```
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                   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_CO
L_CO
L_DA
L_DA
MAIN
MAKE
MAKE
MAKE
MAKE
MAKE
MAKE

MAKE
MAKE
MAP_
MAP_

MAP_
MARK
MARK
MARK
MARK
MARK

```
DDDDDDDD      IIIIII      SSSSSSSS  PPPPPPPP      AAAAAA    TTTTTTTTTT
DDDDDDDD      IIIIII      SSSSSSSS  PPPPPPPP      AAAAAA    TTTTTTTTTT
DD      DD      II      SS        PP      PP  AA      AA      TT
DD      DD      II      SS        PP      PP  AA      AA      TT
DD      DD      II      SS        PP      PP  AA      AA      TT
DD      DD      II      SS        PP      PP  AA      AA      TT
DD      DD      II        SSSSSS  PPPPPPPP    AA      AA      TT
DD      DD      II        SSSSSS  PPPPPPPP    AA      AA      TT
DD      DD      II              SS  PP        AAAAAAAAAA      TT
DD      DD      II              SS  PP        AAAAAAAAAA      TT
DD      DD      II              SS  PP        AA      AA      TT        ....
DD      DD      II              SS  PP        AA      AA      TT        ....
DDDDDDDD      IIIIII    SSSSSSSS    PP        AA      AA      TT        ....
DDDDDDDD      IIIIII    SSSSSSSS    PP        AA      AA      TT        ....

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
```

```
;     1        0001  0 MODULE DISPAT (
;     2        0002  0                 LANGUAGE (BLISS32),
;     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:  F11ACP Structure Level 2
;    34        0034  1 !
;    35        0035  1 ! ABSTRACT:
;    36        0036  1 !
;    37        0037  1 !       This module is the main routine of FCP. It dequeues a request,
;    38        0038  1 !       executes it, and signals completion to the user.
;    39        0039  1 !
;    40        0040  1 ! ENVIRONMENT:
;    41        0041  1 !
;    42        0042  1 !       STARLET operating system, including privileged system services
;    43        0043  1 !       and internal exec routines.
;    44        0044  1 !
;    45        0045  1 !--
;    46        0046  1 !
;    47        0047  1 !
;    48        0048  1 ! AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  20-Dec-1976  14:33
;    49        0049  1 !
;    50        0050  1 ! MODIFIED BY:
;    51        0051  1 !
;    52        0052  1 !       V03-020 ACG0438         Andrew C. Goldstein,    26-Jul-1984  14:01
;    53        0053  1 !               Handle create-if at dispatcher level for improved generality
;    54        0054  1 !
;    55        0055  1 !       V03-019 ACG0427         Andrew C. Goldstein,    8-May-1984  20:31
;    56        0056  1 !               Optimize checksumming of file header in error cases
;    57        0057  1 !
```

```
;    58          0058  1 ;        V03-018 ACG0427         Andrew C. Goldstein,    8-May-1984  11:23
;    59          0059  1 ;                Finish security auditing. Restructure the saved audit
;    60          0060  1 ;                info to save space.
;    61          0061  1 ;
;    62          0062  1 ;        V03-017 CDS0014         Christian D. Saether    10-Apr-1984
;    63          0063  1 ;                Increase number of error retries.
;    64          0064  1 ;
;    65          0065  1 ;        V03-016 RSH0134         R. Scott Hanna          04-APR-1984
;    66          0066  1 ;                Add the PERFORM_AUDIT routine.
;    67          0067  1 ;
;    68          0068  1 ;        V03-015 CDS0013         Christian D. Saether    22-Feb-1984
;    69          0069  1 ;                Call GET_REQD_BFR_CREDITS before starting request.
;    70          0070  1 ;                Call RETURN_CREDITS after running down buffers and locks.
;    71          0071  1 ;
;    72          0072  1 ;        V03-014 CDS0012         Christian D. Saether    29-Dec-1983
;    73          0073  1 ;                Use L_NORM linkage and BIND_COMMON macro.
;    74          0074  1 ;                The DISPATCH module plays games with the disk
;    75          0075  1 ;                i/o channel now.
;    76          0076  1 ;
;    77          0077  1 ;        V03-013 CDS0011         Christian D. Saether    17-Oct-1983
;    78          0078  1 ;                Let MOUNT and ACPCONTROL functions get by
;    79          0079  1 ;                request block checking.
;    80          0080  1 ;
;    81          0081  1 ;        V03-012 CDS0010         Christian D. Saether    16-Oct-1983
;    82          0082  1 ;                Correctly declare BLOCK_CHECK to be a byte.
;    83          0083  1 ;
;    84          0084  1 ;        V03-011 CDS0009         Christian D. Saether    12-Oct-1983
;    85          0085  1 ;                Add calls to START_REQUEST and FINISH_REQUEST
;    86          0086  1 ;                to synchronize with volume activity blocking.
;    87          0087  1 ;
;    88          0088  1 ;        V03-010 CDS0008         Christian D. Saether    14-Sep-1983
;    89          0089  1 ;                Use the allocation_unlock routine to release
;    90          0090  1 ;                the allocation lock.
;    91          0091  1 ;                Use the RELEASE_SERIAL_LOCK routine to run down locks.
;    92          0092  1 ;                Move the request initialization calls into GET_REQUEST
;    93          0093  1 ;                to avoid a useless init after the last request.
;    94          0094  1 ;
;    95          0095  1 ;        V03-009 CDS0007         Christian D. Saether     2-Sep-1983
;    96          0096  1 ;                Remember the original IO_CHANNEL UCB prior to
;    97          0097  1 ;                the request loop.  V03-008 incorrectly did this
;    98          0098  1 ;                in get_request, and when a second request piled
;    99          0099  1 ;                up in the queue, would "restore" the UCB of the
;   100          0100  1 ;                previous request, rather than that of the orignal
;   101          0101  1 ;                channel assignment.
;   102          0102  1 ;
;   103          0103  1 ;        V03-008 CDS0006         Christian D. Saether    27-Aug-1983
;   104          0104  1 ;                Put back the original IO_CHANNEL UCB rather than
;   105          0105  1 ;                just stuffing the UCB it got when originally assigned.
;   106          0106  1 ;                During process deletion, we may get called after
;   107          0107  1 ;                the channel has been deassigned and assigned to
;   108          0108  1 ;                some other device.
;   109          0109  1 ;
;   110          0110  1 ;        V03-007 CDS0005         Christian D. Saether    23-Jun-1983
;   111          0111  1 ;                Invalidate all buffers after a successful window turn.
;   112          0112  1 ;
;   113          0113  1 ;        V03-006 CDS0004         Christian D. Saether     3-May-1983
;   114          0114  1 ;                Remove volume level interlock.  Unlock_xqp routine
```

