Table of contents

```
                                        .TITLE  TSLOCK  Shared file access control
                                        .ENABL  LC
                                        .DSABL  GBL
                                ;
                                ;  Copyright (c) 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985.
                                ;
                                ;  S&H Computer Systems, Inc.
                                ;  Nashville, Tennessee
                                ;
                                ;  Written by Phil Sherrod.
                                ;
                                ;  The routines in this module perform management
                                ;  functions such as record (block) locking for shared files.
                                ;
000000                                  .CSECT  TSLOCK
                                ;
                                ;  Macro calls
                                ;
                                        .MCALL  .ADDR
                                ;
                                ;  Global definitions
                                ;
                                        .GLOBL  DCCSIZ,TSLOCK
                                ;
                                ;  Global references
                                ;
                                        .GLOBL  LSTATE,LPRI,ENQTL
                                        .GLOBL  FC$CHN,FC$FDB,FC$FLK
                                        .GLOBL  FC$CLK,FC$LBN,FL$ACT,FC$UN,FC$ACC,FL$SPN
                                        .GLOBL  DCTRD,DCCRD,FL$NDC,IOWAIT,DC$BLK,DCTWR
                                        .GLOBL  DCCWR,DCTOTU,VALADW,DCAGE,NUMCDB
                                        .GLOBL  FF$FID,FF$CDB,FF$FWD,FF$FLK
                                        .GLOBL  FW$DBN,FW$UN,FW$WLK,VMXSF,VMXSFC,NLINES
                                        .GLOBL  VMLBLK,EMTBLK,CORUSR,CHNADR,VNUMDC,DC$$SZ
                                        .GLOBL  JCDB,CHNNUM,NUMDCD,LOKCSH
                                        .GLOBL  FC$LBN,FF$$SZ,FW$$SZ,DC$NXT,DC$PAR
                                        .GLOBL  LDBASE,LOKMEM,VPAR5,DC$FDB,DC$USE,DC$LNK
                                        .GLOBL  CHKABT,QHIPRI,QNSPND,S$SFWT
                                        .GLOBL  FL$EFL,FT$EFL,FF$NLB,FF$FLG,FC$FLG
                                        .GLOBL  FF$DCD
                                        .GLOBL  LDDEVX,LDPDEV
                                        .GLOBL  C.CSW,C.SBLK,C.DEVQ
                                        .GLOBL  CS$OPN,SETERR,EMTXIT,KPAR5,FC$$SS,FC$NLB,CS$NMX
                                ;
                                ;  Macro to map kernel par 5 to a specified region
                                ;
                                        .MACRO  MAPTO   BASE
                                        MOV     BASE,@#KPAR5
                                        .ENDM   MAPTO
                                ;
                                ;-----------------------------------------------------------------------
                                ;  Module header.  Must be first data in segment.
                                ;
000000  046543                 TSLOCK: .RAD50  /LOK/           ;System overlay region ID name
                                ;
                                ;  Table of offsets to entry points in this module
                                ;
```

```
      58 000002  000112                          .WORD   LOKINI-TSLOCK
      59 000004  000552                          .WORD   DOOPAP-TSLOCK
      60 000006  000560                          .WORD   DOCOPN-TSLOCK
      61 000010  001136                          .WORD   SFSVST-TSLOCK
      62 000012  001200                          .WORD   SFRSST-TSLOCK
      63 000014  001414                          .WORD   DORLK-TSLOCK
      64 000016  001470                          .WORD   DOTLK-TSLOCK
      65 000020  001574                          .WORD   DOCULK-TSLOCK
      66 000022  001622                          .WORD   DOULK1-TSLOCK
      67 000024  001726                          .WORD   DOSFCK-TSLOCK
      68 000026  002246                          .WORD   SFCLS-TSLOCK
      69 000030  004630                          .WORD   SFWRIT-TSLOCK
      70 000032  003706                          .WORD   CLSCDB-TSLOCK
      71 000034  005000                          .WORD   DCRD1-TSLOCK
      72 000036  005162                          .WORD   DCRD2-TSLOCK
```

```
                                ;*******************************************************************************
                                ;  Define symbols for returned error codes
                                ;
000000                          EC0     =       0.
000001                          EC1     =       1.
000002                          EC2     =       2.
000003                          EC3     =       3.
000004                          EC4     =       4.
```

```
                                ;
                                ;  Shared file open access protection table.
                                ;
                                ;  Access-protection enable flags:
                                ;
000001                          AC$SU   =       1               ;Shared update
000002                          AC$SI   =       2               ;Shared input
000004                          AC$PU   =       4               ;Protected update
000010                          AC$PI   =       10              ;Protected input
000020                          AC$EU   =       20              ;Exclusive update
000040                          AC$EI   =       40              ;Exclusive input
                                ;
                                ;  Format of access code passed from user to us:
                                ;
                                ;  Bit  0  : 0==> input, 1==> update.
                                ;  Bits 1-2: 0==> Exclusive,
                                ;            1==> Protected,
                                ;            2==> Shared.
                                ;  Bit  8  : 1==> Suppress data caching.
                                ;
                                ;  Table to convert user access code into AC$ access bits.
                                ;
000040      040                 ACCBIT: .BYTE   AC$EI           ;Exclusive input
000041      020                         .BYTE   AC$EU           ;Exclusive update
000042      010                         .BYTE   AC$PI           ;Protected input
000043      004                         .BYTE   AC$PU           ;Protected update
000044      002                         .BYTE   AC$SI           ;Shared input
000045      001                         .BYTE   AC$SU           ;Shared update
                                ;
                                ;  Table of access exclusion masks.
                                ;  (Bit set implies that access type is denied)
                                ;
000046      377                 ACCMSK: .BYTE   ^C<0>                   ;Exclusive input
000047      377                         .BYTE   ^C<0>                   ;Exclusive update
000050      365                         .BYTE   ^C<AC$PI+AC$SI>         ;Protected input
000051      375                         .BYTE   ^C<AC$SI>               ;Protected update
000052      360                         .BYTE   ^C<AC$PI+AC$PU+AC$SI+AC$SU> ;Shared update
000053      374                         .BYTE   ^C<AC$SI+AC$SU>         ;Shared update
                                        .EVEN
                                ;
                                ;  Data structure list heads
                                ;
000054  000000  000000          FFFREE: .WORD   0,0             ;Head of free FDB list
000060  000000  000000          FFHEAD: .WORD   0,0             ;Head of active FDB list
000064  000000  000000          FCFREE: .WORD   0,0             ;Head of free CDB list
000070  000000  000000          FWFREE: .WORD   0,0             ;Head of free wait element list
000074  000000  000000          DCDBAS: .WORD   0,0             ;Head of list of data cache descriptors
000100  000000                  CBLOCK: .WORD   0               ;Block being read or written
000102  000000                  CBUF:   .WORD   0               ;Address of user's buffer
000104  000000                  CWORDS: .WORD   0               ;Number of words being read or written
                                ;
                                ;  2-word file ID for file to which record locking operation is being directed
                                ;
000106  000000                  CURFID: .WORD   0               ;Device index in low byte, unit # high byte
000110  000000                          .WORD   0               ;Starting block number of file on phys device
```

```
                                        .SBTTL  LOKINI -- Initialization for record locking system
                                ;-------------------------------------------------------------------------
                                ;  LOKINI is called during system initialization to initialize the
                                ;  record locking data base.
