```
FFFFFFFFFFFFFFF      111          111     XXX          XXX
FFFFFFFFFFFFFFF      111          111     XXX          XXX
FFFFFFFFFFFFFFF      111          111     XXX          XXX
FFF               111111       111111     XXX          XXX
FFF               111111       111111     XXX          XXX
FFF               111111       111111     XXX          XXX
FFF                  111          111        XXX   XXX
FFF                  111          111        XXX   XXX
FFF                  111          111        XXX   XXX
FFFFFFFF.FFF         111          111           XXX
FFFFFFFFFFFF         111          111           XXX
FFFFFFFFFFFF         111          111           XXX
FFF                  111          111        XXX   XXX
FFF                  111          111        XXX   XXX
FFF                  111          111        XXX   XXX
FFF                  111          111     XXX          XXX
FFF                  111          111     XXX          XXX
FFF                  111          111     XXX          XXX
FFF               111111111    111111111  XXX          XXX
FFF               111111111    111111111  XXX          XXX
FFF               111111111    111111111  XXX          XXX
```

```
_$25
Symb
----
IOC$
IO_C
IO_C
IO_D
IO_F
IO_S
KILL

KILL
KILL
LB_E
LB_D
LB_F
LB_H
LB_L
LOCA
LOCA
LOCK

LOCK
LOCK
LOCK
LOC_
LOC_
L_CC
L_CC
L_DA
L_DA
MAIN
MAKE
MAKE
MAKE
MAKE
MAKE
MAKE

MAKE
MAK
MAP_
MAP_

MAP_
MARK
MARK
MARK
MARK
MARK
```

FILEID**FILESERV

```
FFFFFFFFFF    IIIIII    LL          EEEEEEEEEE    SSSSSSSS  EEEEEEEEEE  RRRRRRRR    VV      VV
FFFFFFFFFF    IIIIII    LL          EEEEEEEEEE    SSSSSSSS  EEEEEEEEEE  RRRRRRRR    VV      VV
FF              II      LL          EE          SS          EE          RR      RR  VV      VV
FF              II      LL          EE          SS          EE          RR      RR  VV      VV
FF              II      LL          EE          SS          EE          RR      RR  VV      VV
FF              II      LL          EE          SS          EE          RR      RR  VV      VV
FFFFFFFF        II      LL          EEEEEEEE      SSSSSS    EEEEEEEE    RRRRRRRR    VV      VV
FFFFFFFF        II      LL          EEEEEEEE      SSSSSS    EEEEEEEE    RRRRRRRR    VV      VV
FF              II      LL          EE                  SS  EE          RR  RR      VV      VV
FF              II      LL          EE                  SS  EE          RR  RR      VV      VV
FF              II      LL          EE                  SS  EE          RR    RR      VV  VV        ....
FF              II      LL          EE                  SS  EE          RR    RR      VV  VV        ....
FF            IIIIII    LLLLLLLLLL  EEEEEEEEEE  SSSSSSSS    EEEEEEEEEE  RR      RR      VV          ....
FF            IIIIII    LLLLLLLLLL  EEEEEEEEEE  SSSSSSSS    EEEEEEEEEE  RR      RR      VV          ....

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

FILE
V04-
;
;
; S
; Ru
; El
; Li
; Le
; Me
; Co