```
;  115          0115  1 !                 now releases fid locks instead of volume lock.
;  116          0116  1 !
;  117          0117  1 !       V03-005 CDS0003         C Saether               26-Oct-1982
;  118          0118  1 !               Restore original ucb's to assigned channels before exit.
;  119          0119  1 !
;  120          0120  1 !       V03-004 CDS0002         C Saether                6-Oct-1982
;  121          0121  1 !               Add volume level interlock of XQP activity.
;  122          0122  1 !
;  123          0123  1 !       V03-003 CDS0001         C Saether               30-Jul-1982
;  124          0124  1 !               Make changes from ACP to XQP.
;  125          0125  1 !
;  126          0126  1 !       V03-002 LMP0035         L. Mark Pilant,         28-Jun-1982  14:50
;  127          0127  1 !               Correct problems that caused information messages.
;  128          0128  1 !
;  129          0129  1 !       V03-001 ACG0274         Andrew C. Goldstein,    23-Mar-1982  14:49
;  130          0130  1 !               Use longword displacement
;  131          0131  1 !
;  132          0132  1 !       A0102   ACG0082         Andrew C. Goldstein,    8-Nov-1979  21:42
;  133          0133  1 !               Make error cleanup iterative for new write error handling
;  134          0134  1 !
;  135          0135  1 !       A0101   ACG0044         Andrew C. Goldstein,  15-Jun-1979  11:39
;  136          0136  1 !       Add disk quota support
;  137          0137  1 !
;  138          0138  1 !       A0100   ACG00001        Andrew C. Goldstein,  10-Oct-1978  20:02
;  139          0139  1 !       Previous revision history moved to F11A.REV
;  140          0140  1 !**
;  141          0141  1
;  142          0142  1
;  143          0143  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;  144          0144  1 REQUIRE 'SRC$:FCPDEF.B32';
;  145          1135  1
;  146          1136  1 !
;  147          1137  1 ! Establish the max and min function codes for the function dispatch.
;  148          1138  1 !
;  149          1139  1
;  150          1140  1 LITERAL
;  151          1141  1         LOW_FUNCTION    = MINU (
;  152          1142  1                 IO$_ACCESS,
;  153          1143  1                 IO$_CREATE,
;  154          1144  1                 IO$_DEACCESS,
;  155          1145  1                 IO$_DELETE,
;  156          1146  1                 IO$_MODIFY
;  157          1147  1                 ),
;  158          1148  1
;  159          1149  1         HIGH_FUNCTION   = MAXU (
;  160          1150  1                 IO$_CREATE,
;  161          1151  1                 IO$_DEACCESS,
;  162          1152  1                 IO$_DELETE,
;  163          1153  1                 IO$_MODIFY
;  164          1154  1                 );
;  165          1155  1
;  166          1156  1 FORWARD ROUTINE
;  167          1157  1         DISPATCHER      : L_NORM NOVALUE,
;  168          1158  1         UNLOCK_XQP      : L_NORM NOVALUE,
;  169          1159  1         MAIN_HANDLER    : L_NORM NOVALUE,
;  170          1160  1         ZERO_ON_ERROR,
;  171          1161  1         PERFORM_AUDIT   : L_NORM NOVALUE,
```

```
;  172          1162  1          WRITE_AUDIT      : L_NORM NOVALUE;
```

```
;  174          1163  1 GLOBAL ROUTINE DISPATCHER : L_NORM NOVALUE =
;  175          1164  1
;  176          1165  1 !++
;  177          1166  1 !
;  178          1167  1 ! FUNCTIONAL DESCRIPTION:
;  179          1168  1 !
;  180          1169  1 !       This routine is the main routine of FCP. It dequeues a request,
;  181          1170  1 !       executes it, and signals completion to the user.
;  182          1171  1 !
;  183          1172  1 ! CALLING SEQUENCE:
;  184          1173  1 !       DISPATCHER ()
;  185          1174  1 !
;  186          1175  1 ! INPUT PARAMETERS:
;  187          1176  1 !       NONE
;  188          1177  1 !
;  189          1178  1 ! IMPLICIT INPUTS:
;  190          1179  1 !       NONE
;  191          1180  1 !
;  192          1181  1 ! OUTPUT PARAMETERS:
;  193          1182  1 !       NONE
;  194          1183  1 !
;  195          1184  1 ! IMPLICIT OUTPUTS:
;  196          1185  1 !       NONE
;  197          1186  1 !
;  198          1187  1 ! ROUTINE VALUE:
;  199          1188  1 !       NONE
;  200          1189  1 !
;  201          1190  1 ! SIDE EFFECTS:
;  202          1191  1 !       FCP functions executed
;  203          1192  1 !
;  204          1193  1 !--
;  205          1194  1
;  206          1195  2 BEGIN
;  207          1196  2
;  208          1197  2 LABEL
;  209          1198  2         NORMAL_FUNC;                    ! block of normal function execution
;  210          1199  2
;  211          1200  2 LOCAL
;  212          1201  2         FUNCTION,                       ! function being executed
;  213          1202  2         STATUS;                         ! status return of function routine
;  214          1203  2
;  215          1204  2 BIND_COMMON;
;  216          1205  2
;  217          1206  2 EXTERNAL ROUTINE
;  218          1207  2         GET_REQD_BFR_CREDITS : L_NORM,
;  219          1208  2         START_REQUEST   : L_JSB,
;  220          1209  2         FINISH_REQUEST  : L_JSB,
;  221          1210  2         PMS_END         : L_NORM,       ! end performance metering
;  222          1211  2         GET_REQUEST     : L_NORM,       ! get next I/O request
;  223          1212  2         READ_WRITEVB    : L_NORM,       ! process read/write virtual
;  224          1213  2         ACCESS          : L_NORM,       ! ACCESS function routine
;  225          1214  2         CREATE          : L_NORM,       ! CREATE function routine
;  226          1215  2         DEACCESS        : L_NORM,       ! DEACCESS function routine
;  227          1216  2         DELETE          : L_NORM,       ! DELETE function routine
;  228          1217  2         MODIFY          : L_NORM,       ! MODIFY function routine
;  229          1218  2         ACPCONTROL      : L_NORM,       ! ACPCONTROL function routine
;  230          1219  2         MOUNT           : L_NORM,       ! MOUNT function routine
```