                                ;
000112  010146                 LOKINI: MOV     R1,-(SP)
000114  010246                         MOV     R2,-(SP)
000116  010346                         MOV     R3,-(SP)
000120  010446                         MOV     R4,-(SP)
000122  010546                         MOV     R5,-(SP)
000124  013746  000000G                MOV     @#KPAR5,-(SP)
                                ;
                                ;  Initialize pointer to data area
                                ;
000130  012704  000000G                MOV     #VPAR5,R4       ;Get virtual addr to use to access data
000134  013705  000000G                MOV     @#LOKMEM,R5     ;Get physical 64-byte blk # of data area
                                ;
                                ;  Allocate space for wait queue elements
                                ;
000140  012701  000000G                MOV     #FW$$SZ,R1      ;Get size of a wait queue element
000144  010467  177720                 MOV     R4,FWFREE       ;Init head of free list
000150  010567  177716                 MOV     R5,FWFREE+2
000154  012700  177777G                MOV     #NLINES-1,R0    ;Get # elements to allocate - 1
000160  001407                         BEQ     2$              ;Br if only one element to allocate
000162  004767  000272         1$:     CALL    ALLOCL          ;Allocate space for an element
000166  010462  000000G                MOV     R4,FW$WLK(R2)   ;Make previous one point to new one
000172  010562  000002G                MOV     R5,FW$WLK+2(R2)
000176  077007                         SOB     R0,1$           ;Loop if more to allocate
000200  004767  000272         2$:     CALL    ALLOC           ;Allocate last one
                                ;
                                ;  Allocate FDB tables
                                ;
000204  010467  177644                 MOV     R4,FFFREE       ;Set pointer to 1st free FDB
000210  010567  177642                 MOV     R5,FFFREE+2
000214  012701  000000G                MOV     #FF$$SZ,R1      ;Get size of each entry
000220  013700  000000G                MOV     @#VMXSF,R0      ;Get # to allocate
000224  005300                         DEC     R0              ;Minus one
000226  003407                         BLE     3$              ;Br if only one wanted
000230  004767  000224         4$:     CALL    ALLOCL          ;Allocate space
000234  010462  000000G                MOV     R4,FF$FLK(R2)   ;Make previous point to new one
000240  010562  000002G                MOV     R5,FF$FLK+2(R2)
000244  077007                         SOB     R0,4$           ;Loop if more to allocate
000246  004767  000224         3$:     CALL    ALLOC           ;Allocate space for last one
                                ;
                                ;  Allocate CDB tables
                                ;
000252  010467  177606                 MOV     R4,FCFREE       ;Set pointer to 1st free one
000256  010567  177604                 MOV     R5,FCFREE+2
000262  013701  000000G                MOV     @#VMLBLK,R1     ;Get max # blocks each chan can lock
000266  006301                         ASL     R1              ;Reserve two bytes per locked block
000270  062701  000000G                ADD     #FC$LBN,R1      ;Add space needed by rest of entry
000274  013700  000000G                MOV     @#VMXSFC,R0     ;Get # to allocate
000300  005300                         DEC     R0              ;Minus one
000302  003407                         BLE     5$              ;Br if only one wanted
000304  004767  000150         6$:     CALL    ALLOCL          ;Allocate space
000310  010462  000000G                MOV     R4,FC$CLK(R2)   ;Make previous entry point to new one
000314  010562  000002G                MOV     R5,FC$CLK+2(R2)
```

```
     58 000320  077007                        SOB     R0,6$           ;Loop if more to allocate
     59 000322  004767  000150          5$:   CALL    ALLOC           ;Allocate space for last one
     60                                ;
     61                                ;  Allocate data cache descriptors
     62                                ;
     63 000326  013700  000000G             MOV     @#VNUMDC,R0     ;Get # DCD's wanted
     64 000332  001442                      BEQ     20$             ;Br if shared file data caching not wanted
     65 000334  010467  177534              MOV     R4,DCDBAS       ;Set pointer to 1st DCD
     66 000340  010567  177532              MOV     R5,DCDBAS+2
     67 000344  012701  000000G             MOV     #DC$$SZ,R1      ;Get size of data cache descriptor block
     68 000350  005300                      DEC     R0              ;Get 1 less DCD
     69 000352  003407                      BLE     8$              ;Br if only 1 wanted
     70 000354  004767  000100          7$:   CALL    ALLOCL          ;Allocate space for a DCD
     71 000360  010462  000000G             MOV     R4,DC$NXT(R2)   ;Make previous entry point to new one
     72 000364  010562  000002G             MOV     R5,DC$NXT+2(R2)
     73 000370  077007                      SOB     R0,7$
     74 000372  004767  000100          8$:   CALL    ALLOC           ;Allocate last one
     75                                ;
     76                                ;  Set up pointers to data cache buffers
     77                                ;
     78 000376  016704  177472              MOV     DCDBAS,R4       ;Get pointer to 1st DCD
     79 000402  016705  177470              MOV     DCDBAS+2,R5
     80 000406  013701  000000G             MOV     @#LOKCSH,R1     ;Get par value for start of area
     81 000412                          9$:   MAPTO   R5              ;Map to next DCD
     82 000416  010164  000000G             MOV     R1,DC$PAR(R4)   ;Set pointer to cache buffer
     83 000422  062701  000010              ADD     #512./64.,R1    ;Point to next 512-byte buffer
     84 000426  016405  000002G             MOV     DC$NXT+2(R4),R5 ;Get pointer to next DCD
     85 000432  016404  000000G             MOV     DC$NXT(R4),R4
     86 000436  001365                      BNE     9$              ;Br if more to allocate
     87                                ;
     88                                ;  Finished
     89                                ;
     90 000440  012637  000000G         20$:  MOV     (SP)+,@#KPAR5
     91 000444  012605                      MOV     (SP)+,R5
     92 000446  012604                      MOV     (SP)+,R4
     93 000450  012603                      MOV     (SP)+,R3
     94 000452  012602                      MOV     (SP)+,R2
     95 000454  012601                      MOV     (SP)+,R1
     96 000456  000207                      RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Save the pointer to a current list element, allocate and new element,
                                ;  and set mapping to point to the previous element.
                                ;
                                ;  Inputs:
                                ;   R1 = Size of element to allocate
                                ;   R4,R5 = Pointer to current element
                                ;
                                ;  Outputs:
                                ;   R4,R5 = Pointer to new element
                                ;   R2,R3 = Pointer to previous element
                                ;   Mapping is set up to access previous element
                                ;   All registers are preserved.
