALID - invalidate a single TB entry.                      |
| (7)  | 673  | SMP\$TIMEOUT - SMP timeout processing routine                     |
| (8)  | 704  | INIT_SANITY - Initialize SMP sanity timer pointer in CPU database |
| (9)  | 801  | SMP\$SHUTDOWN_CPU - Final actions associated with stopping a CPU. |
| (10) | 1037 | SMP\$INITIATE_BENIGN - Initiate a benign state                    |
| (11) | 1076 | SMP\$SETUP_PFORK -- setup for fork to primary                     |
| (12) | 1165 | SMP\$FORK_TO_PRIMARY- migrate work packet to primary CPU          |
| (12) | 1350 | MESSAGE Display message on virtual console                        |
| (13) | 1410 | SMP\$WRITE_OPA0 - fork routine to broadcast message to console    |
| (14) | 1450 | SYNCH\$INIT_ONCE - Once only initialization for Spinlocks         |

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 1  
X-72 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1)

```

1      .TITLE  SMPROUT - SMP routines for VMS
2      .IDENT  'X-72'
3 ;
4 ;*****
5 ;*
6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
8 ;*  ALL RIGHTS RESERVED.
9 ;*
10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
11 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
14 ;*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
15 ;*  TRANSFERRED.
16 ;*
17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
18 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
19 ;*  CORPORATION.
20 ;*
21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
23 ;*
24 ;*
25 ;*****
26 ;
27 ;++
28 ; Facility:      Symmetric Multiprocessing
29 ;
30 ; Abstract:      The subroutines to implement the acquire and release busy
31 ;                wait spinlocks for symmetric multiprocessing.
32 ;
33 ; Environment:   Kernel Mode.
34 ;
35      .SBTTL  HISTORY
36 ;
37 ; Author:        Rod N. Gamache           Creation Date: 3-Mar-1986
38 ;
39 ; Modified by:
40 ;
41 ;      X-72                Brian Porter           19-APR-1989
42 ;                RESTORE EMB lock rather than RELEASE.
43 ;
44 ;      X-71                DDP0315           Derrell D. Piper           15-Sep-1988
45 ;                Initialize audit server mailbox device lock address.
46 ;
47 ;      X-70                WMC0070           Wayne Cardoza             25-Aug-1988
48 ;                Major changes for redesign of affinity and capabilities.
49 ;                Several routines removed.
50 ;
51 ;      X-69                EMB0355           Ellen M. Batbouta        17-Aug-1988
52 ;                In SMP$SWITCH_CPU, check for primary affinity which
53 ;                is specified by the value zero in the UCB$L_AFFINITY
54 ;                field.
55 ;
56 ;      X-68                WMC0068           Wayne Cardoza             02-Aug-1988
57 ;                Call routine to chnage process from CUR to COM.

```



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 2  
X-72 HISTORY 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1)

58 ;  
59 ; X-67 SJF Stu Farnham 30-Jun-1988  
60 ; Fix rescheduling logic in SETAFF/SETCAP. Correct operand  
61 ; order on an ASHL in SHUTDOWN CPU. Handle BOOT REJECTED state  
62 ; correctly in SMP\$SHUTDOWN\_CPU. Fix a broken BBCC in SMP\$SHUTDOWN\_CPU  
63 ;  
64 ; X-66 HH0320 Hai Huang 07-Jun-1988  
65 ; Unlock the SCHED spinlock with the RESTORE parameter, so  
66 ; SMP\$SETCAP/SMP\$SETAFF can be called with SCHED already held.  
67 ;  
68 ; X-65 RNG5065 Rod Gamache 31-Mar-1988  
69 ; Modify SMP\$IOPPOST\_IRP to use system-wide post queue.  
70 ;  
71 ; X-64 SJF Stu Farnham 9-Feb-1988  
72 ; 1. Changes some names:  
73 ; EXE\$GL WRITE CONS\_POOL becomes SMP\$GL PFORK\_POOL  
74 ; EXE\$WRITE CONS becomes SMP\$SETUP\_PFORK  
75 ; 2. Change initialization of PFORK\_POOL to use contents  
76 ; of SMP\$GB PFORK\_POOL\_SIZE to determine how many pages  
77 ; to allocate.  
78 ; 3. Move redundant code from several modules to this one  
79 ; at entrypoint SMP\$WRITE\_OPAO. Modify SMP\$SHUTDOWN\_CPU  
80 ; to use this entrypoint.  
81 ; 4. Teach SMP\$SHUTDOWN\_CPU to exit at IPL of entry.  
82 ;  
83 ; X-63 SJF Stu Farnham 27-Jan-1988  
84 ; Add SMP\$FORK\_TO\_PRIMARY (and assoc. code) to provide  
85 ; a mechanism for high IPL threads to generate cosole output.  
86 ; Use that mechanism in SMP\$SHUTDOWN\_CPU.  
87 ;  
88 ; X-62 PT00012 Pankaj Tandon 11-Jan-1988  
89 ; Insert in-line code in SMP\$SHUTDOWN\_CPU for correct  
90 ; adjustment of pointers in sanity timer chain.  
91 ;  
92 ; X-61 MSH0329 Michael S. Harvey 28-Oct-1987  
93 ; Don't assume that CPB\$C\_QUORUM is zero. Also, allow  
94 ; recognition of capability value zero when removing a  
95 ; process affinity by accepting 1's complement input  
96 ; instead of 2's complement input.  
97 ;  
98 ; Also, fix a synchronization problem; fix supplied by Rod G.  
99 ;  
100 ; X-60 SJF Stu Farnham 26-Oct-1987  
101 ; Take correct branch when SMP\$TIMEOUT finds a halted CPU.  
102 ;  
103 ; X-59 RNG5059 Rod Gamache 21-Oct-1987  
104 ; Release SCHED spinlock as early as possible when shutting  
105 ; down a CPU.  
106 ;  
107 ; X-58 SJF Stu Farnham 19-Oct-1987  
108 ; Check for halted CPUs in SMP\$TIMEOUT.  
109 ;  
110 ; X-57 RNG5057 Rod Gamache 5-Oct-1987  
111 ; Block re-scheduling while locking the CPU MUTEX.  
112 ;  
113 ; X-56 SJF Stu Farnham 24-Sep-1987  
114 ; Remove checks for process space and SMP enabled from

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 3  
X-72 HISTORY 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1)

```

115 ;           SMP$INVALID. Putting them in the INVALIDATE_TB macro will
116 ;           speed up pagefault processing.
117 ;
118 ;           SSA0001      Stan Amway      21-Sep-1987
119 ;           Change range checking in SMP$SETCAP (CLEAR path) to
120 ;           special case QUORUM capability, assumed to be CPB #0.
121 ;
122 ;           X-55      RNG5055      Rod Gamache      18-Sep-1987
123 ;           Fix missing label.
124 ;
125 ;           X-54      RNG5054      Rod Gamache      9-Sep-1987
126 ;           Add call to SMP$HALT_CPU when a CPU is to terminate itself.
127 ;
128 ;           X-53      RNG5053      Rod Gamache      26-Aug-1987
129 ;           Add benign state processing.
130 ;
131 ;           X-52      SJF           Stu Farnham      18-Aug-1987
132 ;           SMP$INVALID does not return properly for P0 addresses.
133 ;           Clean up the return, and fix a typo.
134 ;
135 ;           X-51      SJF           Stu Farnham      11-Aug-1987
136 ;           Fix register usage in SMP$INVALID
137 ;
138 ;           X-50      SF00050      Stephen Fiorelli 06-Aug-1987
139 ;           Fix Coroutines with smp$invalid.
140 ;
141 ;           X-49      SF00049      Stephen Fiorelli 06-Aug-1987
142 ;           Coroutines with smp$invalid.
143 ;
144 ;           X-48      SJF           Stu Farnham      17-July-1987
145 ;           Check for CPU in BOOT_REJECTED state or (in active set
146 ;           *AND* in RUN state) in SMP$SHUTDOWN_CPU. Check for target CPU
147 ;           in active set in SMP$SETAFF.
148 ;
149 ;           X-47      SJF           Stu Farnham      10-Jun-1987
150 ;           1. initialize SPL$L_TIMO_INT in OPAO devicelock
151 ;           2. Correct register save mask in SMP$SHUTDOWN_CPU
152 ;           to preserve R4
153 ;           3. Exit with correct status in all cases from $SETAFF/
154 ;           $SETCAP.
155 ;           4. Modify SMP$CALCAFF to correctly handle a null
156 ;           capability selection mask as input.
157 ;
158 ;           X-46      SJF           Stu Farnham      5-Jun-1987
159 ;           Modify SYNCH$INIT_ONCE to initialize timeout intervals of
160 ;           all spinlocks ate IPL <= 8 to SMP_LNGSPINWAIT.
161 ;
162 ;           X-45      RNG5045      Rod Gamache      3-Jun-1987
163 ;           Fix problem with bitmask usage in SMP$SETAFF.
164 ;
165 ;           X-44      RNG5044      Rod Gamache      2-Jun-1987
166 ;           Fix problem with SMP$SETCAP not correctly generating
167 ;           a full capability mask.
168 ;
169 ;           X-43      SJF           Stu Farnham      27-May-1987
170 ;           Allow for nested $SETCAP calls for the same capability
171 ;           without modifying capability database. This allows a

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 4  
X-72 HISTORY 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1)

```

172 ;           single clear capability request to reset the database
173 ;           correctly.
174 ;
175 ;           X-42   SJF           Stu Farnham           19-May-1987
176 ;           Add generic, kernel mode $STOP/CPU functions to SMP$
177 ;           SHUTDOWN_CPU.
178 ;
179 ;           X-41   SJF           Stu Farnham           22-Apr-1987
180 ;           Re-do part of edit X-39, which got dropped along the line.
181 ;
182 ;           X-40   SJF           Stu Farnham           22-Apr-1987
183 ;           Fix RESCHED request in SMP$SETAFF and SMP$SETCAP.
184 ;
185 ;           X-39   SJF           Stu Farnham           14-Apr-1987
186 ;           Fix SMP$SETAFF to index SMP$GW_AFFINITY_COUNT correctly.
187 ;           Change EXE$STOP_CPU to SMP$SHUTDOWN_CPU for naming consistency.
188 ;
189 ;           X-38   SJF           Stu Farnham           3-Apr-1987
190 ;           Add stub for EXE$STOP_CPU.
191 ;
192 ;           X-37   RNG5037       Rod Gamache           25-Mar-1987
193 ;           Cleanup SETCAP routine a little. Negative capability
194 ;           values are used to remove a capability.
195 ;
196 ;           X-36   RNG5036       Rod Gamache           24-Mar-1987
197 ;           Change refs from SMP$AR_SPNLKVEC to SMP$GL_SPNLKVEC.
198 ;           Change refs from SMP$AR_SPNLKCNT to SMP$GW_SPNLKCNT.
199 ;           Init SMP$GW_MIN_INDEX and SMP$GW_SPNLKCNT.
200 ;
201 ;           X-35   RNG5035       Rod Gamache           18-Mar-1987
202 ;           Copy IPL vector from LDAT to base image.
203 ;
204 ;           X-34   SJF           Stu Farnham           16-Mar-1987
205 ;           -33   Use SMP_LNGSPINWAIT as timeout interval for SCS,MMG and SCHED
206 ;           spinlocks to reflect long holding times for those locks.
207 ;
208 ;           X-32   SF0001        Stephen Fioreli       12-Mar-1987
209 ;           Place code in nonpaged psect.
210 ;
211 ;           X-31   SJF           Stu Farnham           4-Feb-1987
212 ;           Modify SMP$SETAFF and SMP$SETCAP to request a reschedule
213 ;           of a process whose affinity run mask has been changed
214 ;           if that process is current somewhere.
215 ;
216 ;           Teach SMP$TIMEOUT about the TIME_CONTROL sysgen param
217 ;           so it can detect SMP timeouts disabled.
218 ;
219 ;           Remove unused entrypoints SMP$FIND_SELF, SMP$FIND_OTHER,
220 ;           and SMP$SYNCH_TODR.
221 ;
222 ;           X-30   SJF           Stu Farnham           3-Feb-1987
223 ;           Fix various spellos.
224 ;
225 ;           X-29   WMC0029       Wayne Cardoza         28-Jan-1987
226 ;           Use indirect pointer to I/O structures.
227 ;
228 ;           X-28   RNG5028       Rod Gamache           27-Jan-1987

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 5  
X-72 HISTORY 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1)

|       |      |   |             |
|-------|------|---|-------------|
| 229 ; |      | Add SMP\$TIMEOUT routine.   |             |
| 230 ; |      |   |             |
| 231 ; | X-27 | SJF Stu Farnham   | 27-Jan-1987 |
| 232 ; |      | Fix typo.   |             |
| 233 ; |      |   |             |
| 234 ; | X-26 | SJF Stu Farnham   | 26-Jan-1987 |
| 235 ; |      | Add busywait timeouts. Add SMP\$INIT_SANITY. Do NOT                 |             |
| 236 ; |      | normalize wait intervals in order to save memory                    |             |
| 237 ; |      | references during busywaits.  |             |
| 238 ; |      |   |             |
| 239 ; | X-25 | SJF Stu Farnham   | 5-Jan-1987  |
| 240 ; |      | Initialize SPL\$L_TIMO_INT in SYNCH\$INIT_ONCE. Store               |             |
| 241 ; |      | normalized busywait and spinwait times in SMP\$GQ_SPINWAIT          |             |
| 242 ; |      | and _BUSYWAIT.  |             |
| 243 ; |      |   |             |
| 244 ; | X-24 | SJF Stu Farnham   | 30-Dec-1986 |
| 245 ; |      | Correct check for CPB #/ CPU ID for which to CLEAR                  |             |
| 246 ; |      | affinity.   |             |
| 247 ; |      |   |             |
| 248 ; | X-23 | SJF Stu Farnham   | 22-Dec-1986 |
| 249 ; |      | Rewrite affinity and capability routines, which were                |             |
| 250 ; |      | hopelessly broken.  |             |
| 251 ; |      |   |             |
| 252 ; | X-22 | SJF Stu Farnham   | 9-Dec-1986  |
| 253 ; |      | Remove redundant (un)lock of CPU mutex in SMP\$CALCAFF.             |             |
| 254 ; |      | Streamline revised affinity code.                                   |             |
| 255 ; |      |   |             |
| 256 ; | X-21 | SJF Stu Farnham   | 4-Dec-1986  |
| 257 ; |      | Bug fixes to affinity routines:                                     |             |
| 258 ; |      | - add missing RSB to SMP\$CALCAFF                                   |             |
| 259 ; |      | - fix broken register   |             |
| 260 ; |      | - fix broken branch from CALCAFF_INCLUSIVE                          |             |
| 261 ; |      |   |             |
| 262 ; | X-20 | RNG0020 Rod Gamache   | 5-Nov-1986  |
| 263 ; |      | Fix loop problem in SMP\$INVALID.                                   |             |
| 264 ; |      |   |             |
| 265 ; | X-19 | RNG0019 Rod Gamache   | 31-Oct-1986 |
| 266 ; |      | Move code that adjusts IPLs to SMPSTART modules.                    |             |
| 267 ; |      |   |             |
| 268 ; | X-18 | SJF Stu Farnham   | 30-Oct-1986 |
| 269 ; |      | Remove temporary code.  |             |
| 270 ; |      |   |             |
| 271 ; | X-17 | RNG0017 Rod Gamache   | 29-Oct-1986 |
| 272 ; |      | Add \$\$SPLCODEF macro call, optimize SMP\$INVALID for uniprocessor |             |
| 273 ; |      | and remove use of WHAMI macro.                                      |             |
| 274 ; |      |   |             |
| 275 ; | X-16 | RNG0016 Rod Gamache   | 28-Oct-1986 |
| 276 ; |      | Optimize SMP\$INVALID to return if SMP operation is not enabled.    |             |
| 277 ; |      |   |             |
| 278 ; | X-15 | SJF Stu Farnham   | 6-Oct-1986  |
| 279 ; |      | Add process affinity entrypoints SMP\$SETAFF, SMP\$SETCAP, and      |             |
| 280 ; |      | SMP\$CALCAFF(_INCLUSIVE).   |             |
| 281 ; |      |   |             |
| 282 ; | X-14 | RNG0014 Rod Gamache   | 15-Sep-1986 |
| 283 ; |      | Make SMP\$SWITCH_CPU handle TWP's. Remove temporary branch          |             |
| 284 ; |      | around SMP\$ADJUST_IPL routine.                                     |             |
| 285 ; |      |   |             |

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 6  
X-72 HISTORY 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1)

|       |       |         |   |             |
|-------|-------|---------|---|-------------|
| 286 ; | X-12  | RNG0012 | Rod Gamache   | 8-Sep-1986  |
| 287 ; |       |         | Change JSB to SMP\$ADJUST_IPL to BSBW.                    |             |
| 288 ; |       |         |   |             |
| 289 ; | X-11  | RNG0011 | Rod Gamache   | 8-Sep-1986  |
| 290 ; |       |         | Move HWCLK IPL adjustment into SYNCH\$INIT_ONCE routine.  |             |
| 291 ; |       |         |   |             |
| 292 ; | X-10  | MSH0277 | Michael S. Harvey   | 19-Aug-1986 |
| 293 ; |       |         | Speed up some code paths.                                 |             |
| 294 ; |       |         |   |             |
| 295 ; | X-9   | SJF     | Stu Farnham   | 1-Aug-1986  |
| 296 ; |       |         | Fix spellos.  |             |
| 297 ; |       |         |   |             |
| 298 ; | X-8   | SJF     | Stu Farnham   | 31-Jul-1986 |
| 299 ; |       |         | Add SMP\$SYNCH_TODR.                                      |             |
| 300 ; |       |         |   |             |
| 301 ; | X-7   | WMC0002 | Wayne Cardoza   | 29-Jul-1986 |
| 302 ; |       |         | Get rid of re-executeable flag.                           |             |
| 303 ; |       |         |   |             |
| 304 ; | X-6   | SJF     | Stu Farnham   | 28-Jul-1986 |
| 305 ; |       |         | Rename IPINT spinlock to INVALIDATE. Fix IPL of           |             |
| 306 ; |       |         | acquisition.  |             |
| 307 ; |       |         |   |             |
| 308 ; | X-5   | MSH0270 | Michael S. Harvey   | 25-Jul-1986 |
| 309 ; |       |         | Make sure that mutex is used to protect active CPU        |             |
| 310 ; |       |         | bitmask.  |             |
| 311 ; |       |         |   |             |
| 312 ; | X-3   | WMC0001 | Wayne Cardoza   | 24-Jul-1986 |
| 313 ; |       |         | Initialization routines must return status.               |             |
| 314 ; |       |         |   |             |
| 315 ; | X-2H8 | SJF     | Stu Farnham   | 19-Jun-1986 |
| 316 ; |       |         | Change IPL of IPINT acquire to 21.                        |             |
| 317 ; |       |         |   |             |
| 318 ; | X-2H7 | SJF     | Stu Farnham   | 16-Jun-1986 |
| 319 ; |       |         | Clean up synchronization in SMP\$INVALID.                 |             |
| 320 ; |       |         |   |             |
| 321 ; | X-1H2 | RNG0102 | Rod Gamache   | 6-Jun-1986  |
| 322 ; |       |         | Add initialization routine to setup DEVICE LOCKs for      |             |
| 323 ; |       |         | the MAILBOX CRB and all associated UCBs.                  |             |
| 324 ; |       |         |   |             |
| 325 ; | X-1F8 | SJF     | Stu Farnham   | 22-May-1986 |
| 326 ; |       |         | Modify SMP\$INVALID to use IP_ACK_WAIT macro, and         |             |
| 327 ; |       |         | to request IP interrupts ONLY for S0 addresses.           |             |
| 328 ; |       |         |   |             |
| 329 ; | X-1F7 | RNG1007 | Rod Gamache   | 22-May-1986 |
| 330 ; |       |         | Remove SMP\$PLM_RECORD routine.                           |             |
| 331 ; |       |         |   |             |
| 332 ; | X-1F6 | RNG1006 | Rod Gamache   | 16-May-1986 |
| 333 ; |       |         | Fix SMP\$INVALID instruction order.                       |             |
| 334 ; |       |         |   |             |
| 335 ; | X-1F5 | SJF     | Stu Farnham   | 14-May-1986 |
| 336 ; |       |         | Add SMP\$INVALID  |             |
| 337 ; |       |         |   |             |
| 338 ; | X-1F4 | RNG1004 | Rod Gamache   | 13-May-1986 |
| 339 ; |       |         | Add SMP\$PML_RECORD routine to record poor man's lockdown |             |
| 340 ; |       |         | pages.  |             |
| 341 ; |       |         |   |             |
| 342 ; | X-1F3 | RLRSMP  | Robert L. Rappaport                                       | 13-May-1986 |