```
;     1          0001  0 MODULE FILESERV (       %TITLE  'File System Cache Flush Server'
;     2          0002  0                   MAIN = CACHE_SERVER,
;     3          0003  0                   IDENT = 'V04-000'
;     4          0004  0                   ) =
;     5          0005  1 BEGIN
;     6          0006  1
;     7          0007  1 !
;     8          0008  1 !*****************************************************************************
;     9          0009  1 !*                                                                           *
;    10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    12          0012  1 !*  ALL RIGHTS RESERVED.                                                      *
;    13          0013  1 !*                                                                           *
;    14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    19          0019  1 !*  TRANSFERRED.                                                              *
;    20          0020  1 !*                                                                           *
;    21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    23          0023  1 !*  CORPORATION.                                                              *
;    24          0024  1 !*                                                                           *
;    25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    27          0027  1 !*                                                                           *
;    28          0028  1 !*                                                                           *
;    29          0029  1 !*****************************************************************************
;    30          0030  1 !
;    31          0031  1
;    32          0032  1 !++
;    33          0033  1 ! FACILITY:  VAX/VMS Cluster File System
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !
;    37          0037  1 !       This module is the process responsible for flushing file system
;    38          0038  1 !       caches when requested by other nodes in the cluster. It receives
;    39          0039  1 !       flush requests as kernel mode AST's queued by the swapper process.
;    40          0040  1 !
;    41          0041  1 ! ENVIRONMENT:
;    42          0042  1 !
;    43          0043  1 !       VAX/VMS operating system running as a member of a cluster;
;    44          0044  1 !       kernel mode and file system data structures.
;    45          0045  1 !
;    46          0046  1 !--
;    47          0047  1 !
;    48          0048  1 ! AUTHOR:  Andrew C. Goldstein, CREATION DATE:  17-Jul-1984  19:35
;    49          0049  1 !
;    50          0050  1 ! MODIFIED BY:
;    51          0051  1 !
;    52          0052  1 !       V03-001 ACG0438         Andrew C. Goldstein,    4-Aug-1984  20:43
;    53          0053  1 !               Fix width of cache type field; fix args in call
;    54          0054  1 !               to LIB$FREE_VM.
;    55          0055  1 !
;    56          0056  1 !**
;    57          0057  1
```

```
;    58          0058  1
;    59          0059  1 LIBRARY 'SYS$LIBRARY:LIB';
;    60          0060  1
;    61          0061  1 FORWARD ROUTINE
;    62          0062  1         CACHE_SERVER,                   ! main routine
;    63          0063  1         INITIALIZATION,                 ! kernel mode initialization
;    64          0064  1         GET_REQUEST     : NOVALUE,      ! request handling routine
;    65          0065  1         START_REQUEST   : NOVALUE;      ! start next request
```

```
;    67          0066  1 !
;    68          0067  1 ! Own Storage
;    69          0068  1 !
;    70          0069  1
;    71          0070  1 OWN
;    72          0071  1                                         ! work request queue head
;    73          0072  1         WORK_QUEUE      : VECTOR [2] INITIAL (WORK_QUEUE, WORK_QUEUE),
;    74          0073  1         BUSY,                           ! flag indicating we're busy
;    75          0074  1         CHANNEL,                        ! channel for file system calls
;    76          0075  1         CCB             : REF $BBLOCK,  ! address of channel CCB
;    77          0076  1                                         ! FIB for cache flush calls
;    78          0077  1         IO_STATUS       : VECTOR [4, WORD], ! I/O status block
;    79          0078  1         FIB             : $BBLOCK [FIB$C_LENGTH]
;    80          0079  1                           PRESET ([FIB$W_CNTRLFUNC] = FIB$C_FLUSH_CACHE),
;    81          0080  1                                         ! descriptor for above
;    82          0081  1         FIB_DESC        : VECTOR [2] INITIAL (FIB$C_LENGTH, FIB);
;    83          0082  1
;    84          0083  1 !
;    85          0084  1 ! Structure of queue entry
;    86          0085  1 !
;    87          0086  1
;    88          0087  1 MACRO
;    89          0088  1         QE_FLINK        = 0, 0, 32, 0 %,
;    90          0089  1         QE_BLINK        = 4, 0, 32, 0 %,
;    91          0090  1         QE_UCB          = 8, 0, 32, 0 %;
;    92          0091  1
;    93          0092  1 LITERAL
;    94          0093  1         QUEUE_SIZE      = 12;
;    95          0094  1
;    96          0095  1 !
;    97          0096  1 ! Structure of UCB / cache ID parameter
;    98          0097  1 !
;    99          0098  1
;   100          0099  1 MACRO
;   101          0100  1         CACHE_ID        = 0, 0, 3, 0 %,
;   102          0101  1         UCB_ADDRESS     = 0, 3, 29,0 %;
;   103          0102  1
;   104          0103  1 !
;   105          0104  1 ! Macro to generate a bug check
;   106          0105  1 !
;   107          0106  1
;   108          0107  1 MACRO
;   109        M 0108  1         BUG_CHECK (CODE, MESSAGE) =
;   110        M 0109  1                 BEGIN
;   111        M 0110  1                 BUILTIN BUGW;
;   112        M 0111  1                 EXTERNAL LITERAL %NAME ('BUG$_', CODE);
;   113        M 0112  1                 BUGW (%NAME ('BUG$_', CODE) OR 4)
;   114        M 0113  1                 END
;   115          0114  1                 %;
```