```
;  231     1220  2          CHECKSUM        : L_NORM,       ! checksum a file header
;  232     1221  2          ERR_CLEANUP     : L_NORM,       ! error cleanup routine
;  233     1222  2          CLEANUP         : L_NORM,       ! general cleanup routine
;  234     1223  2          IO_DONE;                        ! I/O completion processing
;  235     1224  2
;  236     1225  2
;  237     1226  2 ! Get the next request, and process it. If
;  238     1227  2 ! the request fails, call the error cleanup before returning
;  239     1228  2 ! completion.  When the last request is dequeued, return.
;  240     1229  2 !
;  241     1230  2
;  242     1231  2 !       ENABLE MAIN_HANDLER;
;  243     1232  3 BEGIN
;  244     1233  3 BUILTIN FP;
;  245     1234  3 .FP = MAIN_HANDLER;
;  246     1235  2 END;
;  247     1236  2
;  248     1237  2 WHILE 1 DO
;  249     1238  2
;  250     1239  2 NORMAL_FUNC:
;  251     1240  3     BEGIN
;  252     1241  3
;  253     1242  3     IF (IO_PACKET = GET_REQUEST()) EQL 0
;  254     1243  3     THEN
;  255     1244  3
;  256     1245  3 ! No more packets.  Exit.
;  257     1246  3 !
;  258     1247  3
;  259     1248  3         RETURN;
;  260     1249  3
;  261     1250  3     FUNCTION = .IO_PACKET[IRP$V_FCODE];
;  262     1251  3
;  263     1252  3     STATUS =
;  264     1253  3
;  265     1254  4         BEGIN
;  266     1255  4
;  267     1256  4         GET_REQD_BFR_CREDITS ();
;  268     1257  4
;  269     1258  4         SELECTONEU .FUNCTION OF
;  270     1259  4             SET
;  271     1260  4
;  272     1261  4             [IO$_READPBLK, IO$_WRITEPBLK]:
;  273     1262  4                 IF READ_WRITEVB ()
;  274     1263  4                 THEN
;  275     1264  4                     LEAVE NORMAL_FUNC
;  276     1265  4                 ELSE 0;
;  277     1266  4
;  278     1267  4             [IO$_ACPCONTROL]: ACPCONTROL ();
;  279     1268  4             [IO$_MOUNT]:        MOUNT ();
;  280     1269  4
;  281     1270  4             [OTHERWISE]:
;  282     1271  4
;  283     1272  5             BEGIN
;  284     1273  5
;  285     1274  5             IF .BLOCK_LOCKID EQL 0
;  286     1275  5             THEN
;  287     1276  6                 BEGIN
```

```
;  288     1277  6                START_REQUEST ();
;  289     1278  6                BLOCK_CHECK = 1;
;  290     1279  5                END,
;  291     1280  5
;  292     1281  5            CASE .FUNCTION FROM LOW_FUNCTION TO HIGH_FUNCTION OF
;  293     1282  5                SET
;  294     1283  6                [IO$_ACCESS]:   BEGIN
;  295     1284  6                                LOCAL STATUS;
;  296     1285  6                                STATUS = ACCESS ();
;  297     1286  6                                IF .STATUS EQL SS$_NOSUCHFILE
;  298     1287  6                                AND .BBLOCK [IO_PACKET[IRP$W_FUNC], IO$V_CREATE]
;  299     1288  6                                THEN
;  300     1289  7                                    BEGIN
;  301     1290  7                                    USER_STATUS = SS$_CREATED;
;  302     1291  7                                    CREATE ()
;  303     1292  7                                    END
;  304     1293  6                                ELSE
;  305     1294  6                                    .STATUS
;  306     1295  5                                END;
;  307     1296  5                [IO$_CREATE]:   CREATE ();
;  308     1297  5                [IO$_DEACCESS]: DEACCESS ();
;  309     1298  5                [IO$_DELETE]:   DELETE ();
;  310     1299  5                [IO$_MODIFY]:   MODIFY ();
;  311     1300  5                [INRANGE]:      (ERR_STATUS (SS$_ILLIOFUNC); 0);
;  312     1301  5                [OUTRANGE]:     (ERR_STATUS (SS$_ILLIOFUNC); 0);
;  313     1302  5                TES
;  314     1303  5            END
;  315     1304  4            TES
;  316     1305  3        END;
;  317     1306  3
;  318     1307  3    IF .AUDIT_COUNT NEQU 0
;  319     1308  3    THEN
;  320     1309  4        BEGIN
;  321     1310  4        IF NOT .STATUS
;  322     1311  4        AND .FILE_HEADER NEQ 0
;  323     1312  4        THEN CHECKSUM (.FILE_HEADER);
;  324     1313  4        PERFORM_AUDIT();
;  325     1314  3        END;
;  326     1315  3
;  327     1316  3    DECR J FROM 2000 TO 1
;  328     1317  3    DO
;  329     1318  4        BEGIN
;  330     1319  4        IF .STATUS THEN IF CLEANUP () THEN EXITLOOP;
;  331     1320  4        STATUS = ERR_CLEANUP ();
;  332     1321  3        END;
;  333     1322  3
;  334     1323  3    UNLOCK_XQP();
;  335     1324  3    PMS_END ();
;  336     1325  3    IO_DONE (.IO_PACKET);
;  337     1326  3
;  338     1327  3    IF .BLOCK_CHECK
;  339     1328  3    THEN
;  340     1329  3        FINISH_REQUEST ();
;  341     1330  3
;  342     1331  2    END;                                    ! end of block NORMAL_FUNC
;  343     1332  2
;  344     1333  1 END;                                       ! end of routine DISPATCHER
```

```
                                                        .TITLE  DISPAT
                                                        .IDENT  \V04-000\

                                                        .EXTRN  GET_REQD_BFR_CREDITS
                                                        .EXTRN  START_REQUEST, FINISH_REQUEST
                                                        .EXTRN  PMS_END, GET_REQUEST
                                                        .EXTRN  READ_WRITEVB, ACCESS
                                                        .EXTRN  CREATE, DEACCESS
                                                        .EXTRN  DELETE, MODIFY, ACPCONTROL
                                                        .EXTRN  MOUNT, CHECKSUM
                                                        .EXTRN  ERR_CLEANUP, CLEANUP
                                                        .EXTRN  IO_DONE

