



\*\*FILE\*\*ID\*\*FREEPG

B 11

FF FFFFFFFF RRRRRRRR EEEEEEEEEE EEEEEEEEEE PPPPPPPP GGGGGGGG  
FF FFFFFFFF RRRRRRRR EEEEEEEEEE EEEEEEEEEE PPPPPPPP GGGGGGGG  
FF RR RR EE EE PP PP GG  
FF FFFFFF RRRRRRRR EEEEEEEEEE EEEEEEEEEE PPPPPPPP GG  
FF FFFFFF RRRRRRRR EEEEEEEEEE EEEEEEEEEE PPPPPPPP GG  
FF RR RR EE EE PP GG GGGGGG  
FF RR RR EE EE PP GG GGGGGG  
FF RR RR EE EE PP GG GG  
FF RR RR EE EE PP GG GG  
FF RR RR EEEEEEEEEE EEEEEEEEEE PP GGGGGG  
FF RR RR EEEEEEEEEE EEEEEEEEEE PP GGGGGG

LL IIIII SSSSSSS  
LL IIIII SSSSSSS  
LL II SS  
LL II SS  
LL II SS  
LL II SSSSS  
LL II SSSSS  
LL II SS  
LL II SS  
LL II SS  
LLLLLLLLLL IIIII SSSSSSS  
LLLLLLLLLL IIIII SSSSSSS

FR  
VO

```
1 0001 0 MODULE FREEPG (LANGUAGE (BLISS32) ,
2 0002 0 IDENT = 'V04-000'
3 0003 0 )
4 0004 0 =
5 0005 1 BEGIN
6 0006 1 ****
7 0007 1 *
8 0008 1 *
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
10 0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
11 0011 1 * ALL RIGHTS RESERVED.
12 0012 1 *
13 0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
14 0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
15 0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
16 0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
17 0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
18 0018 1 * TRANSFERRED.
19 0019 1 *
20 0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
21 0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
22 0022 1 * CORPORATION.
23 0023 1 *
24 0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
25 0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
26 0026 1 *
27 0027 1 *
28 0028 1 ****
29 0029 1 ++
30 0030 1 +
31 0031 1 +
32 0032 1 FACILITY: MTAACP
33 0033 1 +
34 0034 1 ABSTRACT:
35 0035 1 This module handles the requesting and returning of virtual pages.
36 0036 1 +
37 0037 1 ENVIRONMENT:
38 0038 1 +
39 0039 1 Starlet operating system, including privileged system services
40 0040 1 and internal exec routines.
41 0041 1 --
42 0042 1 ++
43 0043 1 +
44 0044 1 +
45 0045 1 +
46 0046 1 AUTHOR: D. H. GILLESPIE, CREATION DATE: 9-JUN-77
47 0047 1 +
48 0048 1 MODIFIED BY:
49 0049 1 +
50 0050 1 V02-004 DMW00023 David Michael Walp 17-Jul-1981
51 0051 1 Included change shipped with 2.4 plus improvements. Added
52 0052 1 additional comments through out the module.
53 0053 1 +
54 0054 1 V02-002 REFORMAT Maria del C. Nasr 30-Jun-1980
55 0055 1 +
56 0056 1 ++
57 0057 1 **
```

FREEPG  
V04-000

D 11  
16-Sep-1984 02:19:01  
14-Sep-1984 12:46:39 VAX-11 Bliss-32 V4.0-742  
[MTAACP.SRC]FREEPG.B32;1