```
;  117          0115  1 GLOBAL ROUTINE CACHE_SERVER =
;  118          0116  1
;  119          0117  1 !++
;  120          0118  1 !
;  121          0119  1 ! FUNCTIONAL DESCRIPTION:
;  122          0120  1 !
;  123          0121  1 !       This is the main program and entry point of the cache server.
;  124          0122  1 !       all it does is to dive into kernel mode to accomplish initialization.
;  125          0123  1 !
;  126          0124  1 ! CALLING SEQUENCE:
;  127          0125  1 !       CACHE_SERVER ()
;  128          0126  1 !
;  129          0127  1 ! INPUT PARAMETERS:
;  130          0128  1 !       NONE
;  131          0129  1 !
;  132          0130  1 ! IMPLICIT INPUTS:
;  133          0131  1 !       NONE
;  134          0132  1 !
;  135          0133  1 ! OUTPUT PARAMETERS:
;  136          0134  1 !       NONE
;  137          0135  1 !
;  138          0136  1 ! IMPLICIT OUTPUTS:
;  139          0137  1 !       NONE
;  140          0138  1 !
;  141          0139  1 ! ROUTINE VALUE:
;  142          0140  1 !       System status code if initialization error
;  143          0141  1 !
;  144          0142  1 ! SIDE EFFECTS:
;  145          0143  1 !       Cache server process started
;  146          0144  1 !
;  147          0145  1 !--
;  148          0146  1
;  149          0147  2 BEGIN
;  150          0148  2
;  151          0149  3 $CMKRNL (ROUTIN = INITIALIZATION)
;  152          0150  3
;  153          0151  1 END;                                    ! End of routine CACHE_SERVER


                                                        .TITLE  FILESERV File System Cache Flush Server
                                                        .IDENT  \V04-000\

                                                        .PSECT  $OWN$,NOEXE,2

                                00000000' 00000000' 00000 WORK_QUEUE:
                                                        .ADDRESS WORK_QUEUE, WORK_QUEUE                 ;
                                                  00008 BUSY:   .BLKB   4
                                                  0000C CHANNEL:.BLKB   4
                                                  00010 CCB:    .BLKB   4
                                                  00014 IO_STATUS:
                                                        .BLKB   8
                                                    00# 0001C FIB:    .BYTE   0[22]                   ;
                                                   0012 00032         .WORD   18                      ;
                                                        00034         .BLKB   40
                                                00000040 0005C FIB_DESC:
                                                        .LONG   64                                      ;
                                                00000000' 00060         .ADDRESS FIB                    ;
```

```
                                                        .EXTRN  SYS$CMKRNL

                                                        .PSECT  $CODE$,NOWRT,2

                                        0000 00000      .ENTRY  CACHE_SERVER, Save nothing                ; 0115
                                     7E   D4 00002      CLRL    -(SP)                                    ; 0149
                                0000V CF   9F 00004      PUSHAB  INITIALIZATION                           ;
                    00000000G   00     02   FB 00008    CALLS   #2, SYS$CMKRNL                           ;
                                           04 0000F      RET                                              ; 0151