                                                        .PSECT  $CODE$,NOWRT,2

                                        001C 00000      .ENTRY  DISPATCHER, Save R2,R3,R4                       ; 1163
                                    6D  0000V CF  9E 00002  MOVAB   MAIN_HANDLER, (FP)                          ; 1234
                                0000G   CF    00  FB 00007 1$:  CALLS   #0, GET_REQUEST                       ; 1242
                                  90    AA      50  D0 0000C      MOVL    R0, -112(BASE)                        ;
                                                01  12 00010      BNEQ    2$                                    ;
                                                    04 00012      RET                                           ;
                                    50    90    AA  D0 00013 2$:  MOVL    -112(BASE), R0                        ; 1250
                53      20  A0                06  00  EF 00017      EXTZV   #0, #6, 32(R0), FUNCTION              ;
                                0000G   CF          00  FB 0001D      CALLS   #0, GET_REQD_BFR_CREDITS              ; 1256
                                        0B          53  D1 00022      CMPL    FUNCTION, #11                         ; 1261
                                                    0F  1F 00025      BLSSU   3$                                    ;
                                        0C          53  D1 00027      CMPL    FUNCTION, #12                         ;
                                                    0A  1A 0002A      BGTRU   3$                                    ;
                                0000G   CF          00  FB 0002C      CALLS   #0, READ_WRITEVB                      ; 1262
                                        D3          50  E8 00031      BLBS    R0, 1$                                ;
                                                    3C  11 00034      BRB     8$                                    ;
                                        38          53  D1 00036 3$:  CMPL    FUNCTION, #56                         ; 1267
                                                    07  12 00039      BNEQ    4$                                    ;
                                0000G   CF          00  FB 0003B      CALLS   #0, ACPCONTROL                        ;
                                                    6B  11 00040      BRB     14$                                   ;
                                        39          53  D1 00042 4$:  CMPL    FUNCTION, #57                         ; 1268
                                                    07  12 00045      BNEQ    5$                                    ;
                                0000G   CF          00  FB 00047      CALLS   #0, MOUNT                             ;
                                                    5F  11 0004C      BRB     14$                                   ;
                                            FF7C    CA  D5 0004E 5$:  TSTL    -132(BASE)                            ; 1274
                                                    07  12 00052      BNEQ    6$                                    ;
                                              0000G 30  00054      BSBW    START_REQUEST                         ; 1277
                                  A7    AA          01  90 00057      MOVB    #1, -89(BASE)                         ; 1278
                          04                  32    53  CF 0005B 6$:  CASEL   FUNCTION, #50, #4                    ; 1281
                0042            003B          0034  0017     0005F 7$:  .WORD   9$-7$,-                            ;
                                                    0049     00067      10$-7$,-                              ;
                                                                        11$-7$,-                              ;
                                                                        12$-7$,-                              ;
                                                                        13$-7$                                ;
                                        05      80  AA  E9 00069      BLBC    -128(BASE), 8$                        ; 1301
                                  80    AA      F4  8F  9B 0006D      MOVZBW  #244, -128(BASE)                      ;
                                                    50  D4 00072 8$:  CLRL    R0                                    ;
                                                    37  11 00074      BRB     14$                                   ;
                                    0000G   CF          00  FB 00076 9$:  CALLS   #0, ACCESS                      ; 1285
                            00000910    8F          50  D1 0007B      CMPL    STATUS, #2320                         ; 1286
```

```
                          29  12 00082       BNEQ    14$                        ;
        51          90    AA  D0 00084       MOVL    -112(BASE), R1             ; 1287
                    20    A1  95 00088       TSTB    32(R1)                     ;
                          20  18 0008B       BGEQ    14$                        ;
  80    AA        0619    8F  3C 0008D       MOVZWL  #1561, -128(BASE)          ; 1290
0000G   CF                00  FB 00093 10$:  CALLS   #0, CREATE                 ; 1296
                          13  11 00098       BRB     14$                        ;
0000G   CF                00  FB 0009A 11$:  CALLS   #0, DEACCESS               ; 1297
                          0C  11 0009F       BRB     14$                        ;
0000G   CF                00  FB 000A1 12$:  CALLS   #0, DELETE                 ; 1298
                          05  11 000A6       BRB     14$                        ;
0000G   CF                00  FB 000A8 13$:  CALLS   #0, MODIFY                 ; 1299
        54                50  D0 000AD 14$:  MOVL    R0, STATUS                 ; 1258
                  02E4    CA  D5 000B0       TSTL    740(BASE)                  ; 1307
                          15  13 000B4       BEQL    16$                        ;
        0D                54  E8 000B6       BLBS    STATUS, 15$                ; 1310
                    04    AA  D5 000B9       TSTL    4(BASE)                    ; 1311
                          08  13 000BC       BEQL    15$                        ;
                    04    AA  DD 000BE       PUSHL   4(BASE)                    ; 1312
0000G   CF                01  FB 000C1       CALLS   #1, CHECKSUM               ;
0000V   CF                00  FB 000C6 15$:  CALLS   #0, PERFORM_AUDIT          ; 1313
        52        07D0    8F  3C 000CB 16$:  MOVZWL  #2000, J                   ; 1316
        08                54  E9 000D0 17$:  BLBC    STATUS, 18$                ; 1319
0000G   CF                00  FB 000D3       CALLS   #0, CLEANUP                ;
        0B                50  E8 000D8       BLBS    R0, 19$                    ;
0000G   CF                00  FB 000DB 18$:  CALLS   #0, ERR_CLEANUP            ; 1320
        54                50  D0 000E0       MOVL    R0, STATUS                 ;
        EA                52  F5 000E3       SOBGTR  J, 17$                     ; 1316
0000V   CF                00  FB 000E6 19$:  CALLS   #0, UNLOCK_XQP             ; 1323
0000G   CF                00  FB 000EB       CALLS   #0, PMS_END                ; 1324
                    90    AA  DD 000F0       PUSHL   -112(BASE)                 ; 1325
0000G   CF                01  FB 000F3       CALLS   #1, IO_DONE                ;
        03          A7    AA  E9 000F8       BLBC    -89(BASE), 20$             ; 1327
                       0000G  30 000FC       BSBW    FINISH_REQUEST             ; 1329
                        FF05  31 000FF 20$:  BRW     1$                         ; 1237
                              04 00102       RET                                ; 1333