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 7  
X-72 HISTORY 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1)

```
343 ;           Add SMP$IOPOST_IRP entrypoint.
344 ;
345 ;           X-1F2   RNG1002           Rod Gamache           24-Apr-1986
346 ;           Add SMP$SWITCH_CPU routine to fork to another CPU based
347 ;           on device affinity.
348 ;--
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 8  
X-72 DECLARATIONS 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (2)

```
350      .SBTTL  DECLARATIONS
351
352      $BIICDEF          ; Define BIIC offsets
353      $BOOSTATEDEF     ; Define bootstrap state flags
354      $CPBDEF          ; Define system capability #s.
355      $CPUDEF          ; Define per CPU data area
356      $CRBDEF          ; Define CRB offsets
357      $DYNDEF          ; Define structure types
358      $FKBDEF          ; Define fork blocks
359      $INIRTNDEF       ; Define initialization routine flag
360      $IPLDEF          ; Define IPLs
361      $IRPDEF          ; Define IRPs
362      $KA810DEF        ; Define KA820 node private space
363      $PCBDEF          ; Define process control block
364      $PRDEF           ; Define Processor Registers
365      $PR8SSDEF        ; Define KA820 processor registers
366      $PSLDEF          ; Define Processor Status Longword
367      $PTEDEF          ; Define page table entry
368      $$SPLDEF         ; Define spinlock structures
369      $$SPLCODDEF      ; Define spinlock indices
370      $$SSDEF          ; Define status returns
371      $$STATEDEF       ; Define scheduling states
372      $$TTYDEF         ; Define TTY WB structure
373      $UCBDEF          ; Define UCB offsets
374
375      .list  meb
376
377      DECLARE_PSECT  EXEC$NONPAGED_DATA
378      .PSECT  EXEC$NONPAGED_DATA,QUAD,WRT,NOEXE,PIC
379  INVALID_IPL:      ; Temporary storage to save IPL on stack
380      .LONG          ; for coroutine call in smp$invalid
381
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 9  
X-72 GET\_CURPCB - Return current PCB address 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (3)

```

383      .SBTTL GET_CURPCB - Return current PCB address
384 ;++
385 ; SMP$GET_CURPCB - Return current PCB address
386 ;
387 ;      This subroutine is used to find the PCB address that is currently
388 ;      executing on the CPU.
389 ;
390 ; Calling sequence:
391 ;
392 ;      JSB/BSBx SMP$GET_CURPCB
393 ;
394 ; Inputs:
395 ;
396 ;      None.
397 ;
398 ; Implicit Inputs:
399 ;
400 ;      CPU database from Interrupt Stack
401 ;
402 ; Outputs:
403 ;
404 ;      R4 = PCB address
405 ;
406 ; Implicit Outputs:
407 ;
408 ;      None
409 ;
410 ; Side Effects:
411 ;
412 ;      None.
413 ;
414 ;--
415
416      DECLARE_PSECT EXEC$NONPAGED_CODE
      .PSECT EXEC$NONPAGED_CODE, LONG, NOWRT, EXE, PIC
417
418      UNIVERSAL_SYMBOL SMP$GET_CURPCB
SMP$GET_CURPCB::
419 ;SMP$GET_CURPCB::
420      DSBINT ENVIRON=UNIPROCESSOR ; Raise IPL above SCHED
      MFPR S^#PR$_IPL, -(SP)
      MTPR #31, S^#PR$_IPL
421      find_cpu_data R4 ; Get per-CPU database address
      MFPR S^#PR$_ISP, R4
      BICL2 G^SMP$GL_BASE_MSK, R4
422      MOVL CPU$L_CURPCB(R4), R4 ; Return PCB address
423      ENBINT ; Restore IPL
      MTPR (SP)+, S^#PR$_IPL
424      RSB ; Return to caller

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 10  
X-72 SMP\$SWITCH\_CPU - Switch to another CPU b 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (4

```

426      .SBTTL SMP$SWITCH_CPU - Switch to another CPU base on device affinity
427 ;++
428 ; SMP$SWITCH_CPU - switch to another CPU
429 ;
430 ;      This subroutine is used to switch execution to another CPU as a
431 ;      fork thread.
432 ;
433 ; Calling sequence:
434 ;
435 ;      JSB/BSBx SMP$SWITCH_CPU
436 ;
437 ; Inputs:
438 ;
439 ;      R3 = TWP/IRP ADDRESS (USED AS FORK BLOCK)
440 ;      R5 = UCB ADDRESS /with affinity mask
441 ;
442 ; Implicit Inputs:
443 ;
444 ;      CPU database from Interrupt Stack
445 ;
446 ; Outputs:
447 ;
448 ;      none.
449 ;
450 ; Implicit Outputs:
451 ;
452 ;      None
453 ;
454 ; Side Effects:
455 ;
456 ;      R0 and R3 are destroyed.
457 ;
458 ;--
459
460      UNIVERSAL_SYMBOL      SMP$SWITCH_CPU
SMP$SWITCH_CPU::
461 ;SMP$SWITCH_CPU::
462      CMPB      #DYN$C_IRP,IRP$B_TYPE(R3) ; Is this an IRP?
463      BNEQ      40$ ; Br if no
464      MOVAB     IRP$C_CDRP(R3),R3 ; Point to CDRP part of IRP as FKB
465      MOVAB     #DYN$C_IRP,FKB$B_TYPE(R3) ; Set structure type
466      MOVAB     B^FORK_THREAD_IRP,FKB$L_FPC(R3) ; Save fork address
467 10$:      MOVAB     UCB$B_FLCK(R5),FKB$B_FLCK(R3) ; Copy fork lock index
468      MOVL      R5,FKB$L_FR3(R3) ; Save R5 in FORK BLOCK
469      POPL      FKB$L_FR4(R3) ; Save return PC in FORK BLOCK
470      FFS      #0,#32,UCB$L_AFFINITY(R5),R0 ; Find first good CPU
471      BNEQ      15$ ; If NEQ, found a CPU
472      MOVL      G^SMP$GL_PRIMID,R0 ; Affinity has been set to primary
473 15$:      MOVL      G^SMP$GL_CPU_DATA[R0],R0 ; Get address of CPU database
474      PUSHL     R1 ; Save scratch register
475      $INSQTI  (R3),CPU$Q_WORK_FQFL(R0),R1 ; Insert work pkt on end of queue
      CLRL      R1
30000$:
      INSQTI  (R3),CPU$Q_WORK_FQFL(R0)
      BCC      30001$
      AOBLSS   #900000,R1,30000$
      .WORD    ^XFEFF

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 11  
X-72 SMP\$SWITCH\_CPU - Switch to another CPU b 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (4

```

        .IIF IDN <FATAL>,<FATAL> , .WORD          BUG$_BADQHDR!4
30001$:
30002$:
476                                     ; Note: R1 is destroyed by $INSQTI
477      POPL      R1                    ; Restore R1
478      BBSSI     #CPU$V_WORK_FQP,CPU$L_WORK_REQ(R0),20$ ; Set work request bit
479 20$:
480      MOVL     CPU$L_PHY_CPUID(R0),R0 ; Get physical CPU ID
481      IPINT_CPU                    ; Generate IP Interrupt
482      JSB      G^SMP$INTPROC
483      RSB                                     ; Return to caller's caller
484 40$:      ; Turn TWP into a fork block
485
486      CMPB     #DYN$C_TWP,TTY$B_WB_TYPE(R3) ; Is this a TWP?
487      BNEQ     99$                    ; Br if no
488      ASSUME   FKB$B_FLCK EQ TTY$B_WB_FLCK
489      ASSUME   FKB$L_FPC EQ TTY$L_WB_FPC
490      ASSUME   FKB$L_FR3 EQ TTY$L_WB_FR3
491      ASSUME   FKB$L_FR4 EQ TTY$L_WB_FR4
492      MOVAB    B^FORK_THREAD_TWP,FKB$L_FPC(R3) ; Save fork address
493      BRB     10$                    ; Continue
494
495
496 99$:      BUG_CHECK INCONSTATE,FATAL ; Inconsistent database if here
        .WORD   ^XFEFF
        .IIF IDN <FATAL>,<FATAL> , .WORD          BUG$_INCONSTATE!4
497
498 ;+
499 ; Fork thread that executes after switching to other CPU via IRP.
500 ;
501 ; R5 = FORK BLOCK ADDRESS
502 ; FKB$L_FR3 = UCB ADDRESS
503 ; FKB$L_FR4 = Routine address
504 ;-
505
506 FORK_THREAD_IRP:
507      SUBL3    #IRP$C_CDRP,R5,R3      ; Backup to IRP
508      MOVL     IRP$C_CDRP+FKB$L_FR3(R3),R5 ; Get UCB address
509      JSB     @IRP$C_CDRP+FKB$L_FR4(R3) ; Call back original routine
510      RSB                                     ; Return to caller
511
512 ;+
513 ; Fork thread that executes after switching to other CPU via TWP.
514 ;
515 ; R5 = FORK BLOCK ADDRESS
516 ; FKB$L_FR3 = UCB ADDRESS
517 ; FKB$L_FR4 = Routine address
518 ;-
519
520 FORK_THREAD_TWP:
521      MOVL     R5,R3                    ; Restore TWP address to R3
522      MOVL     FKB$L_FR3(R3),R5        ; Get UCB address
523      JSB     @FKB$L_FR4(R3)          ; Call back original routine
524      RSB                                     ; Return to caller

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 12  
X-72 IOPOST\_IRP - Place IRP on Per-processor 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (5)

```

526      .SBTTL IOPOST_IRP - Place IRP on Per-processor IOPOST Q.
527 ;++
528 ; SMP$IOPOST_IRP - Place IRP on Per-processor IOPOST Q.
529 ;
530 ;      This subroutine is used to INSQ an IRP on the IOPOST queue for
531 ;      the current CPU.
532 ;
533 ; Calling sequence:
534 ;
535 ;      JSB/BSBx SMP$IOPOST_IRP
536 ;
537 ; Inputs:
538 ;
539 ;      R3 => IRP
540 ;
541 ; Implicit Inputs:
542 ;
543 ;      CPU database from Interrupt Stack
544 ;
545 ; Outputs:
546 ;
547 ;
548 ; Implicit Ouputs:
549 ;
550 ;      None
551 ;
552 ; Side Effects:
553 ;
554 ;      None
555 ;
556 ;--
557
558      UNIVERSAL_SYMBOL      SMP$IOPOST_IRP
SMP$IOPOST_IRP::
559 ;SMP$IOPOST_IRP::
560      PUSHL   R0              ; Save register.
561      ASSUME  SMP$V_ENABLED EQ 0
562      BLBS   G^SMP$GL_FLAGS,20$ ; Br if a multi-processor
563      $INSQTI (R3),G^IOC$GQ_POSTIQ ; INSERT PACKET ON QUEUE
      CLRL   R0
30004$:
      INSQTI (R3),G^IOC$GQ_POSTIQ
      BCC    30005$
      AOBLSS #900000,R0,30004$
      .WORD  ^XFEFF
      .IIF IDN <FATAL>,<FATAL> , .WORD      BUG$_BADQHDR!4
30005$:
30006$:
564      SOFTINT S^#IPL$_IOPOST ; REQUEST FORK
      MTPR   S^#IPL$_IOPOST,S^#PR$_SIRR
565      POPL   R0              ; Restore R0.
566      RSB    ; RETURN
567
568 20$: ; This is a multiprocessor, don't allow rescheduling so that
569 ; we guarantee the SOFTINT will happen on the correct CPU.
570
571      SAVIPL -(SP)          ; Save the current IPL

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 13  
X-72 IOPOST\_IRP - Place IRP on Per-processor 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (5)

```

MFPR      S^#PR$_IPL,-(SP)
572      CMPB      (SP),S^#IPL$_IOPOST      ; Is current IPL high enough?
573      BGEQ      40$                      ; Br if yes, continue
574      SETIPL    S^#IPL$_IOPOST,ENVIRON=UNIPROCESSOR ; Else, raise IPL
MTPR      S^#IPL$_IOPOST,S^#PR$_IPL
575 40$:   $INSQTI  (R3),G^IOC$GQ_POSTIQ      ; INSERT PACKET ON QUEUE
          CLRL      R0
          30007$:
          INSQTI  (R3),G^IOC$GQ_POSTIQ
          BCC      30008$
          AOBLS    #900000,R0,30007$
          .WORD    ^XFEFF
          .IIF IDN <FATAL>,<FATAL> , .WORD      BUG$_BADQHDR!4
          30008$:
          30009$:
576      BNEQ      60$                      ; BRANCH IF QUEUE IS NOT EMPTY
577      FIND_CPU_DATA  R0                  ; GET ADDRESS OF PER-CPU DATA
          MFPR      S^#PR$_ISP,R0
          BICL2    G^SMP$GGL_BASE_MSK,R0
578      CMPL      G^SMP$GGL_PRIMID,CPU$L_PHY_CPUID(R0) ; Are we the primary?
579      BNEQ      80$                      ; Br if not primary
580      SOFTINT   S^#IPL$_IOPOST          ; REQUEST FORK
          MTPR      S^#IPL$_IOPOST,S^#PR$_SIRR
581 60$:   ENBINT      ; RESTORE IPL
          MTPR      (SP)+,S^#PR$_IPL
582      POPL      R0                      ; Restore R0.
583      RSB       ; RETURN
584
585 80$:   ; This is not the primary CPU on an MP
586
587      IPINT_CPU   IOPOST,G^SMP$GGL_PRIMID ; Tell the primary to do a softint
          PUSHL    R0
          MOVL     G^SMP$GGL_PRIMID,R0
          PUSHL    R1
          MOVAL    G^SMP$GGL_CPU_DATA,R1
          MOVL     (R1)[R0],R1
          BBSSI    S^#CPU$V_IOPOST,CPU$L_WORK_REQ(R1),30010$
          30010$:
          POPL     R1
          JSB      G^SMP$INTPROC
          POPL     R0
588      BRB       60$                    ; Non-optimal case, don't bother

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 14  
X-72 SMP\$INVALID - invalidate a single TB ent 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (6

```

590      .SBTTL SMP$INVALID - invalidate a single TB entry.
591 ;++
592 ; FUNTIONAL DESCRIPTION:
593 ;
594 ; SMP$INVALID is called to INVALIDate a single TB entry.
595 ;
596 ; CALLING SEQUENCE:
597 ;
598 ;      BSB/JSB SMP$INVALID
599 ;
600 ; INPUT PARAMETERS:
601 ;
602 ;      R2      =      Address for which to invalidate TB
603 ;
604 ; OUTPUT PARAMETERS:
605 ;
606 ;      NONE
607 ;
608 ; Side effects:
609 ;
610 ;      The other processors in an SMP system are interrupted to invalidate
611 ;      the corresponding entry in their TBs. The routine waits for all
612 ;      processors to perform the invalidate, and then issues a TBACK and
613 ;      continues.
614 ;
615 ;--
616
617      .ENABL LSB
618
619      UNIVERSAL_SYMBOL      SMP$INVALID
SMP$INVALID::
620 ;SMP$INVALID::
621      PUSHL      R0
622      LOCK      LOCKNAME=INVALIDATE,-      ; synch the TB INVALIDATE database
623      PRESERVE=NO,-
624      SAVIPL--(SP)
      .SAVE      LOCAL_BLOCK
      .PSECT     $ABS$,ABS
      .RESTORE
      MFPR      S^#PR$_IPL,-(SP)
      BLBC      G^SMP$GL_FLAGS,30013$
      MOVZBL    S^#SPL$_INVALIDATE,R0
      JSB      G^SMP$ACQUIRE
      BRB      30014$
30013$:
      MTPR      S^#IPL$_INVALIDATE,S^#PR$_IPL
30014$:
625
626      LOCK      MUTEX=SMP$GL_CPU_MUTEX,-; Lock the CPU bitmasks
627      PRESERVE=NO,-
628      SHARE=YES      ; to prevent them from changing
      .SAVE      LOCAL_BLOCK
      .PSECT     $ABS$,ABS
      .RESTORE
      MFPR      S^#PR$_IPL,-(SP)
      MTPR      #31,S^#PR$_IPL
      MFPR      S^#PR$_ISP,R0

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 15  
X-72 SMP\$INVALID - invalidate a single TB ent 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (6

```

        BICL2  G^SMP$GL_BASE_MSK,RO
        CLRL  -(SP)
30017$:
        MULL3  G^SGN$GL_SMP_SPINWAIT,-
              CPU$L_TENUSEC(RO), (SP)
        MULL  CPU$L_UBDELAY(RO), (SP)
30018$:
        BBC  #MTX$V_INTERLOCK,G^SMP$GL_CPU_MUTEX,30020$
        BBS  #CPU$V_BUGCHK,-
              CPU$L_WORK_REQ(RO),30019$
        SOBGTR (SP),30018$
        JSB  G^SMP$TIMEOUT
        BRW  30017$
30019$: BUG_CHECK  CPUEXIT,FATAL
              .WORD  ^XFEFF
              .IIF IDN <FATAL>,<FATAL> , .WORD  BUG$_CPUEXIT!4
30020$: BBSSI  #MTX$V_INTERLOCK,G^SMP$GL_CPU_MUTEX,30017$
        ADDL  #4,SP
        INCB  CPU$B_CPUMTX(RO)
        INCW  G^SMP$GL_CPU_MUTEX
        BBCCI  #MTX$V_INTERLOCK,G^SMP$GL_CPU_MUTEX,30021$
30021$: ENBINT
              MTPR  (SP)+,S^#PR$_IPL
629
630  IP_ACK_WAIT  INTBIT=INV_TBS,-
631  INS1=<MOVL R2,G^SMP$GL_INVALID>,-
632  ACQUIRE=NO
        BLBC  G^SMP$GL_FLAGS,30023$
        MOVL R2,G^SMP$GL_INVALID
        MOVQ  RO,-(SP)
        MFPR  S^#PR$_ISP,RO
        BICL2  G^SMP$GL_BASE_MSK,RO
        ASHL  CPU$L_PHY_CPUID(RO),#1,-
              G^SMP$GL_ACK_MASK
        PUSHL  R2
        MOVL  S^#CPU$V_INV_TBS,R2
        JSB  G^SMP$INTALL_BIT
        POPL  R2
        CLRL  -(SP)
30024$: MULL3  G^SGN$GL_SMP_SPINWAIT,-
              CPU$L_TENUSEC(RO), (SP)
        MULL  CPU$L_UBDELAY(RO), (SP)
30025$: BICL3  G^SMP$GL_ACK_MASK,-
              G^SMP$GL_ACTIVE_CPUS,R1
        BEQL  30027$
        BBS  #CPU$V_BUGCHK,-
              CPU$L_WORK_REQ(RO),30026$
        SOBGTR (SP),30025$
        JSB  G^SMP$TIMEOUT
        BRW  30024$
30026$: BUG_CHECK  CPUEXIT,FATAL
              .WORD  ^XFEFF
              .IIF IDN <FATAL>,<FATAL> , .WORD  BUG$_CPUEXIT!4
30027$: ADDL  #4,SP
        MOVQ  (SP)+,RO
30023$:

```

633

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 16  
X-72 SMP\$INVALID - invalidate a single TB ent 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (C

```

634      MOVL      (SP)+, INVALID_IPL      ; Get the stack straight for coroutine call
635
636      POPL      RO                       ; request ip interrupts and wait
                                           ; restore register around coroutine
637
638      JSB       @(SP)+                   ; return to caller as coroutine to
639                                           ; modify PTE
640      PUSHR     #^M<RO,R1,R2>
641      MOVL      INVALID_IPL,-(SP)       ; Place the IPL back on stack for UNLOCK
642 ;
643 ; set TBACK bit in each CPUs WORKREQ longword
644 ;
645      find_cpu_data RO                   ; Get address of CPU data area
        MFPR     S^#PR$ _ISP,RO
        BICL2    G^SMP$GL_BASE_MSK,RO
646      MOVL      CPU$ _L_PHY_CPUID (RO),RO ; Get current CPU ID
647      ASHL      RO,#1,RO                 ; Convert to bitmask
648      BICL3     RO,G^SMP$GL_ACTIVE_CPUS,R1 ; Clear bit for this CPU
649      BEQL      40$                      ; No other CPUs
650      MOVAL     G^SMP$GL_CPU_DATA,R2     ; Get address of CPU data base vector
651 20$:      FFS      #0,#32,R1,RO        ; Find next CPU ID
652      BEQL      40$                      ; Finished
653      BBCC      RO,R1,30$                ; Clear this CPU's mask
654 30$:      MOVL      (R2)[RO],RO        ; Target CPUs data area
655      BBSI      #CPU$V_TBACK,-
656      CPU$ _L_WORK_REQ (RO),20$ ; Set work request bit
657      BRB       20$
658 40$:
659      UNLOCK    MUTEX=SMP$GL_CPU_MUTEX,-; Unlock CPU bitmasks
660      PRESERVE=NO,-
661      SHARE=YES
        MFPR     S^#PR$ _IPL,-(SP)
        MTPR     #31,S^#PR$ _IPL
        MFPR     S^#PR$ _ISP,RO
        BICL2    G^SMP$GL_BASE_MSK,RO
        CLRL     -(SP)
30028$:     MULL3    G^SGN$GL_SMP_SPINWAIT,-
        CPU$ _L_TENUSEC (RO), (SP)
        MULL     CPU$ _L_UBDELAY (RO), (SP)
30029$:     BBC      #MTX$V_INTERLOCK,G^SMP$GL_CPU_MUTEX,30031$
        BBS       #CPU$V_BUGCHK,-
        CPU$ _L_WORK_REQ (RO),30030$
        SOBGTR   (SP),30029$
        JSB      G^SMP$TIMEOUT
        BRW      30028$
30030$:     BUG_CHECK    CPUEXIT,FATAL
        .WORD    ^XFEFF
        .IIF IDN <FATAL>,<FATAL> , .WORD    BUG$ _CPUEXIT!4
30031$:     BBSI      #MTX$V_INTERLOCK,G^SMP$GL_CPU_MUTEX,30028$
        ADDL      #4,SP
        DECB      CPU$B_CPUMTX (RO)
        DECW      G^SMP$GL_CPU_MUTEX
        BBCCI     #MTX$V_INTERLOCK,G^SMP$GL_CPU_MUTEX,30032$
30032$:
        MTPR     (SP)+,S^#PR$ _IPL
662
663      UNLOCK    LOCKNAME=INVALIDATE,-
664      PRESERVE=NO,-

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 17  
X-72 SMP\$INVALID - invalidate a single TB ent 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (6

```
665          NEWIPL=(SP)+
          BLBC   G^SMP$GL_FLAGS,30035$
          MOVZBL S^#SPL$C_INVALIDATE,R0
          JSB    G^SMP$RELEASE
30035$:
          MTPR   (SP)+,S^#PR$_IPL
666
667          POPR   #^M<R0,R1,R2>          ; Restore registers
668
669          RSB
670
671          .DSABL LSB
```



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 18  
X-72 SMP\$TIMEOUT - SMP timeout processing rou 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (7

```
673      .SBTTL SMP$TIMEOUT - SMP timeout processing routine
674 ;++
675 ; FUNCTIONAL DESCRIPTION:
676 ;
677 ;     This routine is called when a timeout is believed necessary. All
678 ;     possible conditions that can account for a timeout are checked -
679 ;     such as timeouts disabled. If there is a good reason that the timeout
680 ;     can continue, then we will proceed else bugcheck.
681 ;
682 ; INPUTS:
683 ;     NONE.
684 ;
685 ; OUTPUTS:
686 ;     Return if we may continue, else bugcheck.
687 ;
688 ;--
689
690      UNIVERSAL_SYMBOL      SMP$TIMEOUT
SMP$TIMEOUT::
691 ;;SMP$TIMEOUT::
692      PUSHL      R0
693      JSB       G^SMP$CONTROLP_CPUS      ; get halted CPU bitmask
694      TSTL      R0                        ; are any CPUs halted?
695      BNEQ      10$                       ; yes
696      BBC       #EXE$V_NOSPINWAIT,-
697              g^EXE$GL_TIME_CONTROL,100$ ; Bugcheck if timeouts not disabled
698 10$:      MOVL      (SP)+,R0
699      RSB
700
701 100$:     BUG_CHECK      CPUSPINWAIT,FATAL
              .WORD      ^XFEFF
              .IIF IDN <FATAL>,<FATAL> , .WORD      BUG$_CPUSPINWAIT!4
702
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 19  
X-72 INIT\_SANITY - Initialize SMP sanity time 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (8

```
704      .SBTTL INIT_SANITY - Initialize SMP sanity timer pointer in CPU database
705 ;++
706 ; SMP$INIT_SANITY - Initialize SMP sanity timer pointer in CPU database
707 ;
708 ;      This subroutine is used to find the active CPU with the next lowest
709 ;      CPU ID, and adjust its sanity timer pointer relative to that of
710 ;      a CPU joining the active set.
711 ;
712 ; Calling sequence:
713 ;
714 ;      JSB/BSBx SMP$INIT_SANITY
715 ;
716 ; Inputs:
717 ;
718 ;      R2 = Address of the CPU database of the CPU joining the
719 ;      active set.
720 ;
721 ; Implicit Inputs:
722 ;
723 ;      Active set and associated CPU databases.
724 ;
725 ;      It is assumed that, since the calling thread is modifying the
726 ;      active set, the CPU mutex is held by the caller.
727 ;
728 ; Outputs:
729 ;
730 ;      CPU$W_SANITY_TIMER and CPU$W_SANITY_TICKS are initialized for this
731 ;      CPU.
732 ;
733 ;      CPU$L_TPOINTER of the active CPU with the next lowest CPU id
734 ;      is copied to the joining CPU's TPOINTER, and the active CPU's
735 ;      TPOINTER set to point to the CPU$W_SANITY_TIMER of the joining
736 ;      CPU.
737 ;
738 ; Implicit Outputs:
739 ;
740 ;      None
741 ;
742 ; Side Effects:
743 ;
744 ;      None.
745 ;
746 ;--
747
748      UNIVERSAL_SYMBOL      SMP$INIT_SANITY
SMP$INIT_SANITY::
749 ;SMP$INIT_SANITY::
750
751      PUSHR      #^M<R3,R4,R5,R6>
752 ;
753 ; initialize this CPU's sanity timer and tick count
754 ;
755      MOVW      g^SGN$GW_SMP_SANITY_CNT,-
756      CPU$W_SANITY_TIMER(R2)
757      MOVW      g^SGN$GW_SMP_TICK_CNT,-
758      CPU$W_SANITY_TICKS(R2)
759 ;
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 20  
X-72 INIT\_SANITY - Initialize SMP sanity time 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (6

760 ; The TPOINTER of each active CPU points to the sanity timer of the active  
761 ; CPU with the next highest CPU ID. When a CPU joins the active set, it must  
762 ; find the CPU with the next lower CPU ID; that CPU's TPOINTER will be point-

763 ; ing to the joining CPUs higher neighbor. The joining CPU will copy the  
764 ; low neighbor's TPOINTER to its own (i.e., point itself to its high neighbor),  
765 ; and point the low neighbor's TPOINTER to the sanity timer of the joining  
766 ; CPU.

```

767 ;
768     MOVL     CPU$L_PHY_CPUID(R2),R3
769 5$:     MNEGL  R3,R4
770     ROTL     R4,g^SMP$GSL_ACTIVE_CPUS,R4      ; rotate our bit into low position
771     BICL     #1,R4                             ; mask off our bit
772 10$:    FFS     #0,#32,R4,R5                  ; scan for next active CPU
773     BEQL     20$                               ; no other
774     ADDL     R3,R5                             ; found CPU id is (bit position
775     BICL     #^C<31>,R5                       ; + abs(shift count)) MOD 32
776     MOVAL    g^SMP$GSL_CPU_DATA,R6           ; CPU database vector address
777     MOVL     (R6)(R5),R6                       ; CPU database address of neighbor
778     MOVL     CPU$L_TPOINTER(R6),-
779     CPU$L_TPOINTER(R2)                       ; adopt the CPU neighbor was watchin
780     MOVAW    CPU$W_SANITY_TIMER(R2),-
781     CPU$L_TPOINTER(R6)                       ; point neighbors TPOINTER at us
782 ;

```

783 ; it is possible that our neighbor left the active set while the above  
784 ; code ran. Verify that it is still present before returning.

```

785 ;
786     ROTL     R3,#1,R4                         ; form bitmask
787     BITL     R4,g^SMP$GSL_ACTIVE_CPUS
788     BEQL     5$
789 15$:    POPR     #^M<R3,R4,R5,R6>
790     RSB

```

791  
792 20\$:

793 ;  
794 ; there is no active CPU but this one. point our TPOINTER at out SANITY  
795 ; timer.

```

796 ;
797     MOVAW    CPU$W_SANITY_TIMER(R2),-
798     CPU$L_TPOINTER(R2)
799     BRB     15$

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 21  
X-72 SMP\$\$SHUTDOWN\_CPU - Final actions associa 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (5

```
801      .SBTTL SMP$$SHUTDOWN_CPU - Final actions associated with stopping a CPU.
802
803 ;++
804 ;
805 ; FUNCTIONAL DESCRIPTION:
806 ;
807 ;     This routine provides the kernel mode CPU specific functions
808 ;     required to implement the $STOP/CPU DCL command.
809 ;
810 ; CALLING SEQUENCE:
811 ;
812 ;     BSB/JSB      SMP$$SHUTDOWN_CPU
813 ;
814 ; INPUTS:
815 ;
816 ;     R1 = CPU ID to be stopped
817 ;     R2 = 0 -> verify that CPU should be stopped before stopping
818 ;     <> 0 -> skip checks before stopping (OVERRIDE_CHECKS mode)
819 ;     R4 = process PCB address
820 ;
821 ; IMPLICIT INPUTS:
822 ;
823 ;     Scheduling/affinity databases
824 ;     SMP configuration/membership database
825 ;
826 ; OUTPUTS:
827 ;
828 ;     R0 : LBS -> CPU stopped
829 ;     LBC -> CPU not stopped (checks failed)
830 ;     R1,R2 destroyed
831 ;
832 ; IMPLICIT OUTPUTS:
833 ;
834 ;     CPU stopped if appropriate checks passed.
835 ;     Process returned to appropriate COM queue
836 ;
837 ;--
838
839 CR      =      ^X0D      ; Carriage Return character
840 LF      =      ^X0A      ; Line Feed character
841
842 ;
843 ; Define some messages that may be issued when other CPUs begin executing.
844 ;
845 STOP_DONE:
846      .ASCIC <CR><LF>\%SMP-I-STOPPED, CPU #\

847 STOP_DONE2:
848      .ASCIC \ has been stopped.\

849
850
851 ;
852 ; The following translation table is used in converting binary data into
853 ; their equivalent ASCII strings.
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 22  
X-72 SMP\$SHUTDOWN\_CPU - Final actions associa 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (S

```

854 ;
855 CH_TABLE:
856     .ASCII /00/
857     .ASCII /01/
858     .ASCII /02/
859     .ASCII /03/
860     .ASCII /04/
861     .ASCII /05/
862     .ASCII /06/
863     .ASCII /07/
864     .ASCII /08/
865     .ASCII /09/
866     .ASCII /10/
867     .ASCII /11/
868     .ASCII /12/
869     .ASCII /13/
870     .ASCII /14/
871     .ASCII /15/
872     .ASCII /16/
873     .ASCII /17/
874     .ASCII /18/
875     .ASCII /19/
876     .ASCII /20/
877     .ASCII /21/
878     .ASCII /22/
879     .ASCII /23/
880     .ASCII /24/
881     .ASCII /25/
882     .ASCII /26/
883     .ASCII /27/
884     .ASCII /28/
885     .ASCII /29/
886     .ASCII /30/
887     .ASCII /31/
888
889     UNIVERSAL_SYMBOL      SMP$SHUTDOWN_CPU
SMP$SHUTDOWN_CPU::
890 ;SMP$SHUTDOWN_CPU::
891     SAVIPL    -(SP)
                MFPR      S^#PR$_IPL, -(SP)
892     PUSHL    R1                ; save register
893     PUSHL    R2                ; save register
894     MOVL     G^SMP$GL_CPU_DATA[R1], R2
895     JSB      g^SMP$STOP_CPU    ; perform CPU-specific actions
896 ;
897 ; SMP$STOP_CPU may have placed the CPU in STOPPED state. Check for that
898 ; case before continuing. The STOP/CPU code should have weeded out all
899 ; other invalid transitions.
900 ;
901     CMPB     CPU$B_STATE(R2), #CPU$C_STOPPED
902     BNEQ     30$
903 15$: MOVL     #SS$_NORMAL, R0
904 20$: ADDL     #8, SP
905     BRW      340$
906 ;
907 ; a CPU must either be in (RUN state and active set) or BOOT_REJECTED state
908 ;

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 23  
X-72 SMP\$SHUTDOWN\_CPU - Final actions associa 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (9

```

909 30$:   CMPB   CPU$B_STATE(R2),-
910       #CPU$C_BOOT_REJECTED           ; boot_rejected?
911       BNEQ   32$                      ; no
912       MOVB   #CPU$C_STOPPED,-         ; reset state
913       CPU$B_STATE(R2)
914       BRB    15$                      ; return to caller
915 32$:   CMPB   CPU$B_STATE(R2),#CPU$C_RUN ; run state?
916       BNEQ   35$                      ; no, error
917       BBS    CPU$L_PHY_CPUID(R2),-    ; allow stop if also
918       G^SMP$GL_ACTIVE_CPUS,-        ; in active set.
919       40$
920 35$:   MOVL   #SS$_DEVOFFLINE,R0
921       BRB    20$
922       MOVB   #CPU$C_STOPPING,-
923       CPU$B_STATE(R2)                 ; note stop in progress
924 ;
925 ; set affinity to CPU to be stopped
926 ;
927 40$:   CLRL   -(SP)                    ; no return data
928       PUSHL  #CPB$M_FLAG_CHECK_CPU    ; make sure we can run
929       PUSHL  CPU$L_PHY_CPUID(R2)      ; CPU ID
930       PUSHL  #CPB$M_EXPLICIT_AFFINITY
931       PUSHL  R4                        ; PCB
932       CALLS  #5,G^SCH$REQUIRE_CAPABILITY
933       BLBC   R0,20$                   ; failed
934 ;
935 ; prevent other processes from running on this CPU
936 ;
937       SETIPL #IPL$_RESCHED,ENVIRON=UNIPROCESSOR
          MTPR   #IPL$_RESCHED,S^#PR$_IPL
938 ;
939 ; clear affinity of this process for this CPU
940 ;
941       CLRQ   -(SP)                    ; no return data or flags
942       PUSHL  CPU$L_PHY_CPUID(R2)      ; CPU ID
943       PUSHL  #CPB$M_EXPLICIT_AFFINITY
944       PUSHL  R4                        ; PCB
945       CALLS  #5,G^SCH$RELEASE_CAPABILITY
946       BLBC   R0,20$
947 ;
948 ; Acquire the scheduling spinlock .
949 ; Test to see if checks have been overridden. If not, check that this CPU is
950 ; not the last one providing a capability
951 ; which is currently in use, and that there is no process with hard affinity
952 ; for this CPU. Note that it is sufficient to lock SCHED (and not necessary
953 ; to hold the CPU mutex) during these checks as SCHED is acquired before
954 ; the affinity database is modified.
955 ;
956       LOCK   LOCKNAME=SCHED,-
957       PRESERVE=NO                      ; freeze affinities
          BLBC   G^SMP$GL_FLAGS,30040$
          MOVZBL S^#SPL$C_SCHED,R0
          JSB   G^SMP$ACQUIRE
          BRB   30041$
30040$:
          MTPR   S^#IPL$_SCHED,S^#PR$_IPL
30041$:

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 24  
X-72 SMP\$SHUTDOWN\_CPU - Final actions associa 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (S