; Routine Size:  16 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  155     0152  1 ROUTINE INITIALIZATION =
;  156     0153  1
;  157     0154  1 !++
;  158     0155  1 !
;  159     0156  1 ! FUNCTIONAL DESCRIPTION:
;  160     0157  1 !
;  161     0158  1 !       This routine initializes the cache server process. This consists
;  162     0159  1 !       simply of writing the process' PID into system common so the
;  163     0160  1 !       process can be found by the swapper. This routine executes in
;  164     0161  1 !       kernel mode.
;  165     0162  1 !
;  166     0163  1 ! CALLING SEQUENCE:
;  167     0164  1 !       INITIALIZATION ()
;  168     0165  1 !
;  169     0166  1 ! INPUT PARAMETERS:
;  170     0167  1 !       NONE
;  171     0168  1 !
;  172     0169  1 ! IMPLICIT INPUTS:
;  173     0170  1 !       NONE
;  174     0171  1 !
;  175     0172  1 ! OUTPUT PARAMETERS:
;  176     0173  1 !       NONE
;  177     0174  1 !
;  178     0175  1 ! IMPLICIT OUTPUTS:
;  179     0176  1 !       XQP$GL_FILESERV_ENTRY: receives entry point for requests
;  180     0177  1 !       XQP$GL_FILESERVER: receives PID of this process
;  181     0178  1 !
;  182     0179  1 ! ROUTINE VALUE:
;  183     0180  1 !       (Does not return)
;  184     0181  1 !
;  185     0182  1 ! SIDE EFFECTS:
;  186     0183  1 !       Cache server process started
;  187     0184  1 !
;  188     0185  1 !--
;  189     0186  1
;  190     0187  2 BEGIN
;  191     0188  2
;  192     0189  2 LINKAGE
;  193     0190  2         L_FFCHAN        = JSB (; REGISTER = 1, REGISTER = 2)
;  194     0191  2                         : NOTUSED (3, 4, 5, 6, 7, 8, 9, 10, 11);
;  195     0192  2
;  196     0193  2 LOCAL
;  197     0194  2         STATUS;                         ! system status return
;  198     0195  2
;  199     0196  2 EXTERNAL
;  200     0197  2         CTL$GL_PCB      : REF $BBLOCK ADDRESSING_MODE (GENERAL),
;  201     0198  2                                         ! address of process PID
;  202     0199  2         XQP$GL_FILESERV_ENTRY : ADDRESSING_MODE (GENERAL),
;  203     0200  2         XQP$GL_FILESERVER : ADDRESSING_MODE (GENERAL);
;  204     0201  2                                         ! system cell to store PID
;  205     0202  2
;  206     0203  2 EXTERNAL ROUTINE
;  207     0204  2         IOC$FFCHAN      : L_FFCHAN ADDRESSING_MODE (GENERAL);
;  208     0205  2                                         ! find available I/O channel
;  209     0206  2
;  210     0207  2
;  211     0208  2 ! Set up a channel on which to do I/O.
```

```
;  212          0209  2 !
;  213          0210  2
;  214          0211  2 STATUS = IOC$FFCHAN (; CHANNEL, CCB);
;  215          0212  2 IF NOT .STATUS THEN RETURN .STATUS;
;  216          0213  2 CCB[CCB$B_AMOD] = PSL$C_KERNEL + 1;
;  217          0214  2
;  218          0215  2 ! Set up our PID in system space and then wait for work to roll in.
;  219          0216  2 !
;  220          0217  2
;  221          0218  2 XQP$GL_FILESERV_ENTRY = GET_REQUEST;
;  222          0219  2 XQP$GL_FILESERVER = .CTL$GL_PCB[PCB$L_PID];
;  223          0220  2
;  224          0221  2 WHILE 1
;  225          0222  2 DO $HIBER;
;  226          0223  2
;  227          0224  2 1
;  228          0225  1 END;                                    ! End of routine INITIALIZATION


                                                        .EXTRN  CTL$GL_PCB, XQP$GL_FILESERV_ENTRY
                                                        .EXTRN  XQP$GL_FILESERVER
                                                        .EXTRN  IOC$FFCHAN, SYS$HIBER