                                ;
000460                          ALLOCL:
                                ;
                                ;  Save pointer to current element
                                ;
000460  010402                          MOV     R4,R2           ;Save pointer to current element
000462  010503                          MOV     R5,R3
                                ;
                                ;  Allocate a new element
                                ;
000464  004767  000006                  CALL    ALLOC           ;Allocate a new element
                                ;
                                ;  Set mapping to previous element
                                ;
000470                                  MAPTO   R3              ;Map to previous element
                                ;
                                ;  Finished
                                ;
000474  000207                          RETURN
```

```
                        ;-----------------------------------------------------------------------
                        ;  Allocate space for a list element.
                        ;  Adjust both the virtual and physical address pointers to keep the
                        ;  virtual address within the par 5 region.
                        ;
                        ;  Inputs:
                        ;   R1 = Size of element to allocate (bytes)
                        ;   R4 = Virtual address where we are to start allocation
                        ;   R5 = Physical address where we are to start allocation
                        ;
                        ;  Outputs:
                        ;   R4 = New virtual address
                        ;   R5 = New physical address
                        ;   All registers are preserved.
                        ;
000476  010246          ALLOC:  MOV     R2,-(SP)
                        ;
                        ;  Zero the space we are allocating
                        ;
000500  010102                  MOV     R1,R2           ;Get # bytes to allocate
000502  006202                  ASR     R2              ;Get # words to allocate
000504                          MAPTO   R5              ;Map to area we are allocating
000510  005024          1$:     CLR     (R4)+           ;Zero the area
000512  077202                  SOB     R2,1$
                        ;
                        ;  If we are getting too near the end of the par 5 region, reset phys addr
                        ;
000514  020427  017644          CMP     R4,#VPAR5+8100. ;Are we getting near end of par 5 region?
000520  101412                  BLOS    9$              ;Br if not
                        ;
                        ;  We must advance physical address and reset virtual address
                        ;
000522  162704  0000000         SUB     #VPAR5,R4       ;Remove par 5 virtual address bias
000526  010402                  MOV     R4,R2           ;Get virtual address within 8Kb region
000530  072227  177772          ASH     #-6,R2          ;Get # blocks allocated within region
000534  060205                  ADD     R2,R5           ;Advance physical base address
000536  042704  177700          BIC     #^C<77>,R4      ;Leave only byte-within-block virtual addr
000542  062704  0000000         ADD     #VPAR5,R4       ;Rebias virtual address
                        ;
                        ;  Finished
                        ;
000546  012602          9$:     MOV     (SP)+,R2
000550  000207                  RETURN
```

```
                                        .SBTTL  DOOPAP -- EMT to open shared file with access protection
                                ;-------------------------------------------------------------------------
                                ;  The DOOPAP entry point is the newer form of the shared file open EMT
                                ;  which functions like the simple shared file open except it also
                                ;  supports an access protection code stored in the 2nd word of the
                                ;  EMT argument block.  See comments at the front of this module for
                                ;  a description of the format of the access codes.
                                ;
                                ;  Format of the EMT argument block:
                                ;
                                ;       .BYTE   chan,125
                                ;       .WORD   access_code
                                ;
000552  013701  0000020         DOOPAP: MOV     @#EMTBLK+2,R1   ;Get access code
000556  000402                          BR      OPNCOM          ;Enter common open code
```

```
                                        .SBTTL  DOCOPN -- EMT to open channel to shared file
                                ;-------------------------------------------------------------------------
                                ;  This (obsolete) form of the shared file open EMT does not provide for
                                ;  an access code so shared update access is assumed.
                                ;
000560  012701  000005          DOCOPN: MOV     #5,R1           ;Get shared update access code
                                ;
                                ;  Enter common open code.  R1 = Open access code.
                                ;  See if this channel is already opened to a shared file.
                                ;
000564  004767  003564          OPNCOM: CALL    FNDCDB          ;Is channel opened to a shared file?
000570  103402                          BCS     1$              ;Br if not
                                ;
                                ;  This channel is already opened to a shared file.
                                ;  Close channel before reopening.
                                ;
000572  004767  003110                  CALL    CLSCDB          ;Close shared file channel
                                ;
                                ;  Make sure a .LOOKUP or .ENTER has been done to a file.
                                ;
000576  013700  000000G         1$:     MOV     @#CHNADR,R0     ;Point to channel descriptor block
000602  032760  000000G 000000G        BIT     #CS$OPN,C.CSW(R0);Has this channel been opened?
000610  001004                          BNE     2$              ;Br if yes
                                ;
                                ;  Error -- Channel is not open to a file.
                                ;
000612  012700  000001                  MOV     #EC1,R0         ;Return error code 1
000616  000137  000000G                 JMP     @#SETERR
                                ;
                                ;  Channel is open to a file.
                                ;  Try to get a free Channel Descriptor Block (CDB).
                                ;
000622  016704  177236          2$:     MOV     FCFREE,R4       ;Point to free CDB
000626  001004                          BNE     3$              ;Br if there is a free CDB
000630  012700  000002                  MOV     #EC2,R0         ;Error 2 if no free CDB's
000634  000137  000000G                 JMP     @#SETERR
000640  016705  177222          3$:     MOV     FCFREE+2,R5     ;Get PAR pointer
000644  005337  000000G                 DEC     @#NUMCDB        ;One fewer free CDB's
000650                                  MAPTO   R5              ;Map to the CDB
000654  016467  000000G 177202        MOV     FC$CLK(R4),FCFREE ;Set new free list pointer
000662  016467  000002G 177176        MOV     FC$CLK+2(R4),FCFREE+2 ;New free-list PAR pointer
                                ;
                                ;  We have a free CDB (virtual address in R4, PAR pointer in R5).
                                ;  Add this CDB to the list of CDB's for this job and init the CDB.
                                ;
000670  004767  001076                  CALL    INICDB          ;Initialize the CDB
                                ;
                                ;  Search for FDB for file being opened
                                ;
000674  004767  003546                  CALL    FNDFDB          ;Try to locate FDB for file
000700  103051                          BCC     4$              ;Br if found the FDB
                                ;
                                ;  File is not currently opened (as a shared file).
                                ;  Try to find a free FDB.
                                ;
000702  016702  177146                  MOV     FFFREE,R2       ;Is there a free FDB?
000706  001006                          BNE     5$              ;Br if yes
```

```
                                        ;
                                        ;  There are no free FDB's
                                        ;
000710  004767  002772                  CALL    CLSCDB          ;Release the CDB we got for the channel
000714  012700  000003                  MOV     #EC3,R0         ;Return error 3
000720  000137  000000G                 JMP     @#SETERR
                                        ;
                                        ;  There is a free FDB.
                                        ;  Initially zero the FDB.