```

958      TSTL      (SP)+          ; test for OVERRIDE_CHECKS
959      BNEQ      110$          ; OVERRIDE_CHECKS enabled, skip checks
960      TSTW      CPU$W_HARDAFF (R2) ; anyone with hard affinity?
961      BEQL      110$          ; no, go ahead
962
963 100$: UNLOCK   LOCKNAME=SCHED,- ; release SCHED so other CPUs can schedule
964      PRESERVE=NO          ; Don't preserve R0
      BLBC      G^SMP$GL_FLAGS,30046$
      MOVZBL   S^#SPL$C_SCHED,R0
      JSB      G^SMP$RELEASE
30046$:
965      CLRL      R0          ; return bad status
966      ADDL      #4,SP       ; clear stack
967      BRW      340$        ; branch assist
968 ;
969 ; Now we know that there is no process which will be left unable
970 ; to be scheduled if we remove this CPU, or that we have overridden
971 ; checks for such cases. Output a console message,go to IPL 31, join the
972 ; override set, remove the CPU from the active set and the capability
973 ; database, place the process back in the appropriate COM queue, release
974 ; SCHED, *LEAVE* the override set, and loop forever.
975 ;
976 110$: CLRL      -(SP)       ; no returned data
977      MCOML     #0,-(SP)     ; all capabilities
978      PUSHL    CPU$L_PHY_CPUID (R2) ; CPU ID
979      CALLS    #3,G^SCH$REMOVE_CPU_CAP
980
981 127$: POPL     R1          ; Clean up stack
982      MOVPSL   -(SP)       ; Come back to this IPL
983      PUSHAL   350$        ; Resume here
984 ;
985 ; now make this process available for scheduling onto other CPUs
986 ;
987      JSB      G^SCH$CUR_TO_COM
988      UNLOCK   LOCKNAME=SCHED ; release SCHED so other CPUs can schedule
      BLBC      G^SMP$GL_FLAGS,30051$
      PUSHL    R0
      MOVZBL   S^#SPL$C_SCHED,R0
      JSB      G^SMP$RELEASE
      POPL     R0
30051$:
989
990      PUSHAB   STOP_DONE2   ; Get secondary message address
991      PUSHAB   STOP_DONE    ; Get primary message address
992      FIND_CPU_DATA R6
      MFPR     S^#PR$_ISP,R6
      BICL2    G^SMP$GL_BASE_MSK,R6
993      BSBW     MESSAGE
994      ADDL     #8,SP        ; pop parameters
995      SETIPL   #IPL$_POWER,ENVIRON=UNIPROCESSOR
      MTPR     #IPL$_POWER,S^#PR$_IPL
996      BBSSI    R1,G^SMP$GL_OVERRIDE,140$ ; join override
997 140$: BBCCI    R1,G^SMP$GL_ACTIVE_CPUS,150$ ; leave active
998 150$: LOCK    MUTEX=SMP$GL_CPU_MUTEX,-
999      PRESERVE=NO
      MFPR     S^#PR$_IPL,-(SP)
      MTPR     #31,S^#PR$_IPL

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 25  
X-72 SMP\$SHUTDOWN\_CPU - Final actions associa 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (5

```

MFPR      S^#PR$ ISP,RO
BICL2     G^SMP$GL_BASE_MSK,RO
CLRL      -(SP)
30054$:
MULL3     G^SGN$GL_SMP_SPINWAIT,-
          CPU$L_TENUSEC(R0), (SP)
MULL      CPU$L_UBDELAY(R0), (SP)
30055$:
CMPW      G^SMP$GL_CPU_MUTEX,#-1
BEQL      30057$
BBS       #CPU$V_BUGCHK,-
          CPU$L_WORK_REQ(R0),30056$
SOBGTR    (SP),30055$
JSB       G^SMP$TIMEOUT
BRW       30054$
30056$:   BUG_CHECK      CPUEXIT,FATAL
          .WORD          ^XFEFF
          .IIF IDN <FATAL>,<FATAL> , .WORD          BUG$_CPUEXIT!4
30057$:   BSSI          #MTX$V_INTERLOCK,G^SMP$GL_CPU_MUTEX,30054$
          CMPW          G^SMP$GL_CPU_MUTEX,#-1
          BEQL          30059$
          BCCI          #MTX$V_INTERLOCK,G^SMP$GL_CPU_MUTEX,30058$
30058$:   BRB          30054$
30059$:
          ADDL          #4,SP
1000 ;
1001 ; Delete from sanity timer linked list
1002 ;
1003 165$:  MOVL          G^SMP$GL_CPU_DATA[R1],R2          ; R2 is the cpu dbase pointer
1004      MOVL          CPU$L_PHY_CPUID(R2),R3             ; CPU id,
1005      MNEGL         R3,R4
1006      ROTL          R4,g^SMP$GL_ACTIVE_CPUS,R4        ; rotate my bit into least sig. posn
1007      BICL          #1,R4                              ; mask off least sig. posn.
1008 166$:  FFS          #0,#32,R4,R5                     ; find next higher active cpu
1009      BEQL          170$                               ; not possible, since atleast one more sho
1010      ADDL          R3,R5                              ; get my neighbours' id
1011      BICL          #^C<31>,R5                        ; take mod 32, R5 has the id
1012      MOVL          G^SMP$GL_CPU_DATA[R5],R6          ; cpu dbase address
1013      MOVL          CPU$L_TPOINTER(R2),-
1014      CPU$L_TPOINTER(R6)                              ; give him my pointer
1015 ;
1016 ;
1017 ;
1018 170$:  UNLOCK      MUTEX=SMP$GL_CPU_MUTEX,-
1019      PRESERVE=NO
          BCCI          #MTX$V_INTERLOCK,G^SMP$GL_CPU_MUTEX,30060$
30060$:
          BLBC          G^EXE$GL_NS_FLAGS,30069$
          MFPR          #^X92,-4(SP)
30069$:
          MTPR          #0,S^#PR$ TBIA
          MTPR          (SP)+,S^#PR$ IPL
1020      MOVB          #CPU$C_STOPPED,CPU$B_STATE(R2) ; indicate STOPPED
1021      BCCI          R1,G^SMP$GL_OVERRIDE,320$ ; leave override
1022 320$:
1023 ;
1024 ; Perform CPU-specific shutdown of CPU

```



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SMPCUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 26  
X-72 SMP^SHUTDOWN\_CPU - Final actions associa 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (5

```
1025 ;  
1026     JSB     G^SMP$HALT_CPU           ; perform CPU-specific halt  
1027  
1028 ;  
1029 ; Drop IPL and return error status in R0  
1030 ;  
1031 340$: ENBINT  
                MTPR     (SP)+, S^#PR$_IPL  
1032     RSB  
1033 ;  
1034 350$: MOVL  #SS$_NORMAL, R0  
1035     BRB     340$
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 27  
X-72 SMP\$INITIATE\_BENIGN - Initiate a benign 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (10

```

1037      .SBTTL  SMP$INITIATE_BENIGN - Initiate a benign state
1038 ;++
1039 ; FUNTIONAL DESCRIPTION:
1040 ;
1041 ; SMP$INITIATE_BENIGN - Request a benign state
1042 ; SMP$TERMINATE_BENIGN - Leave a benign state
1043 ;
1044 ; ENVIRONMENT:
1045 ;
1046 ;      Kernel mode, mapping must be enabled
1047 ;
1048 ; CALLING SEQUENCE:
1049 ;
1050 ;      JSB      G^SMP$INITIATE_BENIGN
1051 ;      JSB      G^SMP$TERMINATE_BENIGN
1052 ;
1053 ; INPUT PARAMETERS:
1054 ;
1055 ;      None
1056 ;
1057 ; OUTPUT PARAMETERS:
1058 ;
1059 ;      None
1060 ;
1061 ;--
1062
1063      UNIVERSAL_SYMBOL      SMP$INITIATE_BENIGN
SMP$INITIATE_BENIGN::
1064 ;SMP$INITIATE_BENIGN::
1065      LOCK      LOCKNAME=MEGA      ; Serialize access to benign state
      BLBC      G^SMP$GL_FLAGS,30072$
      PUSHL     RO
      MOVZBL    S^#SPL$C_MEGA,R0
      JSB      G^SMP$ACQUIRE
      POPL     RO
      BRB      30073$
30072$:
      MTPR     S^#IPL$_MEGA,S^#PR$_IPL
30073$:
1066      BBSSI    #SMP$V_BENIGN,G^SMP$GL_FLAGS,20$ ; Enter a benign state
1067 20$:      IPINT_ALL INV_TBA      ; Request all other CPUs to follow
      PUSHL     R2
      MOVL     S^#CPU$V_INV_TBA,R2
      JSB      G^SMP$INTALL_BIT_ACQ
      POPL     R2
1068      RSB
1069
1070      UNIVERSAL_SYMBOL      SMP$TERMINATE_BENIGN
SMP$TERMINATE_BENIGN::
1071 ;SMP$TERMINATE_BENIGN::
1072      BBCCI    #SMP$V_BENIGN,G^SMP$GL_FLAGS,20$ ; Leave benign state
1073 20$:      UNLOCK  LOCKNAME=MEGA      ; Restore blocked threads
      BLBC      G^SMP$GL_FLAGS,30078$
      PUSHL     RO
      MOVZBL    S^#SPL$C_MEGA,R0
      JSB      G^SMP$RELEASE
      POPL     RO

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 28  
X-72 SMP\$INITIATE\_BENIGN - Initiate a benign 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1C

30078\$:

1074 RSB

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 29  
X-72 SMP\$SETUP\_PFORK -- setup for fork to pri 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1

```

1076      .SBTTL SMP$SETUP_PFORK -- setup for fork to primary
1077 ;+
1078 ;
1079 ; These routines provide a mechanism to guarantee that a piece of
1080 ; code executes on the primary CPU at fork IPL. It was designed for
1081 ; use by high-IPL threads which need to write data to the console
1082 ; subsystem during steady-state operation, but there is nothing in-
1083 ; herent in the design which limits it to such a use.
1084 ;
1085 ;SMP$SETUP_PFORK - set up for fork to primary
1086 ;
1087 ; The routine functions as follows: it is entered with a JSB, and
1088 ; takes 2 parameters, the number of bytes of
1089 ; data which the user wishes to pass to the fork process, and the
1090 ; fork lock index or FIPL of the fork process.
1091 ;
1092 ; A block of the appropriate size is allocated from a private pool
1093 ; (preallocated during INIT) FKB$W_SIZE, FKB$B_TYPE, and FKB$B_FLCK
1094 ; are filled in, and R5 set to point to the user data area in the packet.
1095 ; On return to the user, the low bit of R0 indicates whether
1096 ; or not a block was successfully allocated; if so, R5 points to the
1097 ; block.
1098 ;
1099 ; INPUTS:
1100 ;
1101 ;      R3 = length of data (bytes)
1102 ;      R4 = FLCK index or FIPL
1103 ;
1104 ; OUTPUTS:
1105 ;
1106 ;      R0 LBS: block successfully allocated.
1107 ;      R5 = address of user data area within block
1108 ;
1109 ;      R0 LBC: no block of sufficient size available
1110 ;      R5 = 0
1111 ;
1112 ;      All registers preserved.
1113 ;
1114 ;-
1115 ;
1116 ; No block of the required size could be found.  return 0 in R0,r5.
1117 ;
1118 NOBLOCK:
1119      UNLOCK  LOCKNAME=EMB,-          ;Unlock buffer pool database
1120      NEWIPL=(SP)+,-
1121      CONDITION=RESTORE          ; Restore IPL
1122      BLBC    G^SMP$GL_FLAGS,30083$
1123      PUSHL  R0
1124      MOVZBL S^#SPL$C_EMB,R0
1125      JSB    G^SMP$RESTORE
1126      POPL   R0
1127
1128      30083$:
1129      MTPR   (SP)+,S^#PR$_IPL
1130      MOVQ   (SP)+,R1
1131      CLRL   R0          ;Indicate failure
1132      CLRL   R5          ;Indicate failure
1133      RSB

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 30  
X-72 SMP\$SETUP\_PFORK -- setup for fork to pri 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1

```

1126
1127     UNIVERSAL_SYMBOL SMP$SETUP_PFORK
SMP$SETUP_PFORK::
1128 ;SMP$SETUP_PFORK::
1129     MOVQ    R1, -(SP)
1130     ADDL3   #FKB$C_LENGTH, R3, R1    ;Include FKB overhead
1131     ADDL    #4, R1                    ; plus extra LW for real FPC
1132     ADDL    S^#EXE$C_ALCGRNMSK, R1   ;Round size up to next boundry
1133     BICL    S^#EXE$C_ALCGRNMSK, R1   ;Round size back to next boundry
1134     MOVAB   G^SMP$GL_PFORK_POOL, -
1135     R2                                ;Get address of buffer pool listhead
1136     LOCK    LOCKNAME=EMB, -          ;Lock buffer pool database
1137     SAVIPL=-(SP), -                  ;Save current IPL
1138     PRESERVE=NO                       ;Don't preserve R0

     MFPR    S^#PR$IPL, -(SP)
     BLBC    G^SMP$GL_FLAGS, 30088$
     MOVZBL  S^#SPL$C_EMB, R0
     JSB     G^SMP$ACQUIRE
     BRB     30089$

30088$:
     MTPR    S^#IPL$IPL, S^#PR$IPL

30089$:
1139     MOVL   R2, R5                    ;Copy address of first free block address
1140 10$:    MOVL   (R0), R5                ;Save address of previous free block
1141     MOVL   (R0), R5                    ;Get address of next free block
1142     BEQL   NOBLOCK                     ;If eql no memory available
1143 15$:    CMPL   R1, 4(R5)               ;Free block big enough?
1144     BGTRU  10$                          ;If gtru no
1145     BEQL   20$                          ;If eql free block is exact size
1146     ADDL3   R5, R1, R2                  ;Calculate address of new free block
1147     MOVL   (R5)+, (R2)+                 ;Copy link to next free block
1148     SUBL3   R1, (R5), (R2)              ;Calculate size of new free block
1149     MOVAL   -(R2), -(R5)                ;Set link to new free block
1150 20$:    MOVL   (R5), (R0)              ;Copy link to new free block
1151     UNLOCK  LOCKNAME=EMB, -             ;Unlock buffer pool database
1152     PRESERVE=NO, -
1153     NEWIPL=(SP)+, -
1154     CONDITION=RESTORE                   ; Restore IPL

     BLBC    G^SMP$GL_FLAGS, 30094$
     MOVZBL  S^#SPL$C_EMB, R0
     JSB     G^SMP$RESTORE

30094$:
     MTPR    (SP)+, S^#PR$IPL
1155     MOVW   R1, FKB$W_SIZE(R5)          ;Size into 3rd lw
1156     CLRB   FKB$B_TYPE(R5)
1157     MOVB   R4, FKB$B_FLCK(R5)           ;Record FLCK
1158     MOVAB  <FKB$C_LENGTH+4>(R5), -
1159     R5                                ;Set adr of iuser buffer in block
1160     MOVQ   (SP)+, R1
1161     MOVL   s^#1, R0                     ; indicate success
1162     RSB
1163

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 31  
X-72 SMP\$FORK\_TO\_PRIMARY- migrate work packet 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1

```

1165      .SBTTL SMP$FORK_TO_PRIMARY- migrate work packet to primary CPU
1166 ; SMP$FORK_TO_PRIMARY- migrate work packet to primary CPU
1167 ;
1168 ; This routine is called after calling SMP$SETUP_PFORK to allocate and
1169 ; initialize a packet, and after moving any caller-dependent data into the
1170 ; packet.
1171 ;
1172 ; The call to the user's fork routine should be assumed to be asynchronous
1173 ; to the final RSB from SMP$FORK_TO_PRIMARY (although it may not be). The
1174 ; user's fork process will be called back on the primary CPU at the
1175 ; specified fork IPL AS IF it were called from the fork dispatcher
1176 ; (NB: actual dispatch mechanism will vary based on the caller's original
1177 ; IPL and CPU). In particular, the specified fork lock will be held, R0-R2
1178 ; will be scratch, and R3 and R4 will contain whatever they did on entry to
1179 ; SMP$FORK_TO_PRIMARY. The one difference is that R5 will point to the user
1180 ; data area within the packet, not to the beginning of the packet.
1181 ;
1182 ; When complete, the user's fork process should RSB. The packet will then be
1183 ; be deallocated.
1184 ;
1185 ;
1186 ; INPUTS:
1187 ;
1188 ;     R5 = address of user buffer area in packet allocated and initial-
1189 ;         ized by SMP$SETUP_PFORK.
1190 ;     0(SP) = User's FPC
1191 ;     4(SP) + address of caller's caller (i.e., return address
1192 ;         for SMP$FORK_TO_PRIMARY
1193 ;
1194 ; OUTPUT: Fork process created and queued to primary.
1195 ;     R5 destroyed.
1196 ;
1197 USER_FPC = 12 ; stack offset of user's FPC
1198
1199      UNIVERSAL SYMBOL SMP$FORK_TO_PRIMARY
SMP$FORK_TO_PRIMARY::
1200
1201 ;SMP$FORK_TO_PRIMARY::
1202      MOVQ    R0, -(SP)
1203      MOVL   R2, -(SP)
1204      SUBL   #<FKB$C_LENGTH+4>, R5 ; back off pointer to start of packet
1205 ;
1206 ; the first divergence is based on whether the request is being made from
1207 ; the primary CPU. Make that determination.
1208 ;
1209      FIND_CPU_DATA   R0 ; get this CPU's DB addr
MFPR    S^#PR$ISP, R0
BICL2  G^SMP$GL_BASE_MSK, R0
1210      CMPL   CPU$L_PHY_CPUID(R0), -
1211          g^SMP$GL_PRIMID ; compare to primary
1212      BEQL   35$ ; primary
1213      BRW    200$ ; branch if secondary
1214 ;
1215 ; primary -- the next divergence is based on IPL and whether we must fork
1216 ;
1217 35$:      MOVZBL FKB$B_FLCK(R5), R1 ;GET FORK lock/FIPL
1218

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 32  
X-72 SMP\$FORK\_TO\_PRIMARY- migrate work packet 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1

```

1219 ; The following test of spinlock index VS. IPL is checked by an
1220 ; ASSUME done in LDAT. Spinlock indices are in the range 20 to 3F (hex).
1221
1222         BBC      #5,R1,40$                ;Br if direct IPL
1223         MOVL     G^SMP$AL_IPLVEC[R1],R1  ;Get FORK IPL from spinlock database
1224 40$:    SAVIPL   R0                        ; and current IPL
1225         MFPR     S^#PR$_IPL,R0
1226         CML     R0,R1                      ; compare current to FIPL
1227         BGTR     100$                      ; BR if too high
1228 ;
1229 ; primary, at or below FIPL -- emulate fork dispatcher. registers
1230 ; available to fork process are R0,R1,R2.
1231 ; R3,R4 have the same contents as at the time of the fork request.
1232 ; Fork lock is held.
1233         FORKLOCK LOCK=FKB$B_FLCK(R5),-    ; get fork lock
1234         FIPL=YES,-                          ; check for degenerate case of IPL-only
1235         SAVIPL=-(SP),-                      ; save IPL
1236         PRESERVE=NO                          ; don't save R0
1237         MFPR     S^#PR$_IPL,-(SP)
1238         MOVZBL   FKB$B_FLCK(R5),R0
1239         BBC      #5,R0,30097$
1240         BLBC     G^SMP$GL_FLAGS,30098$
1241         JSB      G^SMP$ACQUIRE
1242         BRB      30099$
1243
30097$:    MTPR     R0,S^#PR$_IPL
1244         BRB      30098$
30098$:    MTPR     G^SMP$AL_IPLVEC[R0],S^#PR$_IPL
1245
30099$:
1246         ADDL     #<FKB$C_LENGTH+4>,R5      ; adjust R5 to point to user data
1247         JSB      @<USER_FPC+4>(SP)         ; call user's routine through location
1248         SUBL     #<FKB$C_LENGTH+4>,R5      ; restore R5
1249         BRB      30100$                    ; saved on stack.
1250
1251 ; R5 assumed to still point to block
1252 ;
1253         FORKUNLOCK LOCK=FKB$B_FLCK(R5),-
1254         NEWIPL=(SP)+,-
1255         PRESERVE=NO
1256         BLBC     G^SMP$GL_FLAGS,30100$
1257         MOVZBL   FKB$B_FLCK(R5),R0
1258         JSB      G^SMP$RELEASE
1259
30100$:    MTPR     (SP)+,S^#PR$_IPL
1260         BSBW     DEALLOCATE
1261
1262 90$:     MOVL     (SP)+,R2
1263         MOVQ     (SP)+,R0
1264         ADDL     #4,SP                      ; back up over user FPC
1265         RSB
1266
1267 ;
1268 ; primary -- IPL requires forking
1269 ;
1270 100$:    MOVL     USER_FPC(SP),-