                                        0004 00000 INITIALIZATION:
                                                        .WORD   Save R2                                 ; 0152
                          00000000G   00  16 00002      JSB     IOC$FFCHAN                              ; 0211
                0000'  CF             51  7D 00008      MOVQ    R1, CHANNEL                             ;
                       2A             50  E9 0000D      BLBC    STATUS, 2$                              ; 0212
                       50      0000'  CF  D0 00010      MOVL    CCB, R0                                 ; 0213
                    09 A0             01  90 00015      MOVB    #1, 9(R0)                               ;
          00000000G   00       0000V  CF  9E 00019      MOVAB   GET_REQUEST, XQP$GL_FILESERV_ENTRY      ; 0218
                       50 00000000G   00  D0 00022      MOVL    CTL$GL_PCB, R0                          ; 0219
          00000000G   00         60   A0  D0 00029      MOVL    96(R0), XQP:GL_FILESERVER               ;
          00000000G   00              00  FB 00031 1$:  CALLS   #0, SYS$HIBER                           ; 0222
                                      F7  11 00038      BRB     1$                                      ;
                                          04 0003A 2$:  RET                                             ; 0225

; Routine Size:  59 bytes,    Routine Base:  $CODE$ + 0010
```

```
;  230     0226  1 ROUTINE GET_REQUEST (UCB) : NOVALUE =
;  231     0227  1
;  232     0228  1 !++
;  233     0229  1 !
;  234     0230  1 ! FUNCTIONAL DESCRIPTION:
;  235     0231  1 !
;  236     0232  1 !       This routine is entered as a kernel mode AST when a request to
;  237     0233  1 !       flush a cache occurs. We execute the request immediately if
;  238     0234  1 !       we are not currently busy, or queue it internally if we are.
;  239     0235  1 !
;  240     0236  1 ! CALLING SEQUENCE:
;  241     0237  1 !       GET_REQUEST (UCB)
;  242     0238  1 !
;  243     0239  1 ! INPUT PARAMETERS:
;  244     0240  1 !       UCB: address of UCB to operate on, with request type
;  245     0241  1 !            encoded into low 4 bits
;  246     0242  1 !
;  247     0243  1 ! IMPLICIT INPUTS:
;  248     0244  1 !       NONE
;  249     0245  1 !
;  250     0246  1 ! OUTPUT PARAMETERS:
;  251     0247  1 !       NONE
;  252     0248  1 !
;  253     0249  1 ! IMPLICIT OUTPUTS:
;  254     0250  1 !       NONE
;  255     0251  1 !
;  256     0252  1 ! ROUTINE VALUE:
;  257     0253  1 !       NONE
;  258     0254  1 !
;  259     0255  1 ! SIDE EFFECTS:
;  260     0256  1 !       Request initiated or queued
;  261     0257  1 !
;  262     0258  1 !--
;  263     0259  1
;  264     0260  2 BEGIN
;  265     0261  2
;  266     0262  2 BUILTIN
;  267     0263  2         INSQUE;
;  268     0264  2
;  269     0265  2 LOCAL
;  270     0266  2         ENTRY           : REF $BBLOCK;  ! address of allocated queue entry
;  271     0267  2
;  272     0268  2 EXTERNAL ROUTINE
;  273     0269  2         LIB$GET_VM;                     ! allocate virtual memory
;  274     0270  2
;  275     0271  2
;  276     0272  2 IF NOT .BUSY
;  277     0273  2 THEN
;  278     0274  2     START_REQUEST (.UCB)
;  279     0275  2 ELSE
;  280     0276  3     BEGIN
;  281     0277  3     IF NOT LIB$GET_VM (%REF (QUEUE_SIZE), ENTRY)
;  282     0278  3     THEN BUG_CHECK (XQPERR, 'Failed to allocate queue entry');
;  283     0279  3     ENTRY[QE_UCB] = .UCB;
;  284     0280  3     INSQUE (.ENTRY, .WORK_QUEUE[1]);
;  285     0281  2     END;
;  286     0282  2
```