                                        ;
000724  016703  177126          5$:     MOV     FFFREE+2,R3     ;Get PAR pointer to FDB
000730                                  MAPTO   R3              ;Map to the new FDB
000734  016267  000000G 177112          MOV     FF$FLK(R2),FFFREE ;Remove FDB from free list
000742  016267  000002G 177106          MOV     FF$FLK+2(R2),FFFREE+2
000750  010201                          MOV     R2,R1           ;Get virtual address of FDB
000752  012700  000000C                 MOV     #FF$$SZ/2,R0    ;Get # words to zero
000756  005021                  7$:     CLR     (R1)+           ;Zero the entire FDB
000760  077002                          SOB     R0,7$
                                        ;
                                        ;  Add the new FDB to the linked list of active FDB's
                                        ;
000762  016762  177072  000000G         MOV     FFHEAD,FF$FLK(R2),Add FDB to active list
000770  016762  177066  000002G         MOV     FFHEAD+2,FF$FLK+2(R2)
000776  010267  177056                  MOV     R2,FFHEAD       ;Put new FDB at head of active list
001002  010367  177054                  MOV     R3,FFHEAD+2
                                        ;
                                        ;  Initialize the FDB
                                        ;
001006  016762  177074  000000G         MOV     CURFID,FF$FID(R2);Set dev # in low byte, unit # in high byte
001014  016762  177070  000002G         MOV     CURFID+2,FF$FID+2(R2);Set starting block number for file
001022  000417                          BR      8$
                                        ;
                                        ;  An FDB already exists for this file.
                                        ;  See if access protection set by other users allows us to
                                        ;  access the file.
                                        ;
001024  004767  001102          4$:     CALL    CHKACC          ;See if we can access this file
001030  103406                          BCS     10$             ;Br if access is not allowed
001032  005737  000000G                 TST     @#NUMDCD        ;Are we doing data caching?
001036  001411                          BEQ     8$              ;Br if not
001040  004767  004662                  CALL    CLSDCD          ;Flush all data cache entries for file
001044  000406                          BR      8$
                                        ;
                                        ;  Error: Access protection disallows our access to this file.
                                        ;
001046  004767  002634          10$:    CALL    CLSCDB          ;Release the CDB we got
001052  012700  000004                  MOV     #EC4,R0         ;Return error code 4
001056  000137  000000G                 JMP     @#SETERR
                                        ;
                                        ;  Access to the file is allowed.
                                        ;  Make linkage connections between CDB and FDB.
                                        ;
001062                          8$:     MAPTO   R3              ;Map to FDB
001066  016200  000000G                 MOV     FF$CDB(R2),R0   ;Get pointer to current CDB at head of list
001072  016201  000002G                 MOV     FF$CDB+2(R2),R1
001076  010462  000000G                 MOV     R4,FF$CDB(R2)   ;Set new CDB as head of list for file
001102  010562  000002G                 MOV     R5,FF$CDB+2(R2)
```

```
    115 001106                                          MAPTO   R5                  ;Map to our CDB
    116 001112  010064  000000G                         MOV     R0,FC$FLK(R4)       ;Make new CDB point to old CDB list head
    117 001116  010164  000002G                         MOV     R1,FC$FLK+2(R4)
    118 001122  010264  000000G                         MOV     R2,FC$FDB(R4)       ;Make CDB point to FDB
    119 001126  010364  000002G                         MOV     R3,FC$FDB+2(R4)
    120                                 ;
    121                                 ;  Finished opening the file
    122                                 ;
    123 001132  000137  000000G                 JMP     @#EMTXIT
```

```
                                        .SBTTL  SFSVST -- EMT to save channel status (.SAVESTATUS)
                                ;-----------------------------------------------------------------------
                                ;  This routine is called when a .SAVESTATUS EMT is executed.
                                ;  It checks to see if the channel is currently opened to a shared file
                                ;  and if so marks the CDB for the channel as suspended.
                                ;
001136  010446                  SFSVST: MOV     R4,-(SP)
001140  010546                          MOV     R5,-(SP)
001142  013746  000000G                 MOV     @#KPAR5,-(SP)
                                ;
                                ;  See if this channel is currently opened to a shared file
                                ;
001146  004767  003202                  CALL    FNDCDB          ;See if channel is opened to a shared file
001152  103405                          BCS     9$              ;Br if not
                                ;
                                ;  Channel is opened to a shared file.
                                ;  Mark CDB as suspended.
                                ;
001154                                  MAPTO   R5              ;Map to the CDB
001160  152764  000000G 000000G       BISB    #FL$SPN,FC$FLG(R4) ;Mark CDB as suspended
                                ;
                                ;  Finished
                                ;
001166  012637  000000G         9$:     MOV     (SP)+,@#KPAR5
001172  012605                          MOV     (SP)+,R5
001174  012604                          MOV     (SP)+,R4
001176  000207                          RETURN
```

```
                                        .SBTTL  SFRSST -- EMT to reopen a channel (.REOPEN)
                                ;-----------------------------------------------------------------------
                                ;  This routine is called when a .REOPEN EMT is executed.  It checks to see
                                ;  if the file being reopened is a shared file.
                                ;
001200  010246                  SFRSST: MOV     R2,-(SP)
001202  010346                          MOV     R3,-(SP)
001204  010446                          MOV     R4,-(SP)
001206  010546                          MOV     R5,-(SP)
001210  013746  000000G                 MOV     @#KPAR5,-(SP)
                                ;
                                ;  See if an FDB exists for the file being reopened
                                ;
001214  004767  003226                  CALL    FNDFDB          ;Search for an FDB for this file
001220  103466                          BCS     9$              ;Br if no FDB
                                ;
                                ;  We found an FDB for the file.
                                ;  First check to see if there is a suspended CDB for this user that has
                                ;  the same channel number as channel being reopened.
                                ;
001222                                  MAPTO   R3              ;Map to the FDB
001226  016204  000000G                 MOV     FF$CDB(R2),R4   ;Get virtual address of 1st CDB for file
001232  016205  000002G                 MOV     FF$CDB+2(R2),R5 ;Get par mapping for 1st CDB
001236  001457                          BEQ     9$              ;Br if no CDB's for file (should never happen)
001240  113700  000000G                 MOVB    @#CORUSR,R0     ;Get current job index number
001244                          1$:     MAPTO   R5              ;Map to the CDB
001250  132764  000000G 000000G         BITB    #FL$SPN,FC$FLG(R4) ;Is this a suspended CDB?
001256  001407                          BEQ     2$              ;Br if not
001260  120064  000000G                 CMPB    R0,FC$UN(R4)    ;Is this CDB for this user?
001264  001004                          BNE     2$              ;Br if not
001266  123764  000000G 000000G         CMPB    @#CHNNUM,FC$CHN(R4) ;Does the channel number match?
001274  001435                          BEQ     6$              ;Br if yes -- We found the suspended CDB
001276  016405  000002G          2$:    MOV     FC$FLK+2(R4),R5 ;Get par mapping for next CDB
001302  016404  000000G                 MOV     FC$FLK(R4),R4   ;Get virtual address of next CDB
001306  001356                          BNE     1$              ;Loop if another CDB to check
                                ;
                                ;  We could not find a suspended CDB with a matching channel number.
                                ;  See if we can find any suspended CDB (in case user is reopening with
                                ;  a different channel than originally used).
                                ;
001310                                  MAPTO   R3              ;Map to the FDB
001314  016204  000000G                 MOV     FF$CDB(R2),R4   ;Get virtual address of 1st CDB for file
001320  016205  000002G                 MOV     FF$CDB+2(R2),R5 ;Get par mapping for 1st CDB
001324                          4$:     MAPTO   R5              ;Map to the CDB
001330  132764  000000G 000000G         BITB    #FL$SPN,FC$FLG(R4) ;Is this a suspended CDB?