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 33  
X-72 SMP\$FORK\_TO\_PRIMARY- migrate work packet 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1

```

1258          FKB$C_LENGTH(R5); save user's fork address
1259      PUSHAB 90$
1260      MOVAB  W^FORK_DISPATCH,-(SP)    ; fork PC
1261      JMP     G^EXE$FORK                ; create fork process
1262 ;
1263 ; secondary -- queue work packet to primary
1264 ;
1265 200$:    MOVAB  w^FORK_DISPATCH,-      ; save fork PC
1266          FKB$C_LENGTH(R5)
1267      MOVL   USER_FPC(SP),-
1268          FKB$C_LENGTH(R5)              ; caller's REAL FPC
1269      MOVQ   R3,FKB$C_LENGTH(R5)        ; save registers
1270      MOVL   G^SMP$G_L_PRIMID,R0
1271      MOVL   G^SMP$G_L_CPU_DATA[R0],R0 ; Get address of CPU database
1272      $INSQTI (R5),CPU$Q_WORK_FOFL(R0),R1 ; Insert work pkt on end of queue
          CLRL  R1
30101$:
          INSQTI (R5),CPU$Q_WORK_FOFL(R0)
          BCC   30102$
          AOBLSS #900000,R1,30101$
          .WORD  ^XFEFF
          .IIF IDN <FATAL>,<FATAL> , .WORD      BUG$_BADQHDR!4
30102$:
30103$:
1273      BBSSI  #CPU$V_WORK_FQP,CPU$L_WORK_REQ(R0),220$ ; Set work request bit
1274 220$:
1275      MOVL   CPU$L_PHY_CPUID(R0),R0    ; Get physical CPU ID
1276      IPINT_CPU      ; Generate IP Interrupt
          JSB   G^SMP$INTPROC
1277 250$:    MOVQ   (SP)+,R0
1278          MOVL   (SP)+,R2
1279          ADDL   #4,SP                  ; back up over user FPC
1280          RSB
1281
1282 ;+
1283 ; FORK_DISPATCH:
1284 ;     dispatch the user's fork routine.
1285 ;     INPUTS:
1286 ;     R3 = whatever the caller specified
1287 ;     R4 = ditto
1288 ;     R5 = address of FKB
1289 ;     FKB$C_LENGTH(R5) = caller's REAL FPC
1290 ;     <FKB$C_LENGTH+4>(R5) = user's data
1291 ;     Fork lock held
1292 ;     R0-R2 scratch
1293 ;-
1294 FORK_DISPATCH:
1295      PUSHL   FKB$C_LENGTH(R5)          ; push user's FPC
1296      ADDL   #<FKB$C_LENGTH+4>,R5      ; point R5 to user's buffer
1297      JSB   @(SP)+                      ; dispatch user's fork process
1298      SUBL   #<FKB$C_LENGTH+4>,R5      ; point R5 to start of packet
1299      BSBW  DEALLOCATE                  ; return packet to pool
1300      RSB                                ; return to fork dispatcher
1301
1302 ;+
1303 ; DEALLOCATE
1304 ;

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 34  
X-72 SMP\$FORK\_TO\_PRIMARY- migrate work packet 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1

```

1305 ; INPUT:
1306 ;
1307 ;           R5 = ADDRESS OF BLOCK TO BE DEALLOCATED
1308 ;           FKB$W_SIZE(R5) = SIZE OF BLOCK
1309 ;
1310 ; OUTPUTS:
1311 ;
1312 ;           THE SPECIFIED BLOCK IS RETURNED
1313 ;           R5 destroyed
1314 ;-
1315 DEALLOCATE:
1316     MOVZWL  FKB$W_SIZE(R5),R1           ; Get size of block in bytes
1317     LOCK    LOCKNAME=EMB,-             ; Lock buffer pool database
1318     PRESERVE=NO,-
1319     SAVIPL=-(SP)                       ; Save current IPL
        MFPR  S^#PR$_IPL,-(SP)
        BLBC  G^SMP$GL_FLAGS,30107$
        MOVZBL S^#SPL$_EMB,R0
        JSB   G^SMP$ACQUIRE
        BRB   30108$
30107$:
        MTPR  S^#IPL$_EMB,S^#PR$_IPL
30108$:
1320     MOVAB  G^SMP$GL_PFORK_POOL,-
1321     R0                                           ; Get address of listhead
1322 10$:     MOVL  R0,R2                           ;SAVE ADDRESS OF PREVIOUS FREE BLOCK
1323     MOVL  (R2),R0                             ;GET ADDRESS OF NEXT FREE BLOCK
1324     BEQL  20$                                  ;IF EQL END OF LIST
1325 15$:     CMPL  R5,R0                          ;BLOCK LOGICALLY GO HERE?
1326     BGTRU 10$                                  ;IF GTRU NO
1327 20$:     MOVL  R0,(R5)                        ;ASSUME NO AGGLOMERATION
1328     BEQL  30$                                  ;END OF LIST - NO AGGLOMERATION
1329     ADDL3  R5,R1,R4                            ;CALCULATE ADDRESS OF END OF BLOCK
1330     CMPL  R0,R4                               ;END OF BLOCK EQUAL TO NEXT IN LIST?
1331     BGTRU 30$                                  ;IF GTR DO NOT AGGLOMERATE
1332     MOVL  (R0)+,(R5)                          ;MOVE LINK TO BLOCK BEING RELEASED
1333     ADDL  (R0),R1                             ;ACCUMULATE LENGTH OF NEW FREE BLOCK
1334 30$:     MOVL  R2,R4                          ;CALCULATE ENDING ADDRESS OF PREVIOUS BLOCK
1335     MOVL  R5,(R2)+                            ;ASSUME NO AGGLOMERATION
1336     ADDL  (R2),R4                             ;ADD LENGTH TO BLOCK BASE ADDRESS
1337     CMPL  R5,R4                               ;END ADDRESS EQUAL TO BLOCK BEING RELEASED?
1338     BGTRU 40$                                  ;IF GTR DO NOT AGGLOMERATE
1339     ADDL  (R2),R1                             ;ACCUMULATE SIZE OF NEW FREE BLOCK
1340     MOVL  (R5),-(R2)                          ;MOVE LINK TO PREVIOUS FREE BLOCK
1341     MOVL  R2,R5                               ;SET ADDRESS OF NEW FREE BLOCK
1342 40$:     MOVL  R1,4(R5)                      ;SET SIZE OF FREE BLOCK
1343 50$:     UNLOCK LOCKNAME=EMB,-                ; Unlock buffer pool database
1344     PRESERVE=NO,-
1345     NEWIPL=(SP)+,-
1346     CONDITION=RESTORE                       ; Restore IPL
        BLBC  G^SMP$GL_FLAGS,30113$
        MOVZBL S^#SPL$_EMB,R0
        JSB   G^SMP$RESTORE
30113$:
        MTPR  (SP)+,S^#PR$_IPL
1347     CLRL  R5                                 ;ZAP block address
1348     RSB

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 35  
X-72 SMP\$FORK\_TO\_PRIMARY- migrate work packet 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1

1349

1350

.SBTTL MESSAGE Display message on virtual console

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 36  
X-72 MESSAGE Display message on virtual conso 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1

```

1352 ;++
1353 ;
1354 ; This code displays messages for this CPU on the console terminal. The
1355 ; format of the messages is structured such that there is a primary
1356 ; message to be displayed first, then this routine will display this
1357 ; processor's ID in ASCII, then a secondary message is displayed.
1358 ;
1359 ; Inputs:
1360 ;
1361 ;     R6 = Address of per-CPU database
1362 ;
1363 ;     0(SP) = Return address
1364 ;     4(SP) = Address of primary ASCII text to display
1365 ;     8(SP) = Address of secondary ASCII text to display after CPU number
1366 ;
1367 ;--
1368 FIRST_PART = 24 ; Stack offset of address of first half of m
1369 SECOND_PART = 28 ; Stack offset of address of second hal of m
1370 ADDTL_REGS = 8 ; Adjustment for addtl saved registers
1371 MESSAGE: ;
1372     PUSHR    #^M<R1,R2,R3,R4,R5>
1373     MOVZBL   @FIRST_PART(SP),R3 ; length of first part of msg
1374     ADDB     @SECOND_PART(SP),R3 ; length of second part, plus CPU ID
1375     ADDB     #2,R3
1376     MOVL     g^OPASAR_UCB0,R4 ; OPAO UCB addr
1377     MOVZBL   UCB$B_FLCK(R4),-
1378             R4 ; Use OPAO FLCK
1379     JSB     g^SMP$SETUP_PFORK ; allocate and init packet
1380     BLBS    R0,20$ ; continue if allocation succeeded
1381     POPR    #^M<R1,R2,R3,R4,R5>
1382     RSB
1383 ;
1384 ; R5 now points to data area in packet
1385 ;
1386 20$:     PUSHR    #^M<R3,R5> ; save around MOVC
1387     MOVL     <FIRST_PART+ADDTL_REGS>(SP),-
1388             R1 ; addr of first part of msg
1389     MOVZBL   (R1)+,R2 ; length of string in R2, bump address
1390     MOVC3    R2,(R1),(R5) ; move data to packet.
1391             ; NB: Registers saved above
1392     MOVZBL   CPU$L_PHY_CPUID(R6),R0 ; Get this CPU's physical ID
1393 ;
1394 ; NB MOVC3 leaves R3 pointing to the next byte in the output buffer
1395 ;
1396     MOVW     W^CH_TABLE[R0],(R3)+ ; ASCII CPU ID
1397     MOVL     <SECOND_PART+ADDTL_REGS>(SP),-
1398             R1 ; addr of second part of msg
1399     MOVZBL   (R1)+,R2 ; length of string in R3, bump address
1400     MOVC3    R2,(R1),(R3) ; move data to packet.
1401     POPR    #^M<R3,R5>
1402     PUSHAB   30$ ; simulate 'caller's caller'
1403     PUSHAB   G^SMP$WRITE_OPAO ; fork routine address
1404     JMP     G^SMP$FORK_TO_PRIMARY ; dispatch packet
1405
1406 30$:     POPR    #^M<R1,R2,R3,R4,R5> ; SMP$FORK_TO_PRIMARY will return here
1407     RSB
1408

```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 37  
X-72 SMP\$WRITE\_OPA0 - fork routine to broadca 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1

```
1410      .SBTTL  SMP$WRITE_OPA0 - fork routine to broadcast message to console
1411 ;++
1412 ; FUNTIONAL DESCRIPTION:
1413 ;
1414 ;SMP$WRITE_OPA0 - fork routine to broadcast message to console
1415 ;
1416 ; ENVIRONMENT:
1417 ;
1418 ;
1419 ; CALLING SEQUENCE:
1420 ;
1421 ;       JSB      G^SMP$WRITE_OPA0
1422 ;
1423 ; INPUT PARAMETERS:
1424 ;
1425 ;       r3 = length of data
1426 ;       r5 = address of data
1427 ;       R0,r1,r2 scratch
1428 ;       fork lock held.
1429 ;
1430 ; OUTPUT PARAMETERS:
1431 ;
1432 ;       None
1433 ;
1434 ;--
1435
1436      UNIVERSAL_SYMBOL      SMP$WRITE_OPA0
SMP$WRITE_OPA0::
1437 ;SMP$WRITE_OPA0::
1438 ;
1439      MOVL      R3,R1          ; length of data
1440      MOVL      R5,R2          ; address of data to output
1441      MOVL      G^OPASAR_UCB0,R5 ; target device's UCB
1442      JSB      G^IOC$BROADCAST  ; broadcast to OPA0
1443      MOVL      R2,R5          ; restore register
1444      ; NB: this assumes that
1445      ; IOC$BROADCAST preserves
1446      ; all registers, R2 in particular
1447      RSB
1448
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 38  
X-72 SYNCH\$INIT\_ONCE - Once only initialization 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1

```

1450      .SBTTL  SYNCH$INIT_ONCE - Once only initialization for Spinlocks
1451 ;++
1452 ; FUNTIONAL DESCRIPTION:
1453 ;
1454 ; SYNCH$INIT_ONCE - Once only initialization for Spinlocks
1455 ;
1456 ; CALLING SEQUENCE:
1457 ;
1458 ;     JSB      G^SYNCH$INIT_ONCE
1459 ;
1460 ; INPUT PARAMETERS:
1461 ;
1462 ;     R5      =      Pointer to initialization routine flags
1463 ;
1464 ; OUTPUT PARAMETERS:
1465 ;
1466 ;     R0      =      Status of routine
1467 ;
1468 ;--
1469
1470
1471      INITIALIZATION ROUTINE -
1472      SYNCH$INIT_ONCE
1473
1474      .SAVE
1475      .PSECT  EXEC$INIT_001, LONG, RD, WRT, EXE, PIC
1476      .LONG  SYNCH$INIT_ONCE - .
1477      .LONG  ...FLAGS...
1478      .RESTORE
1479
1480      DECLARE_PSECT  EXEC$INIT_CODE
1481      .PSECT  EXEC$INIT_CODE, LONG, RD, WRT, EXE, PIC
1482
1483      SYNCH$INIT_ONCE::
1484
1485      BBS      #BOOSTATE$V_MAPPED, -      ; Br if mapped
1486      G^EXE$G_L_STATE, 10$
1487      BICL    #INIRTN$M_NO_RECALL, (R5)   ; We need to be called again
1488      BRW     90$
1489      10$:
1490      MOVAB   W^SMP$G_L_SPNLKVEC, -      ; Init spinlock vector address
1491      G^SMP$G_L_SPNLKVEC
1492      MOVW    #SPL$ _NUM_LOCKS, -      ; Init count of locks in vector
1493      G^SMP$G_L_SPNLKCNT
1494      MOVW    #SPL$ _MIN_INDEX, -      ; Init first spinlock index
1495      G^SMP$G_L_MIN_INDEX
1496      ;
1497      ; initialize spinwait timeout interval. Certain IPL 8 locks (e.g.,
1498      ; SCHED, MMG) have long holding times; all spinlocks at IPL <= 8
1499      ; therefore have their timeout intervals set to SMP_LNGSPINWAIT
1500      ; to prevent SPINWAIT timeouts in cases of nested acquisition.
1501      ; All locks at IPLs greater than 8 have their timeout initialized to
1502      ; SMP_SPINWAIT.
1503      ;
1504      MOVZWL  #SPL$ _MIN_INDEX, R0      ; starting SPNLKVEC index
1505      20$:
1506      MOVL    W^SMP$G_L_SPNLKVEC[R0], R2 ; address of spinlock
1507      BGEQ    40$
1508      ; empty slot if not S0 address
1509      ASSUME  IPL$ _SCS EQ 8

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 39  
X-72 SYNCH\$INIT\_ONCE - Once only initializati 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1

```

1501      CMPL      SPL$B_IPL(R2),#IPL$SCS          ; locks at IPL <= 8 get long
1502      BGTR      30$                             ; timeout interval
1503      MOVL      G^SGN$GL_SMP_LNGSPINWAIT,-
1504      SPL$L_TIMO_INT(R2)                         ; set up LONG timeout interval
1505      BRB       40$
1506 30$:  MOVL      G^SGN$GL_SMP_SPINWAIT,-
1507      SPL$L_TIMO_INT(R2)                         ; set up timeout interval
1508 40$:  AOBLSS   #SPL$_MAX_INDEX,R0,20$         ; loop until done.
1509
1510      ; Init NULL UCB device lock address
1511
1512      MOVL      G^NL$AR_UCB0,R2                   ; Get address of NULL UCB
1513      MOVAB     G^SMP$GL_MAILBOX,UCB$L_DLCK(R2) ; Set DEVICE LOCK ADDRESS
1514
1515      ; Init SYSTEM CRB device lock address
1516
1517      MOVL      UCB$L_CRB(R2),R2                   ; Get SYSTEM CRB ADDRESS
1518      MOVAB     G^SMP$GL_MAILBOX,CRB$L_DLCK(R2) ; Set DEVICE LOCK ADDRESS
1519
1520      ; Init MAILBOX TEMPLATE UCB device lock address
1521
1522      MOVL      G^MB$AR_UCB0,R2                   ; Get MAILBOX UCB address
1523      MOVAB     G^SMP$GL_MAILBOX,UCB$L_DLCK(R2) ; Set DEVICE LOCK ADDRESS
1524
1525      ; Init JOB CONTROLLER MAILBOX UCB device lock address
1526
1527      MOVL      G^MB$AR_UCB1,R2                   ; Get JBC UCB address
1528      MOVAB     G^SMP$GL_MAILBOX,UCB$L_DLCK(R2) ; Set DEVICE LOCK ADDRESS
1529
1530      ; Init OPERATOR MAILBOX device lock address
1531
1532      MOVL      G^MB$AR_UCB2,R2                   ; Get OPERATOR MAILBOX UCB address
1533      MOVAB     G^SMP$GL_MAILBOX,UCB$L_DLCK(R2) ; Set DEVICE LOCK ADDRESS
1534
1535      ; Init AUDIT_SERVER MAILBOX device lock address
1536
1537      MOVL      G^MB$AR_UCB3,R2                   ; Get AUDIT_SERVER MAILBOX UCB address
1538      MOVAB     G^SMP$GL_MAILBOX,UCB$L_DLCK(R2) ; Set DEVICE LOCK ADDRESS
1539
1540      ; init timeout for OPA0 device lock
1541
1542      MOVL      G^OPA$AR_SPL,R2                   ; pick up spinlock address
1543      MOVL      G^SGN$GL_SMP_SPINWAIT,-
1544      SPL$L_TIMO_INT(R2)                         ; set spinlock interval
1545      ;
1546      ; Copy IPL vector from LDAT to base image
1547      ;
1548      PUSHR     #^M<R3,R4,R5>                     ; Save registers
1549      MOVCC     #<4*SPL$_NUM_LOCKS>,W^SMP$GL_IPL_VEC+<4*SPL$_MIN_INDEX>,-
1550      G^SMP$AL_IPLVEC+<4*SPL$_MIN_INDEX> ; Copy LDAT vector
1551      POPR      #^M<R3,R4,R5>                     ; Restore registers
1552      ;
1553      ; check to see if we can allocate free pages yet...
1554      ;
1555      BBS       #BOOSTATE$V_PFN_INIT,-           ; Br if can do so
1556      G^EXE$GL_STATE,50$
1557      BICL     #INIRTN$M_NO_RECALL,(R5)         ; We need to be called again

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SMPROUT - SMP routines for VMS 10-MAY-1989 16:58:12 VAX MACRO V5.0-8 Page 40  
X-72 SYNCH\$INIT\_ONCE - Once only initializati 19-APR-1989 15:24:09 [SYS.SRC]SMPROUT.MAR;1 (1