Page (1) 2

: 58 0058 1 LIBRARY 'SYSSLIBRARY:LIB.L32';  
: 59 0059 1  
: 60 0060 1 REQUIRE 'SRC\$:MTADEF.B32';  
: 61 0444 1

FR  
VO

```
: 63      0445 1 GLOBAL ROUTINE GET_FREE_PAGE (PAGES, ADDR) : COMMON_CALL NOVALUE =
: 64
: 65
: 66
: 67      0447 1 ++
: 68      0448 1 FUNCTIONAL DESCRIPTION:
: 69          This routine gets the requested number of contiguous pages from
: 70          the free page list. If none are available, it expands virtual memory.
: 71
: 72      0449 1 CALLING SEQUENCE:
: 73          GET_FREE_PAGE(ARG1,ARG2)
: 74
: 75      0450 1 INPUT PARAMETERS:
: 76          ARG1 - number of pages
: 77          ARG2 - address of long word in which to return address of free page
: 78
: 79      0451 1 IMPLICIT INPUTS:
: 80          FREE_PAGE_HEAD - head of free_page list
: 81          LAST_PAGE - last page of virtual memory
: 82
: 83      0452 1 OUTPUT PARAMETERS:
: 84          ARG2 - address of long word in which to return address of free page
: 85
: 86      0453 1 IMPLICIT OUTPUTS:
: 87          none
: 88
: 89      0454 1 ROUTINE VALUE:
: 90          none
: 91
: 92      0455 1 SIDE EFFECTS:
: 93          none
: 94
: 95      0456 1 !--
: 96
: 97      0457 1 BEGIN
: 98
: 99      0458 1 EXTERNAL REGISTER
:100          COMMON_REG;
:101
:102      0459 1 EXTERNAL
:103          FREE_PAGE_HEAD : REF BBLOCK,           ! free page list head
:104          LAST_PAGE;                         ! address of last page
:105
:106      0460 1 EXTERNAL ROUTINE
:107          SYS$EXPREG : ADDRESSING_MODE (ABSOLUTE); ! expand region
:108
:109      0461 1 LOCAL
:110          SIZE;                           ! number of bytes requested
:111          FPAGE : VECTOR [2];           ! page references
:112          TOOBIG : REF BBLOCK;          ! address of space which is
:113                                         ! bigger than need be
:114
:115      0462 1 BIND
:116          FREEPAGE = FPAGE : REF BBLOCK,
:117          ENDADDR = FPAGE[1];
:118
:119      0463 1 TOOBIG = 0;                 ! initialize
```

120 0502 2 SIZE = \$12\*.PAGES;  
121 0503 2 FREEPAGE = .FREE\_PAGE\_HEAD; ! number of bytes requested  
122 0504 2 ! pickup first free page  
123 0505 2  
124 0506 2  
125 0507 2  
126 0508 2  
127 0509 2  
128 0510 2  
129 0511 3  
130 0512 3  
131 0513 3  
132 0514 3  
133 0515 3  
134 0516 3  
135 0517 3  
136 0518 4  
137 0519 4  
138 0520 4  
139 0521 3  
140 0522 3  
141 0523 3  
142 0524 3  
143 0525 3  
144 0526 3  
145 0527 3  
146 0528 3  
147 0529 3  
148 0530 3  
149 0531 3  
150 0532 2  
151 0533 2  
152 0534 2  
153 0535 2  
154 0536 2  
155 0537 2  
156 0538 2  
157 0539 2  
158 0540 3  
159 0541 3  
160 0542 3  
161 0543 3  
162 0544 3  
163 0545 2  
164 0546 2  
165 0547 2  
166 0548 2  
167 0549 3  
168 0550 3  
169 0551 3  
170 0552 3  
171 0553 3  
172 0554 2  
173 0555 2  
174 0556 2  
175 0557 2  
176 0558 1