001336  001403                          BEQ     5$              ;Br if not
001340  120064  000000G                 CMPB    R0,FC$UN(R4)    ;Is this CDB for this user?
001344  001406                          BEQ     3$              ;Br if yes -- Use this CDB
001346  016405  000002G          5$:    MOV     FC$FLK+2(R4),R5 ;Get par mapping for next CDB
001352  016404  000000G                 MOV     FC$FLK(R4),R4   ;Get virtual address of next CDB
001356  001362                          BNE     4$              ;Loop if another CDB to check
                                ;
                                ;  The file being reopened is not a shared file for this user
                                ;
001360  000406                          BR      9$
                                ;
                                ;  The file being reopened is a shared file for this user
```

```
     58                                     ;
     59 001362  113764  000000G 000000G 3$:      MOVB    @#CHNNUM,FC$CHN(R4);Set channel number in CDB
     60 001370  142764  000000G 000000G 6$:      BICB    #FL$SPN,FC$FLG(R4) ;Say CDB no longer suspended
     61                                     ;
     62                                     ;  Finished
     63                                     ;
     64 001376  012637  000000G            9$:     MOV     (SP)+,@#KPAR5
     65 001402  012605                              MOV     (SP)+,R5
     66 001404  012604                              MOV     (SP)+,R4
     67 001406  012603                              MOV     (SP)+,R3
     68 001410  012602                              MOV     (SP)+,R2
     69 001412  000207                              RETURN
```

```
                                        .SBTTL  DORLK  -- EMT to lock a block with wait
                                ;-----------------------------------------------------------------------
                                ;  This routine is called when an EMT is executed which requests that
                                ;  a block be locked and the job suspended if the block is already locked.
                                ;
001414                          DORLK:
                                ;
                                ;  See if this channel is opened to a shared file
                                ;
001414  004767  002734                  CALL    FNDCDB          ;Try to find a CDB for this channel
001420  103004                          BCC     1$              ;Br if we found a CDB
                                ;
                                ;  This channel is not open to a shared file.  Return error code 1.
                                ;
001422  012700  000001                  MOV     #EC1,R0         ;Return error code 1
001426  000137  000000G                 JMP     @#SETERR
                                ;
                                ;  This channel is opened to a shared file.
                                ;  Get number of the block we want to lock.
                                ;
001432  013701  000020          1$:     MOV     @#EMTBLK+2,R1   ;Get # of block to lock
001436  005201                          INC     R1              ;Bias the block number
                                ;
                                ;  Try to lock the requested block
                                ;
001440  004767  000636                  CALL    TRYLK           ;Try to lock the block
001444  103007                          BCC     2$              ;Br if we were able to lock the block
001446  005700                          TST     R0              ;Is block locked by someone else?
001450  001402                          BEQ     3$              ;Br if yes
001452  000137  000000G                 JMP     @#SETERR        ;Some other error
                                ;
                                ;  The block is locked by someone else.
                                ;  Suspend our job until the block is unlocked.
                                ;
001456  004767  001260          3$:     CALL    LKWAIT          ;Wait for block to be unlocked
001462  000763                          BR      1$              ;Now go try again
                                ;
                                ;  We successfully locked the block
                                ;
001464  000137  000000G         2$:     JMP     @#EMTXIT
```

```
                                        .SBTTL  DOTLK  -- EMT to lock a block with status
                                ;-------------------------------------------------------------------------
                                ;  This routine is jumped to when an EMT is executed to try to lock a block
                                ;  with a status code being returned is the block is not available.
                                ;
001470                          DOTLK:
                                ;
                                ;  See if this channel is opened to a shared file
                                ;
001470  004767  002660                  CALL    FNDCDB          ;See if channel is opened to a shared file
001474  103004                          BCC     1$              ;Br if channel is opened to a shared file
001476  012700  000001                  MOV     #EC1,R0         ;Return error code 1 if not opened to shared
001502  000137  0000000                 JMP     @#SETERR
                                ;
                                ;  Try to lock the requested block.
                                ;
001506  013701  0000020         1$:     MOV     @#EMTBLK+2,R1   ;Get # of block to lock
001512  005201                          INC     R1              ;Bias the block number
001514  004767  000562                  CALL    TRYLK           ;Try to lock the block
001520  103402                          BCS     2$              ;Br if unable to lock the block
                                ;
                                ;  We successfully locked the block
                                ;
001522  000137  0000000                 JMP     @#EMTXIT
                                ;
                                ;  The block is not available
                                ;
001526  005700                  2$:     TST     R0              ;Block locked by another job?
001530  001017                          BNE     3$              ;Br if some other error
001532  012700  000003                  MOV     #EC3,R0         ;Assume single block locked
001536                                  MAPTO   R5              ;Map to the CDB
001542  016402  0000006                 MOV     FC$FDB(R4),R2   ;Get pointer to FDB for file
001546                                  MAPTO   FC$FDB+2(R4)    ;Map to the FDB
001554  132762  0000006 0000006         BITB    #FT$EFL,FF$FLG(R2) ;Is the entire file locked?
001562  001402                          BEQ     3$              ;Br if not
001564  012700  000004                  MOV     #EC4,R0         ;Return error code 4
001570  000137  0000006           3$:   JMP     @#SETERR
```

```
                                        .SBTTL  DOCULK -- EMT to unlock all blocks for a channel
                                ;-------------------------------------------------------------------
                                ;  This routine is called by the EMT which is used to unlock all locked
                                ;  blocks for a channel.
                                ;
001574                          DOCULK:
                                ;
                                ;  Make sure this channel is opened to a shared file
                                ;
001574  004767  002554                  CALL    FNDCDB          ;Locate CDB for shared file
001600  103004                          BCC     1$              ;Br if found the CDB
001602  012700  000001                  MOV     #EC1,R0         ;Return error 1 if not opened to shared file
001606  000137  000000G                 JMP     @#SETERR
                                ;
                                ;  Unlock all of the blocks for this channel
                                ;
001612  004767  001442          1$:     CALL    CHNULK          ;Unlock all blocks for the channel
                                ;
                                ;  Finished
                                ;
001616  000137  000000G                 JMP     @#EMTXIT
```

```
                        .SBTTL  DOULK1 -- EMT to unlock a single block
                ;-----------------------------------------------------------------------
                ;  This routine is executed in response to the EMT which unlocks a
                ;  single block.
                ;
001622          DOULK1:
                ;
                ;  Make sure this channel is opened to a shared file
                ;
001622  004767  002526          CALL    FNDCDB          ;Locate CDB for this channel
001626  103004                  BCC     1$              ;Br if channel is opened to a shared file
001630  012700  000001          MOV     #EC1,R0         ;Error code 1 if not opened to shared file
001634  000137  000000G         JMP     @#SETERR
                ;
                ;  If we currently have the entire file locked, unlock the file now.
                ;
001640          1$:     MAPTO   R5              ;Map to the CDB
001644  132764  000000G 000000G BITB    #FL$EFL,FC$FLG(R4);Do we have the entire file locked?