```
;  287          0283  1 END;                                  ! End of routine GET_REQUEST


                                                             .EXTRN  LIB$GET_VM, BUG$_XQPERR

                                          0000 00000 GET_REQUEST:
                                                             .WORD   Save nothing                        ; 0226
                                   5E              08  C2 00002  SUBL2   #8, SP                        ; 0227
                                   09      0000'   CF  E8 00005  BLBS    BUSY, 1$                      ; 0272
                                           04      AC  DD 00009  PUSHL   UCB                           ; 0274
                             0000V CF              01  FB 0000D  CALLS   #1, START_REQUEST
                                                   04  00012  RET
                                           04      AE  9F 00013 1$:    PUSHAB  ENTRY                     ; 0277
                               04  AE              0C  D0 00016  MOVL    #12, 4(SP)
                                           04      AE  9F 0001A  PUSHAB  4(SP)
                             0000G CF              02  FB 0001D  CALLS   #2, LIB$GET_VM
                                   04              50  E8 00022  BLBS    R0, 2$
                                                 FEFF  00025  BUGW                                     ; 0278
                                                0000* 00027  .WORD   <BUG$_XQPERR!4>
                                   50      04      AE  D0 00029 2$:    MOVL    ENTRY, R0               ; 0279
                                08 A0      04      AC  D0 0002D  MOVL    UCB, 8(R0)                  ; 0280
                             0000' DF              60  0E 00032  INSQUE  (R0), @WORK_QUEUE+4           ; 0283
                                                   04  00037  RET

; Routine Size:  56 bytes,    Routine Base:  $CODE$ + 004B
```

```
;  289     0284  1 ROUTINE START_REQUEST (UCB_ARG) : NOVALUE =
;  290     0285  1
;  291     0286  1 !++
;  292     0287  1 !
;  293     0288  1 ! FUNCTIONAL DESCRIPTION:
;  294     0289  1 !
;  295     0290  1 !       This routine actually initiates a cache flush request. It is also
;  296     0291  1 !       entered as the completion AST of a flush operation to check if
;  297     0292  1 !       another one is pending.
;  298     0293  1 !
;  299     0294  1 ! CALLING SEQUENCE:
;  300     0295  1 !       START_REQUEST (UCB_ARG)
;  301     0296  1 !
;  302     0297  1 ! INPUT PARAMETERS:
;  303     0298  1 !       UCB_ARG: address of UCB to operate on
;  304     0299  1 !                0 to get next entry from the work queue
;  305     0300  1 !
;  306     0301  1 ! IMPLICIT INPUTS:
;  307     0302  1 !       NONE
;  308     0303  1 !
;  309     0304  1 ! OUTPUT PARAMETERS:
;  310     0305  1 !       NONE
;  311     0306  1 !
;  312     0307  1 ! IMPLICIT OUTPUTS:
;  313     0308  1 !       NONE
;  314     0309  1 !
;  315     0310  1 ! ROUTINE VALUE:
;  316     0311  1 !       NONE
;  317     0312  1 !
;  318     0313  1 ! SIDE EFFECTS:
;  319     0314  1 !       Next queue entry, if any, initiated
;  320     0315  1 !
;  321     0316  1 !--
;  322     0317  1
;  323     0318  2 BEGIN
;  324     0319  2
;  325     0320  2 BUILTIN
;  326     0321  2         REMQUE;
;  327     0322  2
;  328     0323  2 LOCAL
;  329     0324  2         STATUS,                         ! system service status
;  330     0325  2         ENTRY           : REF $BBLOCK,  ! current work queue entry
;  331     0326  2         UCB             : $BBLOCK [4];  ! local copy of UCB address
;  332     0327  2
;  333     0328  2 EXTERNAL ROUTINE
;  334     0329  2         LIB$FREE_VM;                    ! return virtual memory
;  335     0330  2
;  336     0331  2
;  337     0332  2 ! If no UCB address was given, we just completed a request. Get the next
;  338     0333  2 ! work queue entry to process. If the work queue is empty, there's
;  339     0334  2 ! nothing to do. We loop until we successfully fire off a flush request.
;  340     0335  2 !
;  341     0336  2
;  342     0337  2 UCB = .UCB_ARG;
;  343     0338  2
;  344     0339  2 WHILE 1 DO
;  345     0340  3     BEGIN
```