SIZE = \$12\*.PAGES;  
FREEPAGE = .FREE\_PAGE\_HEAD; ! number of bytes requested  
! pickup first free page  
Look down the freepage list for a region of the correct or larger size.  
If we find a region of the correct size return it. Remember the first  
chunk which is too big, it will be cut down if we do not find a page of  
the correct size  
WHILE .FREEPAGE NEQA FREE\_PAGE\_HEAD DO  
BEGIN  
IF .SIZE EQLU .FREEPAGE[FVPSW\_SIZE]  
THEN  
! we found a section of the correct size, remove it from the list  
! and return it  
BEGIN  
REMQUE(.FREEPAGE, .ADDR);  
RETURN;  
END;  
IF .SIZE LSSU .FREEPAGE[FVPSW\_SIZE]  
THEN  
! this space is too big. so if we do not already have a chuck to  
cut up if needed, then remember this one  
IF .TOOBIG EQLA 0 THEN .TOOBIG = .FREEPAGE;  
.FREEPAGE = .FREEPAGE[FVPSL\_FORWARD];  
END;  
IF .TOOBIG NEQ 0  
THEN  
! if there is entry that is too big, leave it in the free page list but  
make it smaller and use the end of the block to satisfy the request  
BEGIN  
.TOOBIG[FVPSW\_SIZE] = .TOOBIG[FVPSW\_SIZE] - .SIZE;  
.FREEPAGE = .TOOBIG + .TOOBIG[FVPSW\_SIZE];  
END  
ELSE  
! otherwise expand the region and update last page pointer  
BEGIN  
IF NOT SYS\$EXPREG(.PAGES, FREEPAGE, EXEC\_MODE, 0)  
THEN  
ERR EXIT(SS\$ ACPVAFUL);  
LAST\_PAGE = .ENDADDR;  
END;  
.ADDR = .FREEPAGE;  
.FREEPAGE[FVPSW\_SIZE] = .SIZE;  
END; ! end of routine

							.TITLE FREEPG	
							.IDENT \V04-000\	
							.EXTRN FREE_PAGE_HEAD, LAST_PAGE	
							.EXTRN SYS\$EXPREG	
							.PSECT SCODE\$, NOWRT, 2	
							.ENTRY GET_FREE_PAGE, Save R2,R3	: 0445
							SUBL2 #4, SP	
							CLRL TOOBIG	: 0501
							ASHL #9, PAGES, SIZE	: 0502
							PUSHL FREE PAGE HEAD	: 0503
							MOVL FREEPAGE, R1	: 0510
							MOVAB FREE PAGE HEAD, R2	
							CMPL R1, R2	
							BEQL 4S	
							CMPZV #0, #16, 8(R1), SIZE	: 0513
							BNEQ 2S	
							REMOUE (R1), BADDR	: 0519
							RET	: 0518
							MOVL FREEPAGE, R1	: 0523
							CMPZV #0, #16, 8(R1), SIZE	
							BLEQU 3S	
							TSTL TOOBIG	: 0529
							BNEQ 3S	
							MOVL R1, TOOBIG	
							(R1), FREEPAGE	: 0531
							BRB 1S	: 0510
							TSTL TOOBIG	: 0534
							BEQL 5S	
							SUBW2 SIZE, 8(TOOBIG)	: 0541
							MOVZWL 8(TOOBIG), R1	: 0542
							ADDL3 R1, TOOBIG, FREEPAGE	
							BRB 7S	: 0534
							MOVQ #1, -(SP)	: 0550
							PUSHAB FREEPAGE	
							PUSHL PAGES	
							CALLS #4, B\$SYS\$EXPREG	
							BLBS R0, 6S	
							CHMU #764	: 0552
							MOVL ENDADDR, LAST_PAGE	: 0553
							MOVL FREEPAGE, R0	: 0554
							MOVL R0, BADDR	
							MOVW SIZE, 8(R0)	: 0557
							RET	: 0558