; Routine Size:  259 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  346     1334  1 GLOBAL ROUTINE MAIN_HANDLER (SIGNAL, MECHANISM) : L_NORM NOVALUE =
;  347     1335  1
;  348     1336  1 !++
;  349     1337  1 !
;  350     1338  1 ! FUNCTIONAL DESCRIPTION:
;  351     1339  1 !
;  352     1340  1 !       This routine is the main level condition handler. It stores the
;  353     1341  1 !       condition value (FCP error code) in the user status block, unwinds
;  354     1342  1 !       and returns from the function that was executing.
;  355     1343  1 !
;  356     1344  1 ! CALLING SEQUENCE:
;  357     1345  1 !       MAIN_HANDLER (ARG1, ARG2)
;  358     1346  1 !
;  359     1347  1 ! INPUT PARAMETERS:
;  360     1348  1 !       ARG1: address of signal array
;  361     1349  1 !       ARG2: address of mechanism array
;  362     1350  1 !
;  363     1351  1 ! IMPLICIT INPUTS:
;  364     1352  1 !       NONE
;  365     1353  1 !
;  366     1354  1 ! OUTPUT PARAMETERS:
;  367     1355  1 !       NONE
;  368     1356  1 !
;  369     1357  1 ! IMPLICIT OUTPUTS:
;  370     1358  1 !       USER_STATUS: receives signal code
;  371     1359  1 !
;  372     1360  1 ! ROUTINE VALUE:
;  373     1361  1 !       NONE
;  374     1362  1 !
;  375     1363  1 ! SIDE EFFECTS:
;  376     1364  1 !       stack unwound to main level to return to dispatcher
;  377     1365  1 !
;  378     1366  1 !--
;  379     1367  1
;  380     1368  2 BEGIN
;  381     1369  2
;  382     1370  2 MAP
;  383     1371  2         SIGNAL          : REF BBLOCK,   ! signal array arg
;  384     1372  2         MECHANISM       : REF BBLOCK;   ! mechanism array arg
;  385     1373  2
;  386     1374  2 BIND_COMMON;
;  387     1375  2
;  388     1376  2 EXTERNAL ROUTINE
;  389     1377  2         SYS$UNWIND      : ADDRESSING_MODE (ABSOLUTE);
;  390     1378  2
;  391     1379  2
;  392     1380  2 ! Check the signal code. The only permissible ones are SS$_UNWIND, which
;  393     1381  2 ! is ignored, and SS$_CMODUSER. The error status is the 16 bit CHMU code.
;  394     1382  2 ! If the error value is non-zero, store it in the user status (zero
;  395     1383  2 ! means just exit). Set up a return value of 0, unwind to the current
;  396     1384  2 ! depth, and return, causing the invoked function to return with failure
;  397     1385  2 ! to the dispatcher.
;  398     1386  2 !
;  399     1387  2
;  400     1388  2 IF .SIGNAL[CHF$L_SIG_NAME] EQL SS$_UNWIND THEN RETURN;
;  401     1389  2 IF .SIGNAL[CHF$L_SIG_NAME] NEQ SS$_CMODUSER
;  402     1390  2 THEN BUG_CHECK (DNXSIGNAL, FATAL, 'Unexpected signal name in ACP');
```

```
;  403          1391  2
;  404          1392  2 IF .SIGNAL[CHF$L_SIG_ARG1] NEQ 0
;  405          1393  2 AND .USER_STATUS
;  406          1394  2 THEN USER_STATUS = .SIGNAL[CHF$L_SIG_ARG1];
;  407          1395  2
;  408          1396  2 MECHANISM[CHF$L_MCH_SAVR0] = .USER_STATUS;
;  409          1397  2
;  410          1398  2 SYS$UNWIND (MECHANISM[CHF$L_MCH_DEPTH], 0);
;  411          1399  2
;  412          1400  2 RETURN;
;  413          1401  2
;  414          1402  1 END;                                    ! end of routine MAIN_HANDLER


                                                        .EXTRN  SYS$UNWIND, BUG$_UNXSIGNAL

                                                        .ENTRY  MAIN_HANDLER, Save nothing      ; 1334
                                    50      04  AC  D0 00002     MOVL    SIGNAL, R0              ; 1388
                              00000920      8F  04  A0  D1 00006     CMPL    4(R0), #2336
                                                    37  13 0000E     BEQL    3$
                              00000424      8F  04  A0  D1 00010     CMPL    4(R0), #1060            ; 1389
                                                    04  13 00018     BEQL    1$
                                                      FEFF 0001A     BUGW                            ; 1390
                                                     0000* 0001C     .WORD   <BUG$_UNXSIGNAL!4>
                                    50      04  AC  D0 0001E 1$:    MOVL    SIGNAL, R0              ; 1392
                                            08  A0  D5 00022     TSTL    8(R0)
                                                09  13 00025     BEQL    2$
                                    05      80  AA  E9 00027     BLBC    -128(BASE), 2$          ; 1393
                                        80  AA  08  A0  D0 0002B     MOVL    8(R0), -128(BASE)       ; 1394
                                    50      08  AC  D0 00030 2$:    MOVL    MECHANISM, R0           ; 1396
                                        0C  A0  80  AA  D0 00034     MOVL    -128(BASE), 12(R0)
                                                7E  D4 00039     CLRL    -(SP)                   ; 1398
                                7E      08  AC          08  C1 0003B     ADDL3   #8, MECHANISM, -(SP)
                                  00000000G 9F          02  FB 00040     CALLS   #2, @#SYS$UNWIND
                                                        04 00047 3$:    RET                     ; 1402

; Routine Size:  72 bytes,    Routine Base:  $CODE$ + 0103

;  415          1403  1
```

```
;  417     1404  1 GLOBAL ROUTINE ZERO_ON_ERROR (SIGNAL, MECHANISM) =
;  418     1405  1
;  419     1406  1 !++
;  420     1407  1 !
;  421     1408  1 ! FUNCTIONAL DESCRIPTION:
;  422     1409  1 !
;  423     1410  1 !       This condition handler is used in various places to cause a
;  424     1411  1 !       function to return zero if any error is signalled during its
;  425     1412  1 !       operation. The actual error is ignored.
;  426     1413  1 !
;  427     1414  1 ! CALLING SEQUENCE:
;  428     1415  1 !       ZERO_ON_ERROR (SIGNAL, MECHANISM)
;  429     1416  1 !
;  430     1417  1 ! INPUT PARAMETERS:
;  431     1418  1 !       SIGNAL: address of condition signal vector
;  432     1419  1 !       MECHANISM: address of condition mechanism vector
;  433     1420  1 !
;  434     1421  1 ! IMPLICIT INPUTS:
;  435     1422  1 !       NONE
;  436     1423  1 !
;  437     1424  1 ! OUTPUT PARAMETERS:
;  438     1425  1 !       NONE
;  439     1426  1 !
;  440     1427  1 ! IMPLICIT OUTPUTS:
;  441     1428  1 !       NONE
;  442     1429  1 !
;  443     1430  1 ! ROUTINE VALUE:
;  444     1431  1 !       SS$_RESIGNAL
;  445     1432  1 !
;  446     1433  1 ! SIDE EFFECTS:
;  447     1434  1 !       Stack unwound to establisher (FLUSH_QUO_CACHE)
;  448     1435  1 !
;  449     1436  1 !--
;  450     1437  1
;  451     1438  2 BEGIN
;  452     1439  2
;  453     1440  2 MAP
;  454     1441  2         SIGNAL          : REF BBLOCK,   ! signal arg list
;  455     1442  2         MECHANISM       : REF BBLOCK;   ! mechanism arg list
;  456     1443  2
;  457     1444  2
;  458     1445  2 ! Check for an error signal. All others are resignaled  On an error
;  459     1446  2 ! set the return R0 to 0 and unwind to establisher.
;  460     1447  2 !
;  461     1448  2
;  462     1449  2 IF .SIGNAL[CHF$L_SIG_NAME] EQL SS$_CMODUSER
;  463     1450  2 THEN
;  464     1451  3     BEGIN
;  465     1452  3     MECHANISM[CHF$L_MCH_SAVR0] = 0;
;  466     1453  3     $UNWIND (DEPADR = MECHANISM[CHF$L_MCH_DEPTH]);
;  467     1454  2     END;
;  468     1455  2
;  469     1456  2 SS$_RESIGNAL
;  470     1457  1 END;                                    ! End of routine ZERO_ON_ERROR
```