001652  001004                  BNE     4$              ;Br if yes
                ;
                ;  Get the number of the block to unlock
                ;
001654  013700  000020G 3$:     MOV     @#EMTBLK+2,R0   ;Get # of block to unlock
001660  005200                  INC     R0              ;Bias the block number
001662  001003                  BNE     2$              ;Br if unlocking a single block
                ;
                ;  Block number was -1, unlock all of the locked blocks.
                ;
001664  004767  001370  4$:     CALL    CHNULK          ;Unlock all locked blocks
001670  000414                  BR      9$
                ;
                ;  Unlock a single block
                ;
001672  010403          2$:     MOV     R4,R3           ;Get virtual address of CDB
001674  062703  000000G         ADD     #FC$LBN,R3      ;Point to start of block # vector in CDB
001700  013702  000000G         MOV     @#VMLBLK,R2     ;Get # locked block entries in CDB
001704  020023          5$:     CMP     R0,(R3)+        ;See if block is in lock list
001706  001402                  BEQ     6$              ;Br if it is
001710  077203                  SOB     R2,5$           ;Keep looking
001712  000403                  BR      9$              ;Specified block is not locked
001714  004767  001434  6$:     CALL    UNLOCK          ;Unlock the block
001720  005043                  CLR     -(R3)           ;Remove block # from lock table
                ;
                ;  Finished
                ;
001722  000137  000000G 9$:     JMP     @#EMTXIT
```

```
                                        .SBTTL  DOSFCK -- EMT to check for shared file modification
                                ;-----------------------------------------------------------------------
                                ;  This EMT is used to check to see if any other users have written
                                ;  to the shared file specified by our channel #.
                                ;  If no writes by other users have occurred since the last time
                                ;  this EMT was executed, no error is returned by the EMT.
                                ;  If writes have been performed to the file, an error status of 2 is
                                ;  returned.
                                ;
001726                          DOSFCK:
                                ;
                                ;  Make sure this channel is opened to a shared file.
                                ;
001726  004767  002422                CALL    FNDCDB          ;Find CDB for this channel
001732  103415                      BCS     9$              ;Br if channel not opened to shared file
                                ;
                                ;  See if file-modification flag set in our CDB
                                ;
001734                                MAPTO   R5              ;Map to our CDB
001740  132764  000000G 000000G      BITB    #FL$ACT,FC$FLG(R4) ;Has the file been modified by others?
001746  001407                      BEQ     9$              ;Br if not
                                ;
                                ;  The file has been modified
                                ;
001750  142764  000000G 000000G      BICB    #FL$ACT,FC$FLG(R4) ;Reset the modification flag
001756  012700  000002              MOV     #EC2,R0         ;Return error code 2
001762  000137  000000G             JMP     @#SETERR
                                ;
                                ;  The file has not been modified
                                ;
001766  000137  000000G         9$:     JMP     @#EMTXIT
```

```
                                        .SBTTL  INICDB -- Initialize a new CDB
                                ;-----------------------------------------------------------------------
                                ;  Add a new CDB to the list of CDB's for the current job and initialize
                                ;  the CDB.
                                ;
                                ;  Inputs:
                                ;   R4 = Virtual address of CDB.
                                ;   R5 = PAR pointer for CDB.
                                ;   R1 = Open access mode flags.
                                ;
001772  010146                  INICDB: MOV     R1,-(SP)
001774  010246                          MOV     R2,-(SP)
001776  013746  000000G                 MOV     @#KPAR5,-(SP)
                                ;
                                ;  Initialize the CDB
                                ;
002002                                  MAPTO   R5              ;Map to the CDB
002006  010402                          MOV     R4,R2           ;Get pointer to CDB
002010  013700  000000G                 MOV     @#VMLBLK,R0     ;Get # locked block entries
002014  062700  000000C                 ADD     #FC$LBN/2,R0    ;Add size of rest of entry
002020  005022                  1$:     CLR     (R2)+           ;Initially zero the entire CDB
002022  077002                          SOB     R0,1$
002024  113764  000000G 000000G         MOVB    @#CORUSR,FC$UN(R4);Store job number into CDB
002032  113764  000000G 000000G         MOVB    @#CHNNUM,FC$CHN(R4);Store channel number
002040  032701  000400                  BIT     #400,R1         ;Should we suppress data caching?
002044  001403                          BEQ     2$              ;Br if not
002046  152764  000000G 000000G         BISB    #FL$NDC,FC$FLG(R4);Set flag in CDB to suppress data caching
002054  042701  177770          2$:     BIC     #^C<7>,R1       ;Mask out all but protection code
002060                                  .ADDR   #ACCMSK,R0      ;Get address of access mask table
002066  060100                          ADD     R1,R0           ;Point to correct entry in table
002070  111064  000000G                 MOVB    (R0),FC$ACC(R4) ;Set access protection flags
                                ;
                                ;  Add CDB to linked list of CDB's for current job.
                                ;
002074  013764  000000G 000000G         MOV     @#JCDB,FC$CLK(R4);Make new CDB point to rest of list
002102  013764  000002G 000002G         MOV     @#JCDB+2,FC$CLK+2(R4)
002110  010437  000000G                 MOV     R4,@#JCDB       ;Make list head of job point to new CDB
002114  010537  000002G                 MOV     R5,@#JCDB+2
                                ;
                                ;  Finished
                                ;
002120  012637  000000G                 MOV     (SP)+,@#KPAR5
002124  012602                          MOV     (SP)+,R2
002126  012601                          MOV     (SP)+,R1
002130  000207                          RETURN
```

```
                                        .SBTTL  CHKACC -- Check for file open access conflicts
                                ;-----------------------------------------------------------------------
                                ;  CHKACC is called during a file open operation to determine if the
                                ;  access protection set up by other users allows us to access the file.
                                ;
                                ;  Inputs:
                                ;   R2,R3 = Pointer to FDB for file being opened.
                                ;   R1 = File access code as provided in open argument block.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> File access is ok.
                                ;   C-flag set     ==> File access denied.
                                ;
002132  010146                  CHKACC: MOV     R1,-(SP)
002134  010446                          MOV     R4,-(SP)
002136  010546                          MOV     R5,-(SP)
002140  013746  000000G                 MOV     @#KPAR5,-(SP)   ;Preserve mapping
                                ;
                                ;  Convert access code to a bit mask
                                ;
002144                                  .ADDR   #ACCBIT,R0      ;Point to bit mask vector
002152  042701  177770                  BIC     #^C<7>,R1       ;Clear all but access flags
002156  060100                          ADD     R1,R0           ;Point to cell of interest
002160  111000                          MOVB    (R0),R0         ;Get bit mask
                                ;
                                ;  Begin loop which will examine each CDB on list for file
                                ;
002162                                  MAPTO   R3              ;Map to the FDB
002166  016204  000000G                 MOV     FF$CDB(R2),R4   ;Get virtual address of 1st CDB for file
002172  016205  000002G                 MOV     FF$CDB+2(R2),R5 ;Get PAR mapping for 1st CDB
002176  001412                          BEQ     3$              ;Br if there are no CDB's
                                ;
                                ;  See if there is an access conflict with this CDB
                                ;
002200                          1$:     MAPTO   R5              ;Map to the CDB
002204  130064  000000G                 BITB    R0,FC$ACC(R4)   ;Check for access conflict
002210  001007                          BNE     2$              ;Br if there is a conflict
                                ;
                                ;  No conflict with this CDB.  See if there are more to check.