: Routine Size: 124 bytes,    Routine Base: SCODE\$ + 0000

: 177    0559 1

```
179 0560 1 GLOBAL ROUTINE RET_FREE_PAGE (ADDR,CONTRACT) : COMMON_CALL NOVALUE =
180 0561 1
181 0562 1 ++
182 0563 1
183 0564 1 FUNCTIONAL DESCRIPTION:
184 0565 1 This routine returns a block of contiguous pages to the free page list.
185 0566 1 If specified and the page is the last page of virtual memory, then the
186 0567 1 program section is contracted. Space is put back so that the highest
187 0568 1 address is at the tail of the queue. Contiguous memory is represented
188 0569 1 by one free page block.
189 0570 1
190 0571 1 CALLING SEQUENCE:
191 0572 1 RET_FREE_PAGE(ARG1,ARG2)
192 0573 1
193 0574 1 INPUT PARAMETERS:
194 0575 1 ARG1 - address of block to return
195 0576 1 ARG2 - TRUE or FALSE value, signaling if we should try to contract P0
196 0577 1
197 0578 1 IMPLICIT INPUTS:
198 0579 1 The size of the block to be returned is contained in the block
199 0580 1 structure.
200 0581 1
201 0582 1 OUTPUT PARAMETERS:
202 0583 1 none
203 0584 1
204 0585 1 IMPLICIT OUTPUTS:
205 0586 1 if virtual memory is contracted, last_page is updated
206 0587 1
207 0588 1 ROUTINE VALUE:
208 0589 1 none
209 0590 1
210 0591 1 SIDE EFFECTS:
211 0592 1 none
212 0593 1
213 0594 1 --
214 0595 1
215 0596 2 BEGIN
216 0597 2
217 0598 2 EXTERNAL REGISTER
218 0599 2 COMMON_REG;
219 0600 2
220 0601 2 EXTERNAL
221 0602 2 FREE_PAGE_HEAD : REF BBLOCK, ! addr of free page list head
222 0603 2 LAST_PAGE: ! addr of last page of virtual memory
223 0604 2
224 0605 2 EXTERNAL ROUTINE
225 0606 2 SY$CNTREG : ADDRESSING_MODE (ABSOLUTE);
226 0607 2
227 0608 2 MAP
228 0609 2 ADDR : REF BBLOCK; ! address of virtual memory to return
229 0610 2
230 0611 2 LOCAL
231 0612 2 FREEPAGE : REF BBLOCK, ! address of free block
232 0613 2 NEXTPAGE : REF BBLOCK, ! address of next page
233 0614 2 ENDFREE : REF BBLOCK; ! address of the last free page block
234 0615 2
235 0616 2 ! make this block a free block
```

```
: 236
: 237
: 238
: 239
: 240
: 241
: 242
: 243
: 244
: 245
: 246
: 247
: 248
: 249
: 250
: 251
: 252
: 253
: 254
: 255
: 256
: 257
: 258
: 259
: 260
: 261
: 262
: 263
: 264
: 265
: 266
: 267
: 268
: 269
: 270
: 271
: 272
: 273
: 274
: 275
: 276
: 277
: 278
: 279
: 280
: 281
: 282
: 283
: 284
: 285
: 286
: 287
: 288
: 289
: 290
: 291
: 292
: 293
: 294
: 295
: 296
: 297
: 298
: 299
: 300
: 301
: 302
: 303
: 304
: 305
: 306
: 307
: 308
: 309
: 310
: 311
: 312
: 313
: 314
: 315
: 316
: 317
: 318
: 319
: 320
: 321
: 322
: 323
: 324
: 325
: 326
: 327
: 328
: 329
: 330
: 331
: 332
: 333
: 334
: 335
: 336
: 337
: 338
: 339
: 340
: 341
: 342
: 343
: 344
: 345
: 346
: 347
: 348
: 349
: 350
: 351
: 352
: 353
: 354
: 355
: 356
: 357
: 358
: 359
: 360
: 361
: 362
: 363
: 364
: 365
: 366
: 367
: 368
: 369
: 370
: 371
: 372
: 373
: 374
: 375
: 376
: 377
: 378
: 379
: 380
: 381
: 382
: 383
: 384
: 385
: 386
: 387
: 388
: 389
: 390
: 391
: 392
: 393
: 394
: 395
: 396
: 397
: 398
: 399