```
                                        0000 00000          .ENTRY  ZERO_ON_ERROR, Save nothing                     ; 1404
                                50      04   AC   D0 00002  MOVL    SIGNAL, R0                                      ; 1449
                00000424        8F      04   A0   D1 00006  CMPL    4(R0), #1060                                    ;
                                                  15   12 0000E  BNEQ    1$                                          ;
                                50      08   AC   D0 00010  MOVL    MECHANISM, R0                                   ; 1452
                                        0C   A0   D4 00014  CLRL    12(R0)                                          ;
                                                  7E   D4 00017  CLRL    -(SP)                                       ; 1453
                        7E          08  AC        08   C1 00019  ADDL3   #8, MECHANISM, -(SP)                        ;
                00000000G       00      02   FB 0001E  CALLS   #2, SYS$UNWIND                                     ;
                                50      0918 8F   3C 00025 1$:  MOVZWL  #2328, R0                                   ; 1457
                                                  04   0002A  RET                                                    ;

; Routine Size:  43 bytes,    Routine Base:  $CODE$ + 014B
```

```
;  472          1458  1 GLOBAL ROUTINE UNLOCK_XQP : L_NORM NOVALUE =
;  473          1459  1 !++
;  474          1460  1 !
;  475          1461  1 ! FUNCTIONAL DESCRIPTION:
;  476          1462  1 !       This routine releases the xqp synchonization locks.
;  477          1463  1 !--
;  478          1464  2 BEGIN
;  479          1465  2
;  480          1466  2 EXTERNAL ROUTINE
;  481          1467  2         RETURN_CREDITS          : L_NORM,
;  482          1468  2         ALLOCATION_UNLOCK       : L_NORM,
;  483          1469  2         RELEASE_SERIAL_LOCK     : L_NORM;
;  484          1470  2
;  485          1471  2 BIND_COMMON;
;  486          1472  2
;  487          1473  2 LOCAL
;  488          1474  2         LOCKID;
;  489          1475  2
;  490          1476  3 INCR I FROM 1 TO (LB_NUM - 1)
;  491          1477  2 DO
;  492          1478  2     IF .LB_LOCKID [.I] NEQ 0
;  493          1479  2     THEN
;  494          1480  2         RELEASE_SERIAL_LOCK (.I);
;  495          1481  2
;  496          1482  2 ALLOCATION_UNLOCK ();
;  497          1483  2
;  498          1484  2 RETURN_CREDITS ();
;  499          1485  2
;  500          1486  1 END;


                                                .EXTRN  RETURN_CREDITS, ALLOCATION_UNLOCK
                                                .EXTRN  RELEASE_SERIAL_LOCK

                                0004 00000      .ENTRY  UNLOCK_XQP, Save R2                     ; 1458
                        52        01  D0 00002 MOVL    #1, I                                   ; 1474
                                6C AA42  D5 00005 1$:   TSTL    108(BASE)[I]                    ; 1478
                                   07  13 00009 BEQL    2$                                      ;
                                   52  DD 0000B PUSHL   I                                       ; 1480
                0000G   CF         01  FB 0000D CALLS   #1, RELEASE_SERIAL_LOCK                  ;
             EF         52         04  F3 00012 2$:    AOBLEQ  #4, I, 1$                       ; 1478
                0000G   CF         00  FB 00016 CALLS   #0, ALLOCATION_UNLOCK                    ; 1482
                0000G   CF         00  FB 0001B CALLS   #0, RETURN_CREDITS                       ; 1484
                                   04  00020 RET                                               ; 1486

; Routine Size:  33 bytes,    Routine Base:  $CODE$ + 0176
```

```
;  502     1487  1 GLOBAL ROUTINE PERFORM_AUDIT : L_NORM NOVALUE =
;  503     1488  1
;  504     1489  1 !++
;  505     1490  1 !
;  506     1491  1 ! FUNCTIONAL DESCRIPTION:
;  507     1492  1 !
;  508     1493  1 !       This routine outputs any pending audit records that may have resulted
;  509     1494  1 !       from protection checks performed in this file operation. They are
;  510     1495  1 !       deferred to this point because of the disruption caused by the
;  511     1496  1 !       call to FID_TO_SPEC.
;  512     1497  1 !
;  513     1498  1 ! CALLING SEQUENCE:
;  514     1499  1 !       PERFORM_AUDIT ()
;  515     1500  1 !
;  516     1501  1 ! INPUT PARAMETERS:
;  517     1502  1 !       NONE
;  518     1503  1 !
;  519     1504  1 ! IMPLICIT INPUTS:
;  520     1505  1 !       NONE
;  521     1506  1 !
;  522     1507  1 ! OUTPUT PARAMETERS:
;  523     1508  1 !       NONE
;  524     1509  1 !
;  525     1510  1 ! IMPLICIT OUTPUTS:
;  526     1511  1 !       NONE
;  527     1512  1 !
;  528     1513  1 ! ROUTINE VALUE:
;  529     1514  1 !       NONE
;  530     1515  1 !
;  531     1516  1 ! SIDE EFFECTS:
;  532     1517  1 !       NONE
;  533     1518  1 !
;  534     1519  1 !--
;  535     1520  1
;  536     1521  2 BEGIN                                   ! Start of routine PERFORM_AUDIT
;  537     1522  2
;  538     1523  2 LOCAL
;  539     1524  2         AUDIT_BLOCK     : REF BBLOCK;   ! pointer to saved audit block
;  540     1525  2
;  541     1526  2 BIND_COMMON;
;  542     1527  2
;  543     1528  2 EXTERNAL ROUTINE
;  544     1529  2         SERIAL_FILE     : L_NORM;       ! acquire file synchronization lock
;  545     1530  2
;  546     1531  2
;  547     1532  2 ! Step through the list of saved audits and write the audit for each block
;  548     1533  2 ! that contains a valid entry.
;  549     1534  2 !
;  550     1535  2
;  551     1536  2 AUDIT_BLOCK = AUDIT_ARGLIST;
;  552     1537  2 DECR J FROM MAX_AUDIT_COUNT TO 1
;  553     1538  2 DO
;  554     1539  3     BEGIN
;  555     1540  3     IF .AUDIT_BLOCK[AUDIT_TYPE] NEQ 0
;  556     1541  3     THEN
;  557     1542  4         BEGIN
;  558     1543  4         SERIAL_FILE (AUDIT_BLOCK[AUDIT_FID]);
```