```
;  346          0341  3     BUSY = 0;
;  347          0342  3     IF .UCB EQL 0
;  348          0343  3     THEN
;  349          0344  4         BEGIN
;  350          0345  4         IF REMQUE (.WORK_QUEUE[0], ENTRY)
;  351          0346  4         THEN RETURN;
;  352          0347  4         UCB = .ENTRY[QE_UCB];
;  353          0348  4         STATUS = LIB$FREE_VM (%REF (QUEUE_SIZE), ENTRY);
;  354          0349  4         IF NOT .STATUS THEN BUG_CHECK (XQPERR, 'Unexpected VM error');
;  355          0350  3         END;
;  356          0351  3
;  357          0352  3 ! Unpack the cache identifier code from the low bits of the UCB
;  358          0353  3 ! address. Set up the channel and fire off the request. Note that
;  359          0354  3 ! the volume and UCB are uninterlocked while the request is queued.
;  360          0355  3 ! This is harmless since (1) disk UCB's never go away and (2) we ignore
;  361          0356  3 ! the appropriate errors if the volume is now dismounted (and possibly
;  362          0357  3 ! a different volume mounted). The worst that happens is that we do
;  363          0358  3 ! an unnecessary flush on some volume.
;  364          0359  3 !
;  365          0360  3
;  366          0361  3     BUSY = 1;
;  367          0362  3     FIB[FIB$L_CNTRLVAL] = .UCB[CACHE_ID];
;  368          0363  3     UCB[CACHE_ID] = 0;
;  369          0364  3     CCB[CCB$L_UCB] = .UCB;
;  370        P 0365  3     STATUS = $QIO  (CHAN   = .CHANNEL,
;  371        P 0366  3                     FUNC   = IO$_ACPCONTROL,
;  372        P 0367  3                     IOSB   = IO_STATUS,
;  373        P 0368  3                     ASTADR = START_REQUEST,
;  374        P 0369  3                     P1     = FIB_DESC
;  375          0370  3                     );
;  376          0371  3     IF .STATUS
;  377          0372  3     THEN EXITLOOP
;  378          0373  3     ELSE IF .STATUS NEQ SS$_WRITLCK
;  379          0374  3         AND .STATUS NEQ SS$_DEVNOTMOUNT
;  380          0375  3         AND .STATUS NEQ SS$_DEVFOREIGN
;  381          0376  3     THEN BUG_CHECK (XQPERR, 'Unexpected QIO service error');
;  382          0377  3     UCB = 0;
;  383          0378  2     END;                                ! end of loop
;  384          0379  2
;  385          0380  1 END;                                    ! End of routine START_REQUEST


                                                        .EXTRN  LIB$FREE_VM, SYS$QIO

                                        000C 00000 START_REQUEST:
                                                        .WORD   Save R2,R3                      ; 0284
                              53    0000'   CF  9E 00002      MOVAB   BUSY, R3                ;
                              5E            08  C2 00007      SUBL2   #8, SP                  ;
                              52      04    AC  D0 0000A      MOVL    UCB_ARG, UCB            ; 0337
                                            63  D4 0000E 1$:  CLRL    BUSY                    ; 0341
                                            52  D5 00010      TSTL    UCB                     ; 0342
                                            25  12 00012      BNEQ    2$                      ;
                          04  AE      F8    B3  0F 00014      REMQUE  @WORK_QUEUE, ENTRY      ; 0345
                                            74  1D 00019      BVS     4$                      ;
                              51      04    AE  D0 0001B      MOVL    ENTRY, R1               ; 0347
                              52      08    A1  D0 0001F      MOVL    8(R1), UCB              ;
```