: 400
: 401
: 402
: 403
: 404
: 405
: 406
: 407
: 408
: 409
: 410
: 411
: 412
: 413
: 414
: 415
: 416
: 417
: 418
: 419
: 420
: 421
: 422
: 423
: 424
: 425
: 426
: 427
: 428
: 429
: 430
: 431
: 432
: 433
: 434
: 435
: 436
: 437
: 438
: 439
: 440
: 441
: 442
: 443
: 444
: 445
: 446
: 447
: 448
: 449
: 450
: 451
: 452
: 453
: 454
: 455
: 456
: 457
: 458
: 459
: 460
: 461
: 462
: 463
: 464
: 465
: 466
: 467
: 468
: 469
: 470
: 471
: 472
: 473
: 474
: 475
: 476
: 477
: 478
: 479
: 480
: 481
: 482
: 483
: 484
: 485
: 486
: 487
: 488
: 489
: 490
: 491
: 492
: 493
: 494
: 495
: 496
: 497
: 498
: 499
: 500
: 501
: 502
: 503
: 504
: 505
: 506
: 507
: 508
: 509
: 510
: 511
: 512
: 513
: 514
: 515
: 516
: 517
: 518
: 519
: 520
: 521
: 522
: 523
: 524
: 525
: 526
: 527
: 528
: 529
: 530
: 531
: 532
: 533
: 534
: 535
: 536
: 537
: 538
: 539
: 540
: 541
: 542
: 543
: 544
: 545
: 546
: 547
: 548
: 549
: 550
: 551
: 552
: 553
: 554
: 555
: 556
: 557
: 558
: 559
: 560
: 561
: 562
: 563
: 564
: 565
: 566
: 567
: 568
: 569
: 570
: 571
: 572
: 573
: 574
: 575
: 576
: 577
: 578
: 579
: 580
: 581
: 582
: 583
: 584
: 585
: 586
: 587
: 588
: 589
: 590
: 591
: 592
: 593
: 594
: 595
: 596
: 597
: 598
: 599
: 600
: 601
: 602
: 603
: 604
: 605
: 606
: 607
: 608
: 609
: 610
: 611
: 612
: 613
: 614
: 615
: 616
: 617 2 ! ADDR[FVPSB_TYPE] = FVP_TYPE;
: 618 2 ! Search backwards through freepage queue. Insert this page so that the
: 619 2 highest address is at the end of the queue and all others are sorted.
: 620 2
: 621 2 FREEPAGE = .(FREE_PAGE_HEAD + 4);
: 622 2
: 623 2 WHILE .FREEPAGE NEQA FREE_PAGE_HEAD DO
: 624 2 BEGIN
: 625 2 IF .ADDR GTRA .FREEPAGE THEN EXITLOOP;
: 626 2 FREEPAGE = .FREEPAGE[FVPSL_BACKWARD];
: 627 2 END; ! end of while
: 628 2
: 629 2 ! the previous entry has been found or may have either no entries in queue
: 630 2 or this is the lowest address
: 631 2
: 632 2 NEXTPAGE = .FREEPAGE; ! previous or head of list
: 633 2
: 634 2 ! if not head of list calculate next entry addr
: 635 2
: 636 2
: 637 2 IF .NEXTPAGE NEQA FREE_PAGE_HEAD
: 638 2 THEN NEXTPAGE = .FREEPAGE[FVPSW_SIZE] + .NEXTPAGE;
: 639 2
: 640 2
: 641 2 ! if region being returned is contiguous after a current entry in the list
: 642 2
: 643 2 IF .NEXTPAGE EQLA .ADDR
: 644 2 THEN
: 645 2
: 646 2 ! append the new region to the old entry
: 647 2
: 648 2 FREEPAGE[FVPSW_SIZE] = .FREEPAGE[FVPSW_SIZE] + .ADDR[FVPSW_SIZE]
: 649 2
: 650 2
: 651 2 ELSE
: 652 2
: 653 2 ! if not contiguous put in queue and adjust FREEPAGE pointer
: 654 2
: 655 3 BEGIN
: 656 3 INSQUE(.ADDR, .FREEPAGE);
: 657 3 FREEPAGE = .ADDR;
: 658 2 END;
: 659 2
: 660 2
: 661 2 ! now if entry contiguous with following one, merge them
: 662 2
: 663 2
: 664 2 NEXTPAGE = .FREEPAGE + .FREEPAGE[FVPSW_SIZE];
: 665 2
: 666 2 ! is it contiguous with next entry?
: 667 2
: 668 2 IF .NEXTPAGE EQLA .FREEPAGE[FVPSL_FORWARD]
: 669 2 THEN
: 670 3 BEGIN
: 671 3 ! remove next entry from queue
: 672 3
: 673 3
```