```
;  559          1544  4          WRITE_AUDIT (.AUDIT_BLOCK);
;  560          1545  3          END;
;  561          1546  3      AUDIT_BLOCK = .AUDIT_BLOCK + AUDIT_LENGTH;
;  562          1547  2      END;
;  563          1548  2
;  564          1549  1 END;                                    ! End of routine PERFORM_AUDIT


                                                        .EXTRN  SERIAL_FILE

                                          000C 00000    .ENTRY  PERFORM_AUDIT, Save R2,R3                       ; 1487
                              52    0924  CA   9E 00002  MOVAB   2340(BASE), AUDIT_BLOCK                        ; 1536
                              53          04   D0 00007  MOVL    #4, J                                         ; 1537
                                          62   95 0000A 1$:    TSTB    (AUDIT_BLOCK)                           ; 1540
                                          0F   13 0000C  BEQL    2$                                            ;
                                    02    A2   9F 0000E  PUSHAB  2(AUDIT_BLOCK)                                ; 1543
                          0000G   CF          01   FB 00011  CALLS   #1, SERIAL_FILE                          ;
                                          52   DD 00016  PUSHL   AUDIT_BLOCK                                  ; 1544
                          0000V   CF          01   FB 00018  CALLS   #1, WRITE_AUDIT                          ;
                              52          10   C0 0001D 2$:    ADDL2   #16, AUDIT_BLOCK                        ; 1546
                              E7          53   F5 00020  SOBGTR  J, 1$                                        ; 1537
                                          04   00023  RET                                                     ; 1549

; Routine Size:  36 bytes,    Routine Base:  $CODE$ + 0197
```

```
;  566          1550  1 GLOBAL ROUTINE WRITE_AUDIT (AUDIT_BLOCK) : L_NORM NOVALUE =
;  567          1551  1
;  568          1552  1 !++
;  569          1553  1 !
;  570          1554  1 ! FUNCTIONAL DESCRIPTION:
;  571          1555  1 !
;  572          1556  1 !       This routine writes a security audit record based on the specified
;  573          1557  1 !       saved audit block. Most of the information has been collected in
;  574          1558  1 !       CHECK_PROTECT; All that remains is to construct the actual audit
;  575          1559  1 !       arg list and call NSA$EVENT_AUDIT.
;  576          1560  1 !
;  577          1561  1 ! CALLING SEQUENCE:
;  578          1562  1 !       WRITE_AUDIT (AUDIT_BLOCK)
;  579          1563  1 !
;  580          1564  1 ! INPUT PARAMETERS:
;  581          1565  1 !       AUDIT_BLOCK: address fo saved audit info
;  582          1566  1 !
;  583          1567  1 ! IMPLICIT INPUTS:
;  584          1568  1 !       NONE
;  585          1569  1 !
;  586          1570  1 ! OUTPUT PARAMETERS:
;  587          1571  1 !       NONE
;  588          1572  1 !
;  589          1573  1 ! IMPLICIT OUTPUTS:
;  590          1574  1 !       NONE
;  591          1575  1 !
;  592          1576  1 ! ROUTINE VALUE:
;  593          1577  1 !       NONE
;  594          1578  1 !
;  595          1579  1 ! SIDE EFFECTS:
;  596          1580  1 !       NONE
;  597          1581  1 !
;  598          1582  1 !--
;  599          1583  1
;  600          1584  2 BEGIN                                   ! Start of routine WRITE_AUDIT
;  601          1585  2
;  602          1586  2 BUILTIN
;  603          1587  2         CALLG;
;  604          1588  2
;  605          1589  2 MAP
;  606          1590  2         AUDIT_BLOCK     : REF BBLOCK;                   ! audit info block
;  607          1591  2
;  608          1592  2 LOCAL
;  609          1593  2         ARGLIST         : BBLOCK [NSA$K_ARG1_LENGTH],
;  610          1594  2                                                   ! audit argument list
;  611          1595  2         LOC_HEADER;                               ! Local copy of file header address
;  612          1596  2
;  613          1597  2 LINKAGE
;  614          1598  2         ARGLST_IMGNAM   = JSB (REGISTER = 2;) :
;  615          1599  2                           NOPRESERVE (0,1)
;  616          1600  2                           NOTUSED (3,4,5,6,7,8,9,10,11);
;  617          1601  2
;  618          1602  2 EXTERNAL ROUTINE
;  619          1603  2         READ_HEADER     : L_NORM,       ! Read file header
;  620          1604  2         WRITE_DIRTY     : L_NORM,       ! Write dirty buffers on lock
;  621          1605  2         FID_TO_SPEC     : NOVALUE L_NORM, ! Convert FID to file name
;  622          1606  2         NSA$ARGLST_IMGNAM : ARGLST_IMGNAM ADDRESSING_MODE (GENERAL),
```