                                ;
002212  016405  000002G                 MOV     FC$FLK+2(R4),R5 ;Get Par mapping for next CDB
002216  016404  000000G                 MOV     FC$FLK(R4),R4   ;Get pointer to next CDB
002222  001366                          BNE     1$              ;Br if there are more CDB's to check
                                ;
                                ;  There is no access conflict.
                                ;
002224  000241                  3$:     CLC                     ;Signal success on return
002226  000401                          BR      9$
                                ;
                                ;  There is an access conflict
                                ;
002230  000261                  2$:     SEC                     ;Signal error on return
                                ;
                                ;  Finished
                                ;
002232  012637  000000G         9$:     MOV     (SP)+,@#KPAR5   ;Restore par mapping
002236  012605                          MOV     (SP)+,R5
```

```
     58 002240  012604                          MOV     (SP)+,R4
     59 002242  012601                          MOV     (SP)+,R1
     60 002244  000207                          RETURN
```

```
                                        .SBTTL  SFCLS  -- Close a shared file channel
                                ;-----------------------------------------------------------------------
                                ;  The SFCLS subroutine is called from the close channel routine to see
                                ;  if the channel is opened to a shared file.
                                ;
002246  010446                  SFCLS:  MOV     R4,-(SP)
002250  010546                          MOV     R5,-(SP)
002252  013746  000000G                 MOV     @#KPAR5,-(SP)
                                ;
                                ;  See if this channel is open to a shared file
                                ;
002256  004767  002072                  CALL    FNDCDB          ;Try to find CDB for this channel
002262  103402                          BCS     9$              ;Br if channel not opened to shared file
                                ;
                                ;  This channel is opened to a shared file.
                                ;  Close the CDB.
                                ;
002264  004767  001416                  CALL    CLSCDB          ;Close this CDB
                                ;
                                ;  Finished
                                ;
002270  012637  000000G         9$:     MOV     (SP)+,@#KPAR5
002274  012605                          MOV     (SP)+,R5
002276  012604                          MOV     (SP)+,R4
002300  000207                          RETURN
```

```
                                        .SBTTL  TRYLK  -- Try to lock a block
                                ;-----------------------------------------------------------------------
                                ;  TRYLK is called to attempt to lock a block.
                                ;  A biased block number of 0 (original block number 177777) represents
                                ;  a request to lock the entire file.
                                ;
                                ;  Inputs:
                                ;   R1 = Bias block number.
                                ;   R4,R5 = Pointer to CDB for channel locking block.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Block successfully locked.
                                ;   C-flag set ==> Block could not be locked...
                                ;      R0 = 0 ==> Block locked by another user.
                                ;      R0 = 2 ==> User has attempted to lock too many blocks.
                                ;
002302  010246                  TRYLK:  MOV     R2,-(SP)
002304  010346                          MOV     R3,-(SP)
002306  013746  000000G                 MOV     @#KPAR5,-(SP)
002312  010146                          MOV     R1,-(SP)        ;Save block # on top of stack
                                ;
                                ;  If we already have the entire file locked, then this block is locked too
                                ;
002314                                  MAPTO   R5              ;Map to the CDB
002320  132764  000000G 000000G        BITB    #FL$EFL,FC$FLG(R4);Do we have entire file locked?
002326  001116                          BNE     19$             ;Br if yes -- Block is already locked
                                ;
                                ;  If this is a request to lock the entire file, unlock any blocks
                                ;  we currently have locked.
                                ;
                                        TST     (SP)            ;Request to lock entire file?
                                        BNE     1$              ;Br if not
                                        CALL    CHNULK          ;Unlock any blocks we have locked already
                                        BR      8$
                                ;
                                ;  See if we already have the block locked
                                ;
002342  105764  000000G        1$:     TSTB    FC$NLB(R4)      ;Does this CDB have any blocks locked?
002346  001410                          BEQ     8$              ;Br if not
002350  010402                          MOV     R4,R2           ;Get pointer to CDB
002352  062702  000000G                 ADD     #FC$LBN,R2      ;Point to locked-block list in CDB
002356  013703  000000G                 MOV     @#VMLBLK,R3     ;Get # locked-block entries in list
002362  021622                  9$:     CMP     (SP),(R2)+      ;See if block is already locked by CDB
002364  001477                          BEQ     19$             ;Br if block is already locked
002366  077303                          SOB     R3,9$
                                ;
                                ;  Get pointer to FDB for file
                                ;
002370  016402  000000G         8$:     MOV     FC$FDB(R4),R2   ;Get pointer to FDB
002374  016403  000002G                 MOV     FC$FDB+2(R4),R3
                                ;
                                ;  If there are no blocks locked for this file, we can immediately
                                ;  satisy the request.
                                ;
002400                                  MAPTO   R3              ;Map to the FDB
002404  005762  000000G                 TST     FF$NLB(R2)      ;Are there any blocks locked now?
002410  001412                          BEQ     4$              ;if none locked then we can satisfy request
```

```
     58 002412  005716                          TST     (SP)            ;Is this a request to lock entire file?
     59 002414  001460                          BEQ     18$             ;Br if yes -- Can't satisfy now
     60                                 ;
     61                                 ;  This is a request to lock a single block and some blocks are locked
     62                                 ;  in the file.  If the entire file is locked, we cannot satisfy request.
     63                                 ;
     64 002416  132762  000000G 000000G        BITB    #FT$EFL,FF$FLG(R2);Is the entire file locked now?
     65 002424  001054                          BNE     18$             ;Br if entire file locked
     66                                 ;
     67                                 ;  Run down chain of CDB's for this file and see if some other
     68                                 ;  user has this block locked.
     69                                 ;
     70 002426  011601                          MOV     (SP),R1         ;Get # of block to lock for CHKLOK
     71 002430  004767  000146                  CALL    CHKLOK          ;See if any other user has this block locked
     72 002434  103450                          BCS     18$             ;Br if block locked by another user
     73                                 ;
     74                                 ;  The block is free so claim it for our channel.
     75                                 ;  (SP)=block #;  R2,R3=FDB pointer;  R4,R5=CDB pointer.
     76                                 ;
     77 002436  005716                  4$:     TST     (SP)            ;Lock entire file?
     78 002440  001017                          BNE     5$              ;Br if not
     79                                 ;
     80                                 ;  Lock entire file
     81                                 ;
     82 002442                                  MAPTO   R3              ;Map to the FDB
     83 002446  152762  000000G 000000G        BISB    #FT$EFL,FF$FLG(R2);Say entire file is locked
     84 002454  005262  000000G                 INC     FF$NLB(R2)      ;Count # locked blocks
     85 002460                                  MAPTO   R5              ;Map to the CDB
     86 002464  152764  000000G 000000G        BISB    #FL$EFL,FC$FLG(R4);Say this channel has entire file locked
     87 002472  105264  000000G                 INCB    FC$NLB(R4)      ;Say channel has a block locked
     88 002476  000432                          BR      19$             ;Finished
     89                                 ;
     90                                 ;  Lock a single block
     91                                 ;
     92 002500                          5$:     MAPTO   R5              ;Map to the CDB
     93 002504  126437  000000G 000000G        CMPB    FC$NLB(R4),@#VMLBLK ;Max # blocks already locked?