: 293        0674 3        REMQUE(.FREEPAGE[FVPSL\_FORWARD], NEXTPAGE);  
: 294        0675 3        ! inc size of current entry  
: 295        0676 3        FREEPAGE[FVPSW\_SIZE] = .FREEPAGE[FVPSW\_SIZE] + .NEXTPAGE[FVPSW\_SIZE];  
: 296        0677 3        END;  
: 297        0678 3        ! Should we try to contract the P0 virtual address space of the ACP  
: 298        0679 2        IF .CONTRACT  
: 299        0680 2        THEN BEGIN  
: 300        0681 2        ! get highest free area start address  
: 301        0682 2        ENDFREE = .(FREE PAGE\_HEAD + 4);  
: 302        0683 2        NEXTPAGE = .ENDFREE + .ENDFREE[FVPSW\_SIZE] - 1;  
: 303        0684 2        IF .NEXTPAGE EQLA .LAST\_PAGE  
: 304        0685 3        THEN BEGIN  
: 305        0686 3        ! update last\_page and remove last entry from queue  
: 306        0687 3        ! LAST\_PAGE = .ENDFREE - 1;  
: 307        0688 3        REMQUE(.ENDFREE, ENDFREE);  
: 308        0689 3        ! give back the space  
: 309        0690 3        NEXTPAGE = .ENDFREE[FVPSW\_SIZE]/512;  
: 310        0691 3        IF NOT SY\$CNTREG(.NEXTPAGE, 0, EXEC\_MODE, 0)  
: 311        0692 3        THEN BUG\_CHECK(ACPVAFAIL);  
: 312        0693 3        END  
: 313        0694 4        ELSE  
: 314        0695 4        \*\*\*\*\*  
: 315        0696 4        when making changes try to keep the following two CH\$FILLS next to each other  
: 316        0697 4        because BLISS will only generate the code once and branch to it from 2 places  
: 317        0698 4        \*\*\*\*\*  
: 318        0699 4        ! The area return was not on the end on of the Virtual Address  
: 319        0700 4        Space in P0. So zero out the newly return pages, plus all  
: 320        0701 4        pointer, size and type fields of the free pages that where  
: 321        0702 4        appended (beacuse they were contiguous).  
: 322        0703 4        CH\$FILL ( 0, .FREEPAGE[FVPSW\_SIZE] - 12, .FREEPAGE + 12 );  
: 323        0704 4        END  
: 324        0705 4        ELSE  
: 325        0706 4        ! We are not going to try to contract the P0 space. So clean up the  
: 326        0707 4        pages returned. This will zero out the newly return pages, plus all  
: 327        0708 4        pointer, size and type fields of other free pages that were appended.  
: 328        0709 3         
: 329        0710 3         
: 330        0711 3         
: 331        0712 3         
: 332        0713 3         
: 333        0714 3         
: 334        0715 3         
: 335        0716 3         
: 336        0717 3         
: 337        0718 3         
: 338        0719 3         
: 339        0720 3         
: 340        0721 3         
: 341        0722 3         
: 342        0723 3         
: 343        0724 3         
: 344        0725 3         
: 345        0726 2         
: 346        0727 2         
: 347        0728 2         
: 348        0729 2         
: 349        0730 2

```
: 350      0731 2
: 351      0732 2
: 352      0733 2
: 353      0734 1
```

```
! CHSFILL ( 0, .FREEPAGE[FVPSW_SIZE] - 12, .FREEPAGE + 12 );
END:           ! end of routine
```