```
;  623          1607  2                                    ! get image name for audit record
;  624          1608  2          NSA$EVENT_AUDIT : ADDRESSING_MODE (GENERAL);
;  625          1609  2                                    ! Security auditing routine
;  626          1610  2
;  627          1611  2 BIND_COMMON;
;  628          1612  2
;  629          1613  2
;  630          1614  2 ! Build the audit argument list from the saved info.
;  631          1615  2 !
;  632          1616  2
;  633          1617  2 ARGLIST[NSA$B_ARG_FLAG] = .AUDIT_BLOCK[AUDIT_TYPE];
;  634          1618  2 ARGLIST[NSA$L_ARGT_FACMOD_TM] = NSA$K_ARG_MECH_LONG^16 + NSA$K_PKTTYP_FACMOD;
;  635          1619  2 ARGLIST[NSA$L_ARG1_FACMOD] = .AUDIT_BLOCK[AUDIT_ACCESS];
;  636          1620  2 ARGLIST[NSA$L_ARG1_FILNAM_TM] = NSA$K_ARG_MECH_DESCR^16 + NSA$K_PKTTYP_FILNAM;
;  637          1621  2
;  638          1622  2 IF .AUDIT_BLOCK[AUDIT_SUCCESS]
;  639          1623  2 THEN
;  640          1624  3     BEGIN
;  641          1625  3     ARGLIST[NSA$L_ARG_COUNT] = 12;
;  642          1626  3     ARGLIST[NSA$L_ARG_ID] = NSA$K_RECID_FIL_SUCC;
;  643          1627  3     ARGLIST[NSA$B_ARG_PKTNUM] = 4;
;  644          1628  3     ARGLIST[NSA$L_ARGT_PRIVUSED_TM] = NSA$K_ARG_MECH_LONG^16 + NSA$K_PKTTYP_PRIVUSED;
;  645          1629  3     ARGLIST[NSA$L_ARG1_PRIVUSED] = .AUDIT_BLOCK[AUDIT_PRIVS];
;  646          1630  3     END
;  647          1631  2 ELSE
;  648          1632  3     BEGIN
;  649          1633  3     ARGLIST[NSA$L_ARG_COUNT] = 10;
;  650          1634  3     ARGLIST[NSA$L_ARG_ID] = NSA$K_RECID_FIL_FAIL;
;  651          1635  3     ARGLIST[NSA$B_ARG_PKTNUM] = 3;
;  652          1636  3     END;
;  653          1637  2
;  654          1638  2 LOC_HEADER = READ_HEADER (AUDIT_BLOCK[AUDIT_FID], 0);
;  655          1639  2 WRITE_DIRTY (.LB_BASIS[.PRIM_LCKINDX]);
;  656          1640  2 FID_TO_SPEC (.LOC_HEADER);
;  657          1641  2 ARGLIST[NSA$L_ARGT_FILNAM_SIZ] = .FILE_SPEC_LEN<0,16,0>;
;  658          1642  2 ARGLIST[NSA$L_ARG1_FILNAM_PTR] = FULL_FILE_SPEC;
;  659          1643  2 NSA$ARGLST_IMGNAM (ARGLIST[NSA$L_ARG1_IMGNAM_TM]);
;  660          1644  2 CALLG (ARGLIST, NSA$EVENT_AUDIT);
;  661          1645  2
;  662          1646  2 ! Note this entry processed.
;  663          1647  2 !
;  664          1648  2
;  665          1649  2 AUDIT_BLOCK[AUDIT_TYPE] = 0;
;  666          1650  2 AUDIT_COUNT = .AUDIT_COUNT - 1;
;  667          1651  2
;  668          1652  1 END;                                ! End of routine WRITE_AUDIT


                                                          .EXTRN  READ_HEADER, WRITE_DIRTY
                                                          .EXTRN  FID_TO_SPEC, NSA$ARGLST_IMGNAM
                                                          .EXTRN  NSA$EVENT_AUDIT

                                        0004 00000        .ENTRY  WRITE_AUDIT, Save R2                    ; 1550
                              5E        34   C2 00002      SUBL2   #52, SP                                 ;
                              50   04   AC   D0 00005      MOVL    AUDIT_BLOCK, R0                         ; 1617
                        08    AE        60   90 00009      MOVB    (R0), ARGLIST+8                         ;
```

```
                0C   AE 00020002   8F   D0 0000D          MOVL    #131074, ARGLIST+12                     ; 1618
                10   AE       08   A0   D0 00015          MOVL    8(R0), ARGLIST+16                       ; 1619
                14   AE 00040004   8F   D0 0001A          MOVL    #262148, ARGLIST+20                     ; 1620
                     1E       01   A0   E9 00022          BLBC    1(R0), 1$                               ; 1622
                     6E            0C   D0 00026          MOVL    #12, ARGLIST                            ; 1625
                04   AE 00010001   8F   D0 00029          MOVL    #65537, ARGLIST+4                       ; 1626
                09   AE            04   90 00031          MOVB    #4, ARGLIST+9                           ; 1627
                2C   AE 00020003   8F   D0 00035          MOVL    #131075, ARGLIST+44                     ; 1628
                30   AE       0C   A0   D0 0003D          MOVL    12(R0), ARGLIST+48                      ; 1629
                                   0F   11 00042          BRB     2$                                      ; 1622
                     6E            0A   D0 00044 1$:      MOVL    #10, ARGLIST                            ; 1633
                04   AE 00020001   8F   D0 00047          MOVL    #131073, ARGLIST+4                      ; 1634
                09   AE            03   90 0004F          MOVB    #3, ARGLIST+9                           ; 1635
                                   7E   D4 00053 2$:      CLRL    -(SP)                                   ; 1638
                          02       A0   9F 00055          PUSHAB  2(R0)                                   ;
          0000G CF                 02   FB 00058          CALLS   #2, READ_HEADER                         ;
                52                 50   D0 0005D          MOVL    R0, LOC_HEADER                          ;
                50            18   AA   D0 00060          MOVL    24(BASE), R0                            ; 1639
                         0080 CA40   DD 00064          PUSHL   128(BASE)[R0]                           ;
          0000G CF                 01   FB 00069          CALLS   #1, WRITE_DIRTY                         ;
                                   52   DD 0006E          PUSHL   LOC_HEADER                              ; 1640
          0000G CF                 01   FB 00070          CALLS   #1, FID_TO_SPEC                         ;
                18   AE      04E8   CA   3C 00075          MOVZWL  1256(BASE), ARGLIST+24                  ; 1641
                1C   AE      04EA   CA   9E 0007B          MOVAB   1258(BASE), ARGLIST+28                  ; 1642
                52            20   AE   9E 00081          MOVAB   ARGLIST+32, R2                          ; 1643
                     000000000G   00   16 00085          JSB     NSA$ARGLST_IMGNAM                       ;
     00000000G   00                6E   FA 0008B          CALLG   ARGLIST, NSA$EVENT_AUDIT                ; 1644
                               04   BC   94 00092          CLRB    @AUDIT_BLOCK                            ; 1649
                          02E4   CA   D7 00095          DECL    740(BASE)                               ; 1650
                                      04 00099          RET                                             ; 1652

; Routine Size:  154 bytes,    Routine Base:  $CODE$ + 01BB

;  669          1653  1
;  670          1654  1 END
;  671          1655  0 ELUDOM



                                    PSECT SUMMARY

;       Name                    Bytes                           Attributes

;  $CODE$                        597  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        61         0       1000          00:01.9


;                                       COMMAND QUALIFIERS

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

; Size:         597 code + 0 data bytes
; Run Time:       00:52.1
; Elapsed Time:   02:04.0
; Lines/CPU Min:   1906
; Lexemes/CPU-Min: 64875
; Memory Used:  232 pages
; Compilation Complete
```

DEACCS
LIS

DELETE
LIS

DIRSCN
LIS

DISPAT
LIS

CREHDR
LIS

DIRACC
LIS

DELBAD
LIS

CREFCB
LIS

CREWIN
LIS

DISPATCH
LIS

ENTER
LIS

DELFIL
LIS