```

1558      BRW      90$
1559 ;
1560 ; initialize the buffer pool used by SMP$FORK_TO_PRIMARY
1561 ;
1562 50$:  MOVZBL  G^SMP$GB_PFORK_POOL_SIZE,-
1563      R2                ; This many pages are needed.
1564      JSB      G^LDR$ALLOC_PT
1565      BLBC     R0,100$    ; Failure here
1566      SUBL3    G^LDR$GL_SPTBASE,R1,R3 ; Calculate base address
1567      ASHL     #<9-2>,R3,R3 ; and initialize buffer pointer.
1568      BISL3    #^X80000000,R3,-(SP)
1569
1570 80$:  MOVQ     R1,-(SP)    ; Preserve through following call
1571      JSB      G^MMG$ALLOCPFN ; Get one PFN.
1572      MOVQ     (SP)+,R1    ; Restore loop indices.
1573      TSTL     R0          ; Was one available?
1574      BLEQ     100$        ; LEQ: No!
1575      MOVL     R0,(R1)     ; Map page.
1576      BISL2    #<PTE$M_VALID!PTE$C_ERKW!PTE$C_KOWN>,(R1)+ ; Make page valid.
1577      SOGTR    R2,80$
1578 ;
1579 ; save pool address and initialize listhead. Listhead contains a FLINK and
1580 ; a blocksize as the first 2 LW of each block.
1581 ;
1582      MOVL     (SP)+,R3
1583      MOVL     R3,G^SMP$GL_PFORK_POOL
1584      CLRL     (R3)+        ; Zero FLINK (empty list)
1585      MOVZBL  G^SMP$GB_PFORK_POOL_SIZE,-(SP)
1586      MULL3    (SP)+,#512,(R3) ; and set size
1587
1588 90$:  MOVL     #SS$ _NORMAL,R0
1589 100$:  RSB                ; Return
1590
1591
1592      .end

```

## 12 SPINLOCKS.LIS

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 0  
Table of contents

|      |      |  |
|------|------|--|
| (1)  | 45   | HISTORY  |
| (2)  | 297  | DECLARATIONS   |
| (3)  | 312  | SPINWAIT - Spinlock wait timeout macro                         |
| (4)  | 403  | ACQUIRE - Acquire a spinlock through busy waiting              |
| (7)  | 993  | RESTORE - Conditionally release a spinlock                     |
| (10) | 1168 | RELEASE - Release a spinlock                                   |
| (12) | 1375 | REI_CHECK - Check spinlock database consistency                |
| (13) | 1455 | NOLOCKS - Make sure no spinlocks are held                      |
| (14) | 1512 | SMP\$CHKLOCK - Make sure spinlock is owned before proceeding   |
| (15) | 1588 | ALLOC_SPL - Allocate a spinlock                                |
| (16) | 1639 | INIT_SPL - Initialize a spinlock                               |
| (17) | 1692 | ADJUST_IPL - ADJUST THE IPL OF A LOCK, ALTER RANKS ACCORDINGLY |



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:3/ VAX MACRO V5.0-8 Page 1  
X-86 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (1)

```
1      .NLIST CND
8      .TITLE      SPINLOCKS - Spinlock routines for VMS/SMP
11
12     .IDENT 'X-86'
13 ;
14 ;*****
15 ;*
16 ;*  COPYRIGHT (c) 1988 BY
17 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
18 ;*  ALL RIGHTS RESERVED.
19 ;*
20 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
21 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
22 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
23 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
24 ;*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
25 ;*  TRANSFERRED.
26 ;*
27 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
28 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
29 ;*  CORPORATION.
30 ;*
31 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
32 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
33 ;*
34 ;*
35 ;*****
36 ;
37 ;++
38 ; Facility:      Symmetric Multiprocessing
39 ;
40 ; Abstract:      The subroutines to implement the acquire and release
41 ;                synchronization spinlocks for symmetric multiprocessing.
42 ;
43 ; Environment:  Kernel Mode.
44 ;
45     .SBTTL HISTORY
46 ;
47 ; Author: Rod Gamache, Bill Laing, Mike Harvey  Creation Date: 17-May-1985
48 ;
49 ; Modified by:
50 ;
51 ; X-86  MSH0344      Michael S. Harvey      29-Mar-1988
52 ;                Close window where spinlock acquire/release operations
53 ;                could cause deadlocks should an interprocessor interrupt
54 ;                occur while a CPU has set the spinlock access bit.
55 ;
56 ;                Also, modify SPINWAIT macro to utilize the per-CPU
57 ;                timer cells, rather than the obsolete EXE$xxx versions
58 ;                of the timer cells.
59 ;
60 ; X-85  MSH0333      Michael S. Harvey      18-Nov-1987
61 ;                Recognize BENIGN STATE and VIRTCONS requests in the
62 ;                spinwait code for CPUs that are in the OVERRIDE set
63 ;                and at high IPL. This prevents system deadlocks caused
64 ;                when these conditions are ignored at this level.
65 ;
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 2  
X-86 HISTORY 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (1)

66 ; X-84 MSH0327 Michael S. Harvey 11-Nov-1987  
67 ; OVERRIDE check in spinwait macro is too broad. This  
68 ; currently leads to deadlocks over IP IPL traffic.  
69 ;  
70 ; X-83 RNG5083 Rod N. Gamache 28-Oct-1987  
71 ; Allow members of OVERRIDE set to raise IPL if needed in  
72 ; the minimum version.  
73 ;  
74 ; X-82 MSH0328 Michael S. Harvey 26-Oct-1987  
75 ; Prevent lowering of IPL on ACQUIRE spinwait path if  
76 ; CPU is a member of the OVERRIDE set and has called  
77 ; ACQUIRE at an IPL lower than that of the desired lock.  
78 ;  
79 ; X-81 MSH0327 Michael S. Harvey 23-Oct-1987  
80 ; Spinwait code must accomodate the case when it executes  
81 ; at an IPL that blocks ACTIVE set driven IP interrupt  
82 ; requests. Such requests MUST be acknowledged in the  
83 ; spinwait code, else the system will deadlock and crash.  
84 ;  
85 ; X-80 RNG5080 Rod N. Gamache 11-Sep-1987  
86 ; Fix previous edit - we must check for BUG\_DONE in order  
87 ; to allow the primary CPU to 'break' spinlocks.  
88 ; Get free check of 'invalid' release of a spinlock when it  
89 ; is not owned in the minimum version of release.  
90 ; Initialize SPINLOCK size when spinlock is allocated.  
91 ;  
92 ; X-78 RNG5078 Rod N. Gamache 8-Jul-1987  
93 ; Remove unnecessary checks in SMP minimum version of release  
94 ; code path.  
95 ;  
96 ; X-77 RNG5077 Rod N. Gamache 15-Jun-1987  
97 ; Don't record release PC in minimum version of SPINLOCKS.  
98 ;  
99 ; X-76 MSH0313 Michael S. Harvey 15-May-1987  
100 ; Record release PC when multiple acquisitions are being  
101 ; wiped out.  
102 ;  
103 ; Also, don't charge process for busywait time in  
104 ; minimal version of spinlock acquire (already done  
105 ; for full-blown version).  
106 ;  
107 ; X-75 SJF Stu Farnham 4-May-1987  
108 ; Use SMP\_LNGSPINWAIT for timeout interval for spinlocks  
109 ; held at IPLs <= 8 to compensate for the additive effects  
110 ; of the locks at that IPL with long holding times.  
111 ;  
112 ; X-74 RNG5074 Rod Gamache 4-Apr-1987  
113 ; Fix problem with ACQUIRE paths: make sure IPL is GTR 2  
114 ; when using FIND\_CPU\_DATA macro.  
115 ;  
116 ; X-73 RNG5073 Rod Gamache 24-Mar-1987  
117 ; Change all refs from SMP\$AR\_SPNLKVEC to SMP\$GL\_SPNLKVEC.  
118 ; Change refs from SMP\$GL\_SPNLKCNT to SMP\$GW\_SPNLKCNT  
119 ; Check override membership before setting IPLs on uniprocessor.  
120 ;  
121 ; X-72 SJF Stu Farnham 16-Mar-1987  
122 ; Add a check for spinlock starvation to SPINWAIT macro.

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 3  
X-86 HISTORY 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (1)

123 ;                   This occurs when contention for a lock is high, and  
124 ;                   ownership keeps changing. Under those conditions, a CPU  
125 ;                   may time out not because a lock is being held too long,  
126 ;                   but because that CPU may keep losing the race to acquire  
127 ;                   the lock. In such a case a new timeout quantum should be  
128 ;                   given instead of taking a CPUSPINWAIT bugcheck.  
129 ;  
130 ;           X-71    MJW0113           Michael J. Worcester    26-Feb-1987  
131 ;                   Change reference of DYN\$C\_SPL\_DEVICELOCK to  
132 ;                   SPL\$C\_SPL\_DEVICELOCK.  
133 ;  
134 ;           X-70    SJF                Stu Farnham            16-Feb-1987  
135 ;                   Reverse X-66 in the interests of efficiency. Parameter  
136 ;                   now expressed in the units of the calculation (10 usec)  
137 ;  
138 ;           X-68,69 MSH0246           Michael S. Harvey        10-Feb-1987  
139 ;                   Reduce spinlock counter magnitude. Remove obsolete ASSUME.  
140 ;  
141 ;           X-67    CEG002            Clair Grant             6-Feb-1987  
142 ;                   Fix branch destination out of range  
143 ;  
144 ;           X-66    SJF                Stu Farnham            5-Feb-1987  
145 ;                   Convert milli- to micro- seconds in SPINWAIT macro  
146 ;  
147 ;           X-65    WCT0025           Ward C. Travis         28-Jan-1987  
148 ;                   Handle zero and other non-S0 addresses in the  
149 ;                   spinlock address vector.  
150 ;  
151 ;           X-64    RNG5064           Rod Gamache            27-Jan-1987  
152 ;                   Add minimized version of spinlock acquisition.  
153 ;  
154 ;           X-63    SJF                Stu Farnham            23-Dec-1986  
155 ;                   Initialize SPL\$L\_TIMO\_INT in SMP\$INIT\_SPL.  
156 ;                   Add spinwait timeouts.  
157 ;  
158 ;           X-62    MSH0280           Michael S. Harvey        19-Dec-1986  
159 ;                   Break up SPLIPLERR into more recognizable and separate  
160 ;                   error conditions, SPLIPLLOW and SPLIPLHIGH.  
161 ;  
162 ;           X-61    MSH0291           Michael S. Harvey        11-Dec-1986  
163 ;                   Allow high-IPL busywaits to see bugcheck signals. This  
164 ;                   mechanism predates an effective spinlock timeout mechanism.  
165 ;  
166 ;           X-60    MSH0287           Michael S. Harvey        1-Dec-1986  
167 ;                   Prevent IPL bouncing between Qbus devicelock IPLs and bus  
168 ;                   grant IPL that can occur on the busywait. This one was  
169 ;                   found by a technical writer! Good thing they're around to  
170 ;                   keep the engineers honest, eh!  
171 ;  
172 ;           X-59    MSH0284           Michael S. Harvey        23-Oct-1986  
173 ;                   Recognize the OVERRIDE set. Allow members of the OVERRIDE  
174 ;                   set to acquire/release spinlocks without regard to the  
175 ;                   spinlock's associated IPL.  
176 ;  
177 ;           X-58    RNG0058           Rod Gamache            29-Oct-1986  
178 ;                   Change IPL\$ \_SCHED to IPL\$ \_RESCHED.  
179 ;

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 4  
X-86 HISTORY 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (1)

|       |      |         |   |             |
|-------|------|---------|---|-------------|
| 180 ; | X-57 | RNG0057 | Rod Gamache   | 6-Oct-1986  |
| 181 ; |      |         | Allow CHKLOCK to continue if called when INIT is running.     |             |
| 182 ; |      |         |   |             |
| 183 ; | X-56 | MSH0280 | Michael S. Harvey   | 24-Sep-1986 |
| 184 ; |      |         | Allow bugcheck to effectively break held locks.               |             |
| 185 ; |      |         | Add SPLRSTERR to more readily distinguish the different       |             |
| 186 ; |      |         | spinlock release failures.                                    |             |
| 187 ; |      |         |   |             |
| 188 ; | X-54 | RNG0054 | Rod Gamache   | 18-Sep-1986 |
| 189 ; |      |         | Allow CHK_LOCK to be called when IPL is 31.                   |             |
| 190 ; |      |         |   |             |
| 191 ; | X-48 | WCT     | Ward C. Travis  | 11-Sep-1986 |
| 192 ; |      |         | ADJUST_IPL deloopified: needed to bug out if a lock's         |             |
| 193 ; |      |         | rank was requested to change.                                 |             |
| 194 ; |      |         |   |             |
| 195 ; | X-47 | RNG0047 | Rod N. Gamache  | 10-Sep-1986 |
| 196 ; |      |         | Fix incorrect order of CMP arguments in ADJUST_IPL routine.   |             |
| 197 ; |      |         |   |             |
| 198 ; | X-46 | RNG0046 | Rod N. Gamache  | 8-Sep-1986  |
| 199 ; |      |         | Fix typos from previous edit.                                 |             |
| 200 ; |      |         |   |             |
| 201 ; | X-45 | WCT     | Ward C. Travis  | 22-Aug-1986 |
| 202 ; |      |         | Add SMP\$ADJUST_IPL to allow boot-time reordering of          |             |
| 203 ; |      |         | SPL IPLs.   |             |
| 204 ; |      |         |   |             |
| 205 ; | X-44 | MSH0262 | Michael S. Harvey   | 16-Jul-1986 |
| 206 ; |      |         | Add DEVICELOCK path which doesn't set the IPL on acquisition. |             |
| 207 ; |      |         | Also, waive IPL violation on acquisition of DEVICELOCKS       |             |
| 208 ; |      |         | if the lock in question is already owned by the acquiring     |             |
| 209 ; |      |         | processor.  |             |
| 210 ; |      |         |   |             |
| 211 ; | X-43 | RNG0043 | Rod N. Gamache  | 14-Jul-1986 |
| 212 ; |      |         | Record PC of callers of SMP\$RESTORE.                         |             |
| 213 ; |      |         |   |             |
| 214 ; | X-42 | MSH0261 | Michael S. Harvey   | 14-Jul-1986 |
| 215 ; |      |         | Add some robustness on spinlock validation.                   |             |
| 216 ; |      |         |   |             |
| 217 ; | X-40 | MSH0259 | Michael S. Harvey   | 11-Jul-1986 |
| 218 ; |      |         | Extend the IPL array in the per-CPU database to longwords.    |             |
| 219 ; |      |         | Also, detect overflow on spinlock ownership count. Also,      |             |
| 220 ; |      |         | restore SPINLOCK data type/size initialization.               |             |
| 221 ; |      |         |   |             |
| 222 ; | X-38 | RNG0038 | Rod Gamache   | 26-Jun-1986 |
| 223 ; |      |         | Don't init SPINLOCK data type/size - at least for now.        |             |
| 224 ; |      |         |   |             |
| 225 ; | X-36 | MSH0252 | Michael S. Harvey   | 16-Jun-1986 |
| 226 ; |      |         | Add support for data structure typing of the spinlock         |             |
| 227 ; |      |         | control block.  |             |
| 228 ; |      |         |   |             |
| 229 ; | X-35 | MSH0250 | Michael S. Harvey   | 12-Jun-1986 |
| 230 ; |      |         | Allow spinlock acquisition from IPL 31 for those              |             |
| 231 ; |      |         | processors that could not possibly have any                   |             |
| 232 ; |      |         | lower IPL code threads preempted, by virtue of                |             |
| 233 ; |      |         | being in a boot sequence.                                     |             |
| 234 ; |      |         |   |             |
| 235 ; | X-34 | MSH0249 | Michael S. Harvey   | 11-Jun-1986 |
| 236 ; |      |         | Start using 'real' bugcheck messages.                         |             |

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 5  
X-86 HISTORY 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (1)

237 ;  
238 ; X-29 MSH0243 Michael S. Harvey 20-May-1986  
239 ; Remove devicelocks from generalized rank enforcement.  
240 ; Also, clean up the ownership mechanisms for spinlocks.  
241 ;  
242 ; X-28 MSH0247 Michael S. Harvey 16-May-1986  
243 ; Keep track of spinlock related IPL usage for ease of  
244 ; analysis and to speed up consistency checking.  
245 ;  
246 ; MSH0246 Michael S. Harvey 15-May-1986  
247 ; Count the number of failed and successful spinlock  
248 ; acquisitions. Also, extend spin count from 64 to 96 bits.  
249 ;  
250 ; X-27 MSH0245 Michael S. Harvey 14-May-1986  
251 ; Indicate when the CPU is busywaiting to avoid charging  
252 ; a process for the time involuntarily spent waiting  
253 ; for access to the spinlock.  
254 ;  
255 ; X-26 MSH0243 Michael S. Harvey 14-May-1986  
256 ; When busy-waiting, do so at the lowest practical IPL  
257 ; to avoid unnecessary blockage of other events.  
258 ;  
259 ; X-25 MSH0244 Michael S. Harvey 14-May-1986  
260 ; Fix out of phase symbols.  
261 ;  
262 ; X-24 MSH0243 Michael S. Harvey 13-May-1986  
263 ; Reverse rank assignments and redesign the rank verification  
264 ; code to remove some bugs and speed up the code overall. Make  
265 ; other changes to speed up spinlock acquisition and release.  
266 ; Interlock access to the spinlock structure itself.  
267 ;  
268 ; X-23 MSH0242 Michael S. Harvey 8-May-1986  
269 ; Replace logical CPU ID with physical CPU ID.  
270 ;  
271 ; X-1F2 RNG1002 Rod Gamache 7-May-1986  
272 ; Remove extra check in SMP\$ACQUIRE. Add comments.  
273 ;  
274 ; X-1A7 RNG4007 Rod Gamache 4-Mar-1986  
275 ; Move SMP\$FIND\_SELF routine to module SMPROUT.  
276 ;  
277 ; X-1A6 RNG4006 Rod Gamache 19-Feb-1986  
278 ; Re-position SMP\$FIND\_SELF to a PSECT which is not  
279 ; overlaid by BUF\$FATAL stuff.  
280 ;  
281 ; X-1A5 RNG4005 Rod Gamache 18-Feb-1986  
282 ; Add \$CPUDEF external call.  
283 ;  
284 ; X-1A4 RNG4004 Rod Gamache 7-Feb-1986  
285 ; Add routine to find the PER CPU data area given on the  
286 ; PHYSICAL CPU ID.  
287 ;  
288 ; X-1A3 RNG4003 Rod Gamache 25-Nov-1985  
289 ; Add routines to allocate and initialize a spinlock.  
290 ;  
291 ; X-1A2 RNG4002 Rod Gamache 19-Nov-1985  
292 ; Add routines to make sure a specified spinlock is held and  
293 ; that no spinlocks are held.