```
                                   04   AE   9F 00023         PUSHAB  ENTRY                          ; 0348
                 04   AE                 0C   D0 00026         MOVL    #12, 4(SP)                     ;
                                   04   AE   9F 0002A         PUSHAB  4(SP)                          ;
              0000G   CF                 02   FB 0002D         CALLS   #2, LIB$FREE_VM                ;
                      04                 50   E8 00032         BLBS    STATUS, 2$                     ; 0349
                                       FEFF 00035         BUGW                                   ;
                                      0000* 00037         .WORD   <BUG$_XQPERR!4>                ;
                      63                 01   D0 00039 2$:    MOVL    #1, BUSY                       ; 0361
   2C   A3        52  03                 00   EF 0003C         EXTZV   #0, #3, UCB, FIB+24            ; 0362
                      52                 07   8A 00042         BICB2   #7, UCB                        ; 0363
                 08   B3                 52   D0 00045         MOVL    UCB, @CCB                      ; 0364
                                         7E   7C 00049         CLRQ    -(SP)                          ; 0370
                                         7E   7C 0004B         CLRQ    -(SP)                          ;
                                         7E   D4 0004D         CLRL    -(SP)                          ;
                                   54   A3   9F 0004F         PUSHAB  FIB_DESC                       ;
                                         7E   D4 00052         CLRL    -(SP)                          ;
                                   A9   AF   9F 00054         PUSHAB  START_REQUEST                  ;
                                   0C   A3   9F 00057         PUSHAB  IO_STATUS                      ;
                                         38   DD 0005A         PUSHL   #56                            ;
                                   04   A3   DD 0005C         PUSHL   CHANNEL                        ;
                                         7E   D4 0005F         CLRL    -(SP)                          ;
          00000000G   00                 0C   FB 00061         CALLS   #12, SYS$QIO                   ;
                      24                 50   E8 00068         BLBS    STATUS, 4$                     ; 0371
          0000025C    8F                 50   D1 0006B         CMPL    STATUS, #604                   ; 0373
                                         16   13 00072         BEQL    3$                             ;
          0000007C    8F                 50   D1 00074         CMPL    STATUS, #124                   ; 0374
                                         0D   13 0007B         BEQL    3$                             ;
          00000064    8F                 50   D1 0007D         CMPL    STATUS, #100                   ; 0375
                                         04   13 00084         BEQL    3$                             ;
                                       FEFF 00086         BUGW                                   ; 0376
                                      0000* 00088         .WORD   <BUG$_XQPERR!4>                ;
                                         52   D4 0008A 3$:    CLRL    UCB                            ; 0377
                                   FF7F   31 0008C         BRW     1$                             ; 0339
                                              04 0008F 4$:    RET                                    ; 0380

; Routine Size:  144 bytes,    Routine Base:  $CODE$ + 0083


;  386          0381  1
;  387          0382  1 END
;  388          0383  0 ELUDOM



;                                    PSECT SUMMARY
;
;       Name                      Bytes                        Attributes
;
;   $OWN$                           100  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $CODE$                          275  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

```
;                                Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1          18619        18         0        1000        00:01.9
;
;
;
;
;
;                                      COMMAND QUALIFIERS
;
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:FILESERV/OBJ=OBJ$:FILESERV MSRC$:FILESERV/UPDATE=(ENH$:FILESERV)
;
; Size:         275 code + 100 data bytes
; Run Time:        00:08.4
; Elapsed Time:    00:18.4
; Lines/CPU Min:    2732
; Lexemes/CPU-Min: 11935
; Memory Used:  83 pages
; Compilation Complete
```

EXTIDX.LIS

GTLCAT.LIS

GETLOC.LIS

EXTEND.LIS

EXTHDR.LIS

FILUTL.LIS

GETREQ.LIS

EXTCONTIG.LIS

ERASE.LIS

GETTIM.LIS

FIND.LIS

GETFIB.LIS

INIFC2.LIS

INIFCP.LIS

EXTFCB.LIS

FILESERV.LIS

FILESIZE.LIS

GETPTR.LIS