## .EXTRN SYSSCNTREG, BUGS\_ACPVAFAIL

				.ENTRY	RET FREE PAGE, Save R2,R3,R4,R5,R6	0560
				MOVAB	FREE_PAGE_HEAD+4, R6	0618
				MOVL	ADDR_R1	
				MOVB	#1, 10(R1)	
				MOVL	FREE_PAGE_HEAD+4, FREEPAGE	0623
				MOVAB	FREE_PAGE_HEAD, R2	0625
				CMPL	FREEPAGE, R2	
				BEQL	2\$	
				CMPL	R1, FREEPAGE	0627
				BGTRU	2\$	
				MOVL	4(FREEPAGE), FREEPAGE	0628
				BRB	1\$	0625
				MOVL	FREEPAGE, NEXTPAGE	0634
				MOVAB	FREE_PAGE_HEAD, R2	0638
				CMPL	NEXTPAGE, R2	
				BEQL	3\$	
				MOVZWL	8(FREEPAGE), R2	0639
				ADDL2	R2, NEXTPAGE	
				CMPL	NEXTPAGE, R1	0644
				BNEQ	4\$	
				ADDW2	8(R1), 8(FREEPAGE)	0649
				BRB	5\$	
				INSQUE	(R1), (FREEPAGE)	0656
				MOVL	ADDR, FREEPAGE	0657
				MOVZWL	8(FREEPAGE), NEXTPAGE	0664
				ADDL2	FREEPAGE, NEXTPAGE	
				CMPL	NEXTPAGE, (FREEPAGE)	0668
				BNEQ	6\$	
				REMQUE	80(FREEPAGE), NEXTPAGE	0674
				ADDW2	8(NEXTPAGE), 8(FREEPAGE)	0678
				BLBC	CONTRACT, 7\$	0683
				MOVL	FREE_PAGE_HEAD+4, ENDFREE	0689
				MOVZWL	8(ENDFREE), R2	0690
				CMPL	-1(R2)[ENDFREE], NEXTPAGE	
				BNEQ	NEXTPAGE, LAST_PAGE	0692
				REMQUE	7\$	
				MOVAB	-1(R1), LAST PAGE	0698
				REMQUE	(ENDFREE), ENDFREE	0699
				MOVZWL	8(ENDFREE), NEXTPAGE	0703
				DIVL2	#512, NEXTPAGE	
				MOVQ	#1 -(SP)	0704
				CLRL	-(SP)	
				PUSHL	NEXTPAGE	
				CALLS	#4, @SYSSCNTREG	
				BLBS	R0, 8\$	0706
				BUGW		
				.WORD	<BUGS_ACPVAFAIL!4>	0692
				RET		

51        00        51        08     A0    3C 000A2 7\$:     MOVZWL 8(FREEPAGE), R1  
                   51        0C    C2 000A6     SUBL2 #12, R1  
                   6E        00    2C 000A9     MOVC5 #0, (SP), #0, R1, 12(FREEPAGE)  
                   0C        A0    000AE     RET  
                   04        000B0 8\$:     RET

: 0732  
: 0734

: Routine Size: 177 bytes,    Routine Base: \$CODE\$ + 007C

: 354        0735 1  
: 355        0736 1 END  
: 356        0737 1  
: 357        0738 0 ELUDOM

## PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	301	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

## Library Statistics

File	Total	Symbols	Pages Mapped	Processing Time
\$_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	5	0	00:01.9

## COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:\$FREEPG/OBJ=OBJ\$:\$FREEPG MSRC\$:\$FREEPG/UPDATE=(ENH\$:\$FREEPG)

: Size:        301 code + 0 data bytes  
: Run Time:    00:10.4  
: Elapsed Time: 00:30.5  
: Lines/CPU Min: 4265  
: Lexemes/CPU-Min: 18635  
: Memory Used: 110 pages  
: Compilation Complete

0254 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