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 6  
X-86 HISTORY 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (1)

294 ;  
295 ;--

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 7  
X-86 DECLARATIONS 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (2)

```
297      .SBTTL  DECLARATIONS
298
299      $BIICDEF      ; Define BIIC offsets
300      $CPBDEF      ; Define CPU capabilities
301      $CPUDEF      ; Define per CPU data area
302      $DYNDEF      ; Define data structure types
303      $IPLDEF      ; Define IPLs
304      $KA810DEF    ; Define KA820 node private space
305      $PRDEF       ; Define Processor Registers
306      $PR8SSDEF    ; Define KA820 processor registers
307      $PSLDEF      ; Define Processor Status Longword
308      $$SPLCODDEF  ; Define spinlock indices
309      $$SPLDEF     ; Define Spinlock structures
310      $$SDEF       ; Define status returns
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 8  
X-86 SPINWAIT - Spinlock wait timeout macro 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (3)

```

312      .SBTTL  SPINWAIT - Spinlock wait timeout macro
313
314 ;
315 ; SPINWAIT: Macro to time out spinlock spinwaits.
316 ;
317 ; Parameters:  SPINLOCK : register containing address of spinlock
318 ;                for which to wait
319 ;                CPUBASE : register containing base address of CPU database
320 ;
321 ;                DONELBL  : label at end of successful completion of spinwait
322 ;                L1, etc.: local labels
323 ;
324 ; This macro will result in a bugcheck if the spinlock interlock is not
325 ; freed within the timeout interval contained in SPL$$_TIMO_INT.
326 ;
327      .MACRO  SPINWAIT SPINLOCK, INS1, INS2, INS3, INS4, INS5, CPUBASE=R1, DONELBL, -
                                     ?L1, ?L2, ?L3, ?L4, ?L5, ?L6
328
329      .LIST  MEB
330      MFPR  #PR$_IPL, -(SP)          ; Save current IPL
331      CLRQ  -(SP)                   ; Make room on stack
332 L1:     MOVL  SPL$_OWN_CPU(SPINLOCK), -
333          4(SP)                     ; save current owner
334      MULL3 SPL$_TIMO_INT(SPINLOCK), -
335          CPU$_UBDELAY(CPUBASE), (SP)
336      MULL  CPU$_TENUSEC(CPUBASE), (SP)
337 L2:
338          'INS1'
339          'INS2'
340          'INS3'
341          'INS4'
342          'INS5'
343      CMPL  8(SP), G^EXE$GL_IPINT_IPL      ; Blocking IP interrupts?
344      BLSS  L4                             ; If LSS no, just spinwait
345      BBS   #15, G^XDT$GW_OWNER_ID, L5     ; Is XDELTA active?
346      DSBINT ENVIRON=UNIPROCESSOR        ; Yes, prepare for benign state
347      JSB   G^XDT$CPU_WAIT                ; Enter benign state
348      ENBINT                                     ; Clean up from benign state
349 L5:     BBS   #CPU$_V_BUGCHK, -          ; Watch for pending bugcheck
350          CPU$_WORK_REQ(CPUBASE), L3      ;
351      BBC   CPU$_PHY_CPUID(CPUBASE), -
352          G^SMP$GL_OVERRIDE, L4          ; Branch if not an OVERRIDE
353          ; set member
354      ASSUME CPB$_C_PRIMARY EQ 0
355      BLBC  CPU$_CAPABILITY(CPUBASE), L6   ; Ignore VIRTCONS if secondary
356      .IF DIF <CPUBASE>, R1
357      PUSHR #^M<R0, R1, R2>                ; Save some registers
358      MOVL  CPUBASE, R1                    ; Copy per-CPU db address
359      .IFF
360      PUSHR #^M<R0, R2>                    ; Save some registers
361      .IFTF
362      JSB   G^SMP$VIRTCONS_SERVER         ; Check for VIRTCONS request
363      .IFT
364      POPR  #^M<R0, R1, R2>                ; Restore registers
365      .IFF
366      POPR  #^M<R0, R2>                    ; Restore registers
367 L6:     BCCI  #CPU$_V_INV_TBS, -          ; Watch for pending TBIS
368          CPU$_WORK_REQ(CPUBASE), L4      ;

```



**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 9  
X-86 SPINWAIT - Spinlock wait timeout macro 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (3)

```

369      .IF DIF <CPUBASE>,R1
370      PUSHR    #^M<R0,R1,R2>                ; Save some registers
371      MOVL     CPUBASE,R1                    ; Copy per-CPU db address
372      .IFF
373      PUSHR    #^M<R0,R2>                    ; Save some registers
374      .IFTF
375      JSB      G^SMP$INVALID_SINGLE          ; Participate in TBIS exchange
376      .IFT
377      POPR     #^M<R0,R1,R2>                ; Restore registers
378      .IFF
379      POPR     #^M<R0,R2>                    ; Restore registers
380      .ENDC
381
382 L4:    SOBGTR  (SP),L2                        ; decrement retry count and retry
383 ;
384 ; The spinlock did not become available in the specified period of time.
385 ;
386      CML     4(SP),-
387      SPL$L_OWN_CPU(SPINLOCK) ; has owner changed?
388      BNEQ    L1                               ; yes, give new quantum and retry
389      BSBW    SMP$TIMEOUT                       ; Bugcheck if necessary
390      BRB     L1                               ; Retry if we come back from TIMEOUT
391 L3:    BUG_CHECK CPUEXIT,FATAL                ; This CPU was induced to bugcheck
392
393 DONELBL:
394      ADDL    #12,SP                            ; clean up stack
395
396      .NLIST  MEB
397      .ENDM   SPINWAIT
398
399
400      DECLARE_PSECT  EXEC$NONPAGED_CODE
401

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 10  
X-86 ACQUIRE - Acquire a spinlock through bus 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (

```
403      .SBTTL ACQUIRE - Acquire a spinlock through busy waiting
404 ;++
405 ; SMP$ACQUIRE - Acquire a SPINLOCK or FORKLOCK and force synchronization
406 ; SMP$ACQUIREL - Acquire a DEVICELOCK and force synchronization
407 ; SMP$ACQNOIPL - Acquire a DEVICELOCK without further synchronization
408 ;
409 ;       These subroutines acquire a spinlock with an interlocked operation.
410 ;       If the spinlock is already in use, the CPU busy waits.
411 ;
412 ; Calling sequence:
413 ;
414 ;       JSB/BSBx from KERNEL mode
415 ;
416 ; Input Parameters:
417 ;
418 ;       RO -> The number of the lock being acquired for entry SMP$ACQUIRE
419 ;       RO -> The spinlock structure address for entry SMP$ACQUIREL and
420 ;             SMP$ACQNOIPL
421 ;
422 ; Implicit Inputs:
423 ;
424 ;       IPL is less than or equal to IPL required by lock for entries
425 ;             SMP$ACQUIRE and SMP$ACQUIREL.
426 ;
427 ; Output Parameters:
428 ;
429 ; Implicit Outputs:
430 ;
431 ;       Local processor synchronization is achieved by setting IPL to
432 ;       the IPL associated with the spinlock for entries
433 ;       SMP$ACQUIRE and SMP$ACQUIREL. Local processor
434 ;       synchronization is assumed to be already achieved
435 ;       on entry via SMP$ACQNOIPL.
436 ;
437 ;       Regardless of the entry point, the spinlock is acquired,
438 ;       thus extending the local processor synchronization to
439 ;       all other processors.
440 ;
441 ; Side Effects:
442 ;
443 ;       Note that busy-waits on locks that are held by another CPU
444 ;       will execute at the caller's IPL.
445 ;
446 ;       The spinlock performance/debug database is updated.
447 ;
448 ;       RO is altered.
449 ;
450 ;--
451
824
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 11  
X-86 ACQUIRE - Acquire a spinlock through bus 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (

```

863
864     .ENABL  LSB
865
866 ;
867 ; SPINLOCK ACQUISITION ENTRY POINTS
868 ;
869
870 10$:  DSBINT  #31,ENVIRON=UNIPROCESSOR      ; Disable all interrupts
871     find_cpu_data R1                        ; Get per-CPU database address
872     BBC      CPU$L_PHY_CPUID(R1),-         ; Force setting IPL for non-
873     G^SMP$GL_OVERRIDE,12$                 ; members of the OVERRIDE set
874     CMPB     (SP),SPL$B_IPL(R0)           ; Is entry IPL < SPINLOCK_IPL?
875     BGEQ     15$                          ; Br if no... leave IPL alone
876 12$:  MOVZBL  SPL$B_IPL(R0), (SP)         ; Return to spinlock's IPL
877 15$:  ENBINT  ; Restore IPL
878     BRB      SKIPSET                      ; And skip setting new IPL
879
880     .ALIGN  LONG
881     UNIVERSAL_SYMBOL      SMP$ACQNOIPL
882 ;SMP$ACQNOIPL::
883     PUSHL   R1                            ; Get some working registers
884     find_cpu_data R1                      ; Get per-CPU database address
885     BRB     SKIPSET                      ; Don't alter current IPL
886
887
888     .ALIGN  LONG
889     UNIVERSAL_SYMBOL      SMP$ACQUIRE
890 ;SMP$ACQUIRE::
891     MOVL    W^SMP$GL_SPNLKVEC[R0],R0      ; Get spinlock address in R0
892     Get the address of the spin lock data structure
893
894     UNIVERSAL_SYMBOL      SMP$ACQUIREL
895 ;SMP$ACQUIREL::
896     PUSHL   R1                            ; Get some working registers
897     TSTL    G^SMP$GL_OVERRIDE             ; Is anyone in override?
898     BNEQ    10$                          ; Br if yes, do long test
899     ;
900     ; Else, raise the current IPL
901     ;
902     SETIPL  SPL$B_IPL(R0),-              ; who change process IPL to spinlock's IPL
903     ENVIRON=UNIPROCESSOR
904     find_cpu_data R1                      ; Get per-CPU database address
905
906 SKIPSET:
907
908 ;*****
909 ;                               ACQUIRE SPINLOCK
910 ;*****
911 ;
912 ; Attempt to acquire the spinlock itself with interlocked protection. If
913 ; the acquisition fails, busy wait for the lock to become available.
914 ;
915 ;     R0      = Address of desired spinlock structure
916 ;     R1      = Address of current per-CPU database
917 ;     (SP)    = Saved R1
918 ;     4(SP)   = Caller's PC
919

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 12  
X-86 ACQUIRE - Acquire a spinlock through bus 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (

```

920 20$:   BBSSI   #SPL$V_INTERLOCK,-           ; Interlock access to variable
921                SPL$B_SPINLOCK(R0),40$      ; part of spinlock structure
922
923 ; The lock is now acquired for this CPU by virtue of the interlock bit changing
924 ; from 0 to 1.
925
926        MOVL   R1,SPL$L_OWN_CPU(R0)         ; Remember which CPU acquired it
927 30$:   INCW   SPL$W_OWN_CNT(R0)           ; Acquire spinlock ownership
928 35$:   POPL   R1                          ; Restore registers
929        RSB                   ; Return with spinlock acquired
930
931 ; Come here if the attempt to acquire a spinlock failed because the spinlock
932 ; is already owned. There are two important cases to consider. Either the
933 ; spinlock is already owned by the current CPU, or by another CPU. In the
934 ; former case, allow this acquisition attempt to complete successfully. In
935 ; the latter case, this CPU must busy wait until the desired spinlock is
936 ; released by the owning CPU.
937 ;
938 ; Assumptions:
939 ;
940 ;     o   Cache is enabled - thus preventing accesses directly to memory
941 ;         while we are spinwaiting. This prevents excessive memory
942 ;         accesses while spinwaiting.
943 ;
944 ;     o   Cache coherency is maintained by the hardware. Therefore, the
945 ;         cache contents will be immediately updated whenever the owning
946 ;         CPU releases the spinlock.
947 ;
948 40$:   CMPL   R1,SPL$L_OWN_CPU(R0)         ; Is spinlock already owned
949                ; by this CPU?
950        BEQL   30$                          ; Br if yes, continue
951
952 ; Detect bugcheck caller and allow immediate lock 'break' (if PRIMARY)
953 ; This check must discriminate between CPUs, e.g. one CPU may enter
954 ; BUG_CHECK while another CPU hasn't. Therefore, we cannot simply change
955 ; the transfer vector to another routine and simplify the checks used in
956 ; the 'normal' path.
957 ;
958 ; Also, we MUST allow for the PRIMARY to 'break' locks held by other
959 ; CPUs when it is completing the 'BUGCHECK'.
960
961        BBC    CPU$L_PHY_CPUID(R1),-       ; Br if this CPU did not bugcheck
962                G^SMP$GL_BUG_DONE,60$      ;
963
964        CMPL   CPU$L_PHY_CPUID(R1),-       ; Is this the PRIMARY?
965                G^SMP$GL_PRIMID           ;
966        BEQL   30$                          ; Br if yes, Allow PRIMARY to
967                ; 'break' lock
968        BUG_CHECK INCONSTATE,FATAL        ; Should never get to this state
969
970 60$:   INCB   CPU$B_BUSYWAIT(R1)         ; Avoid charging busywait time
971
972 ; The busy wait loop below assumes that a cache is present and that
973 ; cache coherency is maintained on all available processors. Note that
974 ; if the cache is not present and working, the busy wait is likely to
975 ; impact the system performance overall by making many references to memory.
976

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

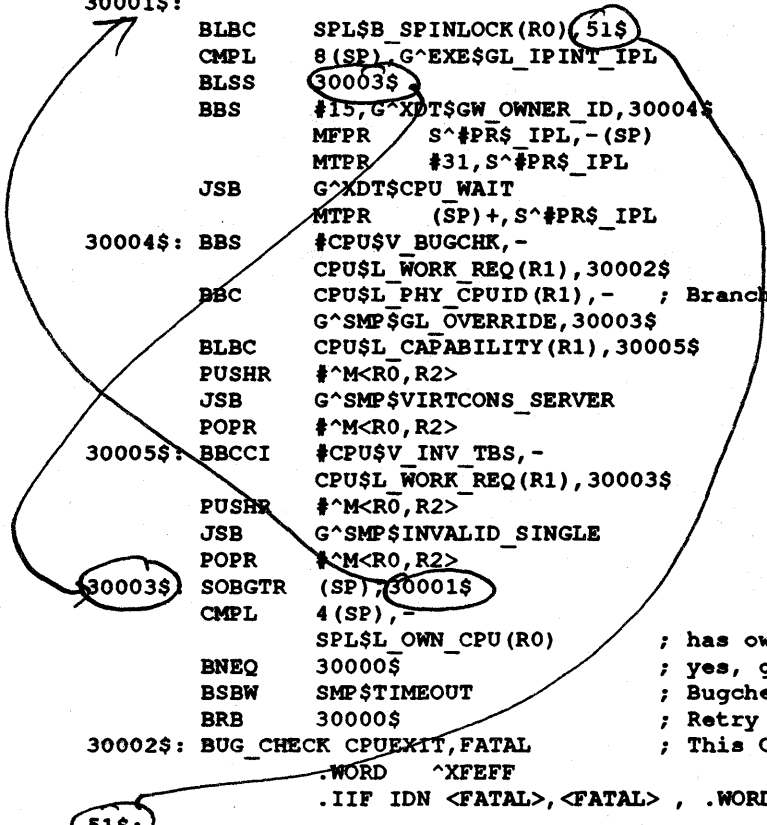
SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 13  
X-86 ACQUIRE - Acquire a spinlock through bus 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (

```

977
978     ASSUME SPL$V_INTERLOCK EQ 0
979     SPINWAIT SPINLOCK=R0,-                ; Spin waiting for lock
980     INS1=<BLBC     SPL$B_SPINLOCK(R0),51$>,-
981     DONELBL=51$,-
982     CPUBASE=R1
      MFPR #PR$ IPL,-(SP)                ; Save current IPL
      CLRQ -(SP)                        ; Make room on stack
30000$: MOVL SPL$L_OWN_CPU(R0),-
      4(SP)                            ; save current owner
      MULL3 SPL$L_TIMO_INT(R0),-
      CPU$L_UBDELAY(R1),(SP) }          calculate how many spins CPU is
      MULL CPU$L_TENUSEC(R1),(SP) }    to wait
30001$: BLBC SPL$B_SPINLOCK(R0),51$
      CMPL 8(SP),G^EXE$GL_IPINT_IPL    ; Blocking IP interrupts?
      BLSS 30003$                       ; If LSS no, just spinwait
      BBS #15,G^XDT$GW_OWNER_ID,30004$ ; Is XDELTA active?
      MFPR S^#PR$ IPL,-(SP)
      MTPR #31,S^#PR$ IPL
      JSB G^XDT$CPU_WAIT                ; Enter benign state
30004$: BBS MTPR (SP)+,S^#PR$ IPL
      #CPU$V_BUGCHK,-                  ; Watch for pending bugcheck
      CPU$L_WORK_REQ(R1),30002$
      BBC CPU$L_PHY_CPUID(R1),-        ; Branch if not an OVERRIDE
      G^SMP$GL_OVERRIDE,30003$         ; set member
      BLBC CPU$L_CAPABILITY(R1),30005$ ; Ignore VIRTCONS if secondary
      PUSHR #^M<R0,R2>                 ; Save some registers
      JSB G^SMP$VIRTCONS_SERVER        ; Check for VIRTCONS request
      POPR #^M<R0,R2>                 ; Restore registers
30005$: BBCCI #CPU$V_INV_TBS,-        ; Watch for pending TBIS
      CPU$L_WORK_REQ(R1),30003$
      PUSHR #^M<R0,R2>                 ; Save some registers
      JSB G^SMP$INVALID_SINGLE         ; Participate in TBIS exchange
      POPR #^M<R0,R2>                 ; Restore registers
30003$: SOBGTR (SP),30001$             ; decrement retry count and retry
      CMPL 4(SP),-
      SPL$L_OWN_CPU(R0)                ; has owner changed?
      BNEQ 30000$                      ; yes, give new quantum and retry
      BSBW SMP$TIMEOUT                 ; Bugcheck if necessary
      BRB 30000$                       ; Retry if we come back from TIMEOUT
30002$: BUG_CHECK CPUEXIT,FATAL
      .WORD ^XFEFF
      .IIF IDN <FATAL>,<FATAL> , .WORD BUG$_CPUEXIT!4
51$:
983     ADDL #12,SP                      ; clean up stack
984     DECB CPU$B_BUSYWAIT(R1)         ; Resume charging process
985     BRW 20$                          ; Retry if available
986     .DSABL LSB
987
989
991

```

*calculate how many spins CPU is to wait*



*clear*

*tyagani*

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 14  
X-86 RESTORE - Conditionally release a spinlo 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (

```
993      .SBTTL RESTORE - Conditionally release a spinlock
994 ;++
995 ; SMP$RESTORE - Conditionally release a spinlock or forklock
996 ; SMP$RESTOREL - Conditionally release a devicelock
997 ;
998 ;      This subroutine releases a single acquisition on a spinlock that
999 ;      another CPU may be busy-waiting on. If this is the last acquisition
1000 ;      of this spinlock by the current CPU, the spinlock is effectively
1001 ;      made available to other CPUs.
1002 ;
1003 ; Calling sequence:
1004 ;
1005 ;      JSB/BSBx from KERNEL mode
1006 ;
1007 ; Input Parameters:
1008 ;
1009 ;      R0 -> The number of the lock being acquired for entry SMP$RESTORE.
1010 ;      R0 -> The spinlock structure address for entry SMP$RESTOREL.
1011 ;
1012 ; Implicit Inputs:
1013 ;
1014 ;      IPL must be greater than ASTDEL and must be no lower than the IPL
1015 ;      at which the spinlock had originally been acquired.
1016 ;
1017 ; Implicit Outputs:
1018 ;
1019 ;      The appropriate interlocked bit is cleared for the spinlock.
1020 ;
1021 ; Side Effects:
1022 ;
1023 ;      The spinlock becomes available for another CPU to acquire if the
1024 ;      ownership count drops far enough.
1025 ;
1026 ;--
1027
1123
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 15  
X-86 RESTORE - Conditionally release a spinlo 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (

```
1141
1142     .ENABL  LSB
1143 ;
1144 ; SPINLOCK RESTORE ENTRY POINTS
1145 ;
1146
1147     .ALIGN  LONG
1148     UNIVERSAL_SYMBOL      SMP$RESTORE
1149 ;SMP$RESTORE::          ; Remove one acquisition
1150     MOVL    W^SMP$GL_SPNLKVEC[R0],R0      ; Get spinlock address in R0
1151
1152     UNIVERSAL_SYMBOL      SMP$RESTOREL
1153 ;SMP$RESTOREL::
1154 ;
1155 ; Decrement ownership count, leave if still owned
1156 ;
1157     DECW    SPL$W_OWN_CNT(R0)              ; One less lock of database
1158     BLSS   RELEASE                      ; Br if no locks left
1159     RSB
1160
1161     .DSABL  LSB
1162
1164
1166
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 16  
X-86 RELEASE - Release a spinlock 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (10)

```
1168      .SBTTL RELEASE - Release a spinlock
1169 ;++
1170 ; SMP$RELEASE - Release a spinlock or forklock
1171 ; SMP$RELEASEL - Release a devicelock
1172 ;
1173 ;      This subroutine completely releases a spinlock that another CPU may
1174 ;      be busy-waiting on in a single operation. That is, the spinlock may
1175 ;      have been acquired multiple times by the current CPU, this CPU uses
1176 ;      this subroutine to release all acquisitions of this spinlock all at
1177 ;      once.
1178 ;
1179 ; Calling sequence:
1180 ;
1181 ;      JSB/BSBx from KERNEL mode
1182 ;
1183 ; Input Parameters:
1184 ;
1185 ;      R0 -> The number of the lock being acquired for entry SMP$RELEASE.
1186 ;      R0 -> The spinlock structure address for entry SMP$RELEASEL.
1187 ;
1188 ; Implicit Inputs:
1189 ;
1190 ;      IPL must be greater than ASTDEL and must be no lower than the IPL
1191 ;      at which the spinlock had originally been acquired.
1192 ;
1193 ; Implicit Outputs:
1194 ;
1195 ;      The appropriate interlocked bit is cleared for the spinlock.
1196 ;
1197 ; Side Effects:
1198 ;
1199 ;--
```



CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 17  
X-86 RELEASE - Release a spinlock 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (11)

```
1325
1342
1343     .ENABLE LSB
1344 ;
1345 ; SPINLOCK RELEASE ENTRY POINTS
1346 ;
1347
1348     .ALIGN LONG
1349     UNIVERSAL_SYMBOL      SMP$RELEASE
1350 ;SMP$RELEASE::          ; Release spinlock
1351     MOVL      W^SMP$GL_SPNLKVEC[R0],R0      ; Get spinlock address in R0
1352
1353     UNIVERSAL_SYMBOL      SMP$RELEASEL
1354 ;SMP$RELEASEL::
1355     MCOMW     #0,SPL$W_OWN_CNT(R0)          ; Release spinlock completely
1356
1357 RELEASE:
1358
1359     CLRL      SPL$L_OWN_CPU(R0)              ; Invalidate owner CPU field
1360
1361 ; The spinlock is now released by virtue of the interlock bit changing
1362 ; from 1 to 0.
1363
1364     BBCCI     #SPL$V_INTERLOCK,-            ; Unlock spinlock data structure
1365             SPL$B_SPINLOCK(R0),60$        ;
1366 50$:      RSB
1367
1368 60$:      BUG_CHECK SPIRELERR,FATAL        ; Error releasing spinlock
1369     .DSABL   LSB
1370
1372
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 18  
X-86 REI\_CHECK - Check spinlock database cons 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (

```
1375      .SBTTL REI_CHECK - Check spinlock database consistency
1376 ;++
1377 ; SMP$REI_CHECK - Check spinlock database consistency
1378 ;
1379 ;      This subroutine makes sure that all locks are released before
1380 ;      performing an REI.
1381 ;
1382 ;      If the current mode is USER, SUPERVISOR or EXECUTIVE, this
1383 ;      routine assumes that no spinlocks are held by this CPU and
1384 ;      simply REIs for the caller.
1385 ;
1386 ; Calling sequence:
1387 ;
1388 ;      JSB/BSBx      G^SMP$REI_CHECK from any mode
1389 ;
1390 ;      NOTE - Although we get here via a JSB, the return PC is popped from
1391 ;      the stack just prior to performing the REI. The reason that
1392 ;      there is a JSB rather than a JMP is so we can see where the
1393 ;      bad REI was being performed.
1394 ;
1395 ; Input Parameters:
1396 ;
1397 ;      0(SP) = Caller return address
1398 ;      4(SP) = PC for REI instruction
1399 ;      8(SP) = PSL for REI instruction
1400 ;
1401 ; Implicit Inputs:
1402 ;
1403 ; Implicit Outputs:
1404 ;
1405 ;      This code does not return to the caller. An REI to the PC/PSL
1406 ;      pair below the caller's return address on the stack is executed.
1407 ;
1408 ; Side Effects:
1409 ;
1410 ;      ***TBS***
1411 ;
1412 ;--
1413
1446
1447      UNIVERSAL_SYMBOL      SMP$REI_CHECK
1448 ;SMP$REI_CHECK::          ; Verify spinlock IPL usage
1449      TSTL      (SP)+      ; Clean caller's address from stack
1450      REI              ; Return from exception or interrupt
1451
1453
```

CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 19  
X-86 NOLOCKS - Make sure no spinlocks are held 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (

```
1455      .SBTTL NOLOCKS - Make sure no spinlocks are held
1456 ;++
1457 ; SMP$NOLOCKS - Make sure no spinlocks are owned.
1458 ;
1459 ;      This subroutine makes sure that no spinlocks are held by this CPU
1460 ;      at or above the current IPL.
1461 ;
1462 ; Calling sequence:
1463 ;
1464 ;      JSB/BSBx from KERNEL mode
1465 ;
1466 ; Input Parameters:
1467 ;
1468 ;      None
1469 ;
1470 ; Implicit Inputs:
1471 ;
1472 ;      Spinlock database.
1473 ;      IPL must be greater than ASTDEL.
1474 ;
1475 ; Implicit Outputs:
1476 ;
1477 ;      None
1478 ;
1479 ; Side Effects:
1480 ;
1481 ;      ***TBS***
1482 ;
1483 ;--
1484
1505
1506      UNIVERSAL_SYMBOL      SMP$NOLOCKS
1507 ;SMP$NOLOCKS::
1508      RSB
1509      ; Return to caller
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 20  
X-86 SMP\$CHKLOCK - Make sure spinlock is owned 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (

```
1512      .SBTTL SMP$CHKLOCK - Make sure spinlock is owned before proceeding
1513 ;++
1514 ; SMP$CHKLOCK - Make sure spinlock is owned by this CPU before proceeding
1515 ;
1516 ;      This subroutine makes sure that a specified spinlock is owned
1517 ;      by the current CPU before proceeding.
1518 ;
1519 ; Calling sequence:
1520 ;
1521 ;      JSB/BSBx from KERNEL mode
1522 ;
1523 ; Input Parameters:
1524 ;
1525 ;      R0 = Spinlock index
1526 ;
1527 ; Implicit Inputs:
1528 ;
1529 ;      Spinlock database.
1530 ;
1531 ; Implicit Outputs:
1532 ;
1533 ;      IPL must be greater than ASTDEL.
1534 ;
1535 ; Side Effects:
1536 ;
1537 ;      R0 is altered
1538 ;
1539 ;--
1581
1582      UNIVERSAL_SYMBOL      SMP$CHKLOCK
1583 ;SMP$CHKLOCK::          ; Verify ownership of specified lock
1584      RSB                  ; Return to caller
1585
```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 21  
X-86 ALLOC\_SPL - Allocate a spinlock 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (15)

```

1588      .SBTTL ALLOC_SPL - Allocate a spinlock
1589 ;++;
1590 ; SMP$ALLOC_SPL - Allocate a dynamic spinlock
1591 ;
1592 ;      This subroutine allocates a dynamic spinlock and zeroes the structure.
1593 ;
1594 ; Calling sequence:
1595 ;
1596 ;      JSB/BSBx from KERNEL mode
1597 ;
1598 ; Inputs:
1599 ;
1600 ;      None.
1601 ;
1602 ; Implicit Inputs:
1603 ;
1604 ;      Spinlock database.
1605 ;
1606 ; Outputs:
1607 ;
1608 ;      R0 = status of request, SS$_SUCCESS or SS$_INSMEM
1609 ;      R2 = address of structure on success
1610 ;
1611 ; Implicit Outputs:
1612 ;
1613 ;      None
1614 ;
1615 ; Side Effects:
1616 ;
1617 ;      R1 is destroyed.
1618 ;
1619 ;--
1620
1621      UNIVERSAL_SYMBOL      SMP$ALLOC_SPL
1622 ;SMP$ALLOC_SPL::
1623      PUSHR      #^M<R3,R4,R5>      ; Save registers
1624      MOVZBL     #SPL$K_LENGTH,R1    ; Get size of spinlock structure
1625      JSB       G^EXE$ALONONPAGED   ; Allocate the structure
1626      BLBC      R0,110$              ; Exit if error
1627      PUSHL     R2                    ; Save spinlock address
1628      MOVCS     #0,(SP),#0,R1,(R2)   ; Zero the structure
1629      POPL      R2                    ; Save spinlock address
1630      MOVL      #<<SPL$C_SPL_DEVICELOCK@24>!<DYN$C_SPL@16>!SPL$C_LENGTH>,-
1631      SPL$W_SIZE(R2)                ; Set data structure information
1632      MOVZBL     #1,R0                ; Return success
1633 90$: POPR      #^M<R3,R4,R5>      ; Restore registers
1634      RSB
1635
1636 110$: MOVZWL     #SS$_INSMEM,R0      ; Return error code
1637      BRB      90$                  ; Leave

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 22  
X-86 INIT\_SPL - Initialize a spinlock 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (16)

```

1639      .SBTTL INIT_SPL - Initialize a spinlock
1640 ;++
1641 ; SMP$INIT_SPL - Initialize a dynamic spinlock
1642 ;
1643 ;      This subroutine initializes a dynamic spinlock and zeroes the structure.
1644 ;
1645 ; Calling sequence:
1646 ;
1647 ;      JSB/BSBx from KERNEL mode
1648 ;
1649 ; Inputs:
1650 ;
1651 ;      None.
1652 ;
1653 ; Implicit Inputs:
1654 ;
1655 ;      Spinlock database.
1656 ;
1657 ; Outputs:
1658 ;
1659 ;      R0 = IPL for spinlock
1660 ;
1661 ; Implicit Outputs:
1662 ;
1663 ;      None
1664 ;
1665 ; Side Effects:
1666 ;
1667 ;      None.
1668 ;
1669 ;--
1670
1671      UNIVERSAL_SYMBOL      SMP$INIT_SPL
1672 ;SMP$INIT_SPL::
1673      MOVB      R0,SPL$B_IPL(R2)      ; Initialize IPL for spinlock
1674      MCOMB     #0,SPL$B_RANK(R2)     ; Initialize rank field
1675      MCOMW     #0,SPL$W_OWN_CNT(R2)  ; Initialize ownership
1676      CLRL     SPL$L_OWN_CPU(R2)     ; No cpu owner
1677
1678      ASSUME     SPL$W_SIZE+2 EQ SPL$B_TYPE
1679      ASSUME     SPL$B_TYPE+1 EQ SPL$B_SUBTYPE
1680      MOVL     #<<SPL$C_SPL_DEVICELOCK@24>!<DYN$C_SPL@16>!SPL$C_LENGTH>,-
1681              SPL$W_SIZE(R2)        ; Set data structure information
1682      CMPB     SPL$B_IPL(R2),#IPL$SCS ; locks at IPL <= 8 get long
1683      BGTRU    10$                   ; timeout interval
1684      MOVL     G^SGN$GL_SMP_LNGSPINWAIT,-
1685              SPL$L_TIMO_INT(R2)    ; set up LONG timeout interval
1686      RSB
1687      10$:    MOVL     G^SGN$GL_SMP_SPINWAIT,-
1688              SPL$L_TIMO_INT(R2)    ; set up timeout interval
1689      RSB
1690              ; Return to caller

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 23  
X-86 ADJUST\_IPL - ADJUST THE IPL OF A LOCK, A 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (

```

1692      .SBTTL  ADJUST_IPL - ADJUST THE IPL OF A LOCK, ALTER RANKS ACCORDINGLY
1693
1694 ;+++
1695 ;      SMP$ADJUST_IPL  - Alter IPL of a static lock as required to
1696 ;                      preserve integrity of the acquisition process.
1697 ;                      This requires the lock's rank to remain unchanged.
1698 ;
1699 ;  INPUT:      Calling environment: JSB/BSBx
1700 ;           R0:      Lock address
1701 ;           R1:      New IPL
1702 ;
1703 ;  OUTPUT:
1704 ;           R0 has LBS on success. If R0 has LBC on exit, non-fatal error.
1705 ;
1706 ;           Implicit output: Ranks of locks will shuffle to accomodate new
1707 ;           placement of the passed lock; passed lock will be given passed
1708 ;           IPL.
1709 ;
1710 ;  NOTES:
1711 ;           Fatal errors exit via BUG_CHECK .
1712 ;
1713 ;---
1714
1715      .ENABL  LSB
1716
1717      UNIVERSAL_SYMBOL      SMP$ADJUST_IPL
1718
1719 ;SMP$ADJUST_IPL::
1720
1721      PUSHR   #^M<R2,R3,R4>          ; R0, R1 destroyed during operation.
1722
1723      MOVL   R0,R2                    ; Keep address in a safe register
1724      MOVL   R1,R4                    ; Keep IPL in a safe register
1725 ;
1726 ; Firewall code: Verify (address) to be a spinlock and bounds check the IPL.
1727 ;
1728      CMPW   #SPL$C_LENGTH,SPL$W_SIZE(R2) ; This thing the right size?
1729      BNEQU  10$                       ; NEQ: No!
1730      CMPB   #DYN$C_SPL,SPL$B_TYPE(R2)   ; This thing an SPL?
1731      BEQL   20$                       ; NEQ: No!
1732 10$:      BUG_CHECK SPLNOTSPL,FATAL      ; Bug out: invalid structure.
1733
1734 20$:      CMPB   R4,#IPL$RESCHED        ; Now check IPL.
1735      BLSSU  30$
1736      CMPB   R4,#31                     ; Accept only 3 <= IPL <= 31.
1737      BLEQU  40$
1738 30$:      BUG_CHECK SPLINVIPL,FATAL     ; Bug out: invalid IPL.
1739 ;
1740 ; Load flag value into R3. A zeroed R3 will mean RAISE THE IPL.
1741 ;
1742 40$:      CLRL   R3                     ; Assume we will RAISE IPL.
1743      CMPB   SPL$B_IPL(R2),R4           ; Are we changing IPL to same IPL?
1744      BEQL   EXIT_SUCCESS              ; EQL: yes, so do nothing, just exit.
1745      BLSSU  60$
1746      INCL   R3                         ; GTRU : We are lowering the IPL.
1747 60$:      MOVZWL #SPL$MIN_INDEX,R1     ; Set R3 non-zero for LOWERING IPL.
1748 70$:      CMPL  W^SMP$GL_SPNLKVEC[R1],R2 ; Scan spinlock vector to find match.

```

**CONFIDENTIAL AND PROPRIETARY  
DIGITAL EQUIPMENT CORPORATION**

SPINLOCKS - Spinlock routines for VMS/SMP 10-MAY-1989 16:58:37 VAX MACRO V5.0-8 Page 24  
X-86 ADJUST\_IPL - ADJUST THE IPL OF A LOCK, A 1-APR-1988 09:19:41 [SYS.SRC]SPINLOCKS.MAR;1 (

```

1749          BEQL    80$                ; If equal, R1 holds our map index.
1750          AOBLSS #SPL$ _MAX_INDEX,R1,70$ ; Loop until end or branch-exit.
1751 ;
1752 ; If we get here, trouble - the passed lock was not in the vector.
1753 ;
1754          BUG_CHECK SPLNOTMAP,FATAL    ; Bugcheck, lock was not found.
1755 ;
1756 ; Register assignments are as follows:
1757 ;
1758 ;      R1 -- will hold index of lock above/below passed lock.
1759 ;      R2 -- passed lock address.
1760 ;      R3 -- contains zero if we are raising IPL, nonzero otherwise.
1761 ;      R4 -- new ipl for passed lock.
1762 ;
1763 80$:      TSTL    R3                ; Is the next lock above or below?
1764          BNEQ    90$                ; NEQ, below; EQL, above.
1765 82$:      DECL    R1                ; Move index down to next lock.
1766          BLSS    EXIT_IPL          ; Off the low end? Then done.
1767          MOVL    W^SMP$GL_SPNLKVEC[R1],R0 ; Get address of 'next' spinlock
1768          BGEQ    82$                ; Valid locks are in S0 space
1769          CMPB    R4,SPL$B_IPL(R0)   ; Compare against new IPL.
1770          BLEQ    EXIT_IPL          ; If new IPL > other's IPL,
1771          ; rank would have to change.
1772 85$:      BUG_CHECK SPLINVIPL,FATAL  ; Bugcheck on attempted rank changes.
1773 ;
1774 90$:      INCL    R1                ; Move index up to next lock.
1775          CMPL    #SPL$ _MAX_INDEX,R1 ; Off the high end?
1776          BGEQ    EXIT_IPL          ; GEQ: Yes. Then done.
1777          MOVL    W^SMP$GL_SPNLKVEC[R1],R0 ; Get address of higher spinlock
1778          BGEQ    90$                ; Valid locks are in S0 space
1779          CMPB    R4,SPL$B_IPL(R0)   ; Compare against new IPL.
1780          BLSS    85$                ; If new IPL < other's IPL, fatal error.
1781 ;
1782 EXIT_IPL:
1783          MOVB    R4,SPL$B_IPL(R2)   ; Store new IPL into passed lock.
1784 EXIT_SUCCESS:
1785          MOVZBL  S^#SS$ _NORMAL,R0  ; Indicate successful exit.
1786 EXIT:     POPR    #^M<R2,R3,R4>    ; Restore registers and ...
1787          RSB                    ; ... exit. R0 has LBS on success.
1788 ;
1789          .DSABL  LSB
1790 ;
1791          .END

```