     94 002512  103015                          BHIS    6$              ;Br if yes -- Error cannot lock another
     95 002514  010401                          MOV     R4,R1           ;Get addr of CDB
     96 002516  062701  000000G                 ADD     #FC$LBN,R1      ;Point to list of block numbers
     97 002522  005721                  7$:     TST     (R1)+           ;Search for free entry in list
     98 002524  001376                          BNE     7$              ;(free entry will be zero)
     99 002526  011641                          MOV     (SP),-(R1)      ;Store new block number into list
    100 002530  105264  000000G                 INCB    FC$NLB(R4)      ;Say another block locked by channel
    101 002534                                  MAPTO   R3              ;Map to the FDB
    102 002540  005262  000000G                 INC     FF$NLB(R2)      ;Say another block locked in file
    103 002544  000407                          BR      19$             ;Finished
    104                                 ;
    105                                 ;  User is requesting to simultaneously lock more blocks than allowed
    106                                 ;
    107 002546  012700  000002          6$:     MOV     #EC2,R0         ;Return error 2
    108 002552  000261                          SEC                     ;Signal error on return
    109 002554  000404                          BR      20$
    110                                 ;
    111                                 ;  The requested block is locked by another job
    112                                 ;
    113 002556  005000                  18$:    CLR     R0              ;Return error 0
    114 002560  000261                          SEC                     ;Signal error on return
```

```
    115 002562  000401                         BR      20$
    116                                 ;
    117                                 ;  We successfully locked the block
    118                                 ;
    119 002564  000241                  19$:    CLC                     ;Signal success on return
    120                                 ;
    121                                 ;  Finished
    122                                 ;
    123 002566  012601                  20$:    MOV     (SP)+,R1
    124 002570  012637  0000000                 MOV     (SP)+,@#KPAR5
    125 002574  012603                          MOV     (SP)+,R3
    126 002576  012602                          MOV     (SP)+,R2
    127 002600  000207                          RETURN
```

```
                                        .SBTTL  CHKLOK -- See if a block is locked by another job
                                ;-------------------------------------------------------------------
                                ;  This routine is called to determine if a specific block is locked
                                ;  by another job.
                                ;
                                ;  Inputs:
                                ;   R1 = Biased block number.
                                ;   R2,R3 = Pointer to FDB for file to be checked.
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> Block is locked by another job.
                                ;   C-flag cleared ==> Block is not locked.
                                ;
002602  010146          CHKLOK: MOV     R1,-(SP)
002604  010246                  MOV     R2,-(SP)
002606  010346                  MOV     R3,-(SP)
002610  010446                  MOV     R4,-(SP)
002612  010546                  MOV     R5,-(SP)
002614  013746  000000G         MOV     @#KPAR5,-(SP)
                                ;
                                ;  Get pointer to start of CDB chain for this file
                                ;
002620                          MAPTO   R3              ;Map to FDB
002624  016204  000000G         MOV     FF$CDB(R2),R4   ;Get virtual address of 1st CDB for file
002630  001431                  BEQ     8$              ;Br if no CDB's for file
002632  016205  000002G         MOV     FF$CDB+2(R2),R5 ;Get par mapping for 1st CDB
002636  113700  000000G         MOVB    @#CORUSR,R0     ;Get our job index number
                                ;
                                ;  See if next CDB has specified block locked
                                ;
002642                  1$:     MAPTO   R5              ;Map the CDB
002646  105764  000000G         TSTB    FC$NLB(R4)      ;Does this CDB have any blocks locked?
002652  001413                  BEQ     2$              ;Br if not
002654  120064  000000G         CMPB    R0,FC$UN(R4)    ;Does this CDB belong to our job?
002660  001410                  BEQ     2$              ;Br if yes -- We only care about other jobs
002662  010402                  MOV     R4,R2           ;Get address of base of CDB
002664  062702  000000G         ADD     #FC$LBN,R2      ;Point to vector of locked block #'s
002670  013703  000000G         MOV     @#VMLBLK,R3     ;Get # locked block entries
002674  020122          3$:     CMP     R1,(R2)+        ;See if requested block is locked
002676  001410                  BEQ     7$              ;Br if yes
002700  077303                  SOB     R3,3$           ;Check all entries in CDB
                                ;
                                ;  Block not locked by this CDB, check next CDB
                                ;
002702  016405  000002G 2$:     MOV     FC$FLK+2(R4),R5 ;Get par for next CDB
002706  016404  000000G         MOV     FC$FLK(R4),R4   ;Get virtual address of next CDB
002712  001353                  BNE     1$              ;Loop if more CDB's to check
                                ;
                                ;  This block is not locked by any channel
                                ;
002714  000241          8$:     CLC                     ;Signal success on return
002716  000401                  BR      9$
                                ;
                                ;  This block is locked by another job
                                ;
002720  000261          7$:     SEC                     ;Signal failure on return
                                ;
```

```
                                        ;  Finished
                                        ;
002722  012637  0000006         9$:     MOV     (SP)+,@#KPAR5
002726  012605                         MOV     (SP)+,R5
002730  012604                         MOV     (SP)+,R4
002732  012603                         MOV     (SP)+,R3
002734  012602                         MOV     (SP)+,R2
002736  012601                         MOV     (SP)+,R1
002740  000207                         RETURN
```

```
                                        .SBTTL  LKWAIT -- Wait for a locked block
                                ;-----------------------------------------------------------------------
                                ;  LKWAIT is called to suspend the execution of the current job until
                                ;  a requested block is unlocked.
                                ;
                                ;  Inputs:
                                ;   R1 = Number of block to lock (biased by adding 1).
                                ;   R4,R5 = Pointer to CDB.
                                ;
002742  010146                  LKWAIT: MOV     R1,-(SP)
002744  010246                          MOV     R2,-(SP)
002746  010346                          MOV     R3,-(SP)
002750  010446                          MOV     R4,-(SP)
002752  010546                          MOV     R5,-(SP)
002754  013746  0000006                MOV     @#KPAR5,-(SP)
                                ;
                                ;  Get the address of the FDB for this file.
                                ;
002760                                  MAPTO   R5              ;Map to the CDB
002764  016402  0000006                MOV     FC$FDB(R4),R2   ;Get virtual address of FDB
002770  016403  0000026                MOV     FC$FDB+2(R4),R3 ;Get par mapping for FDB
                                ;
                                ;  Before we suspend the job, check to see if it has been aborted.
                                ;
002774  004737  0000006                CALL    @#CHKABT        ;See if job has been aborted
                                ;
                                ;  Get a free wait queue element.
                                ;
003000  016704  175064                  MOV     FWFREE,R4       ;Get virtual addr of free wait queue element
003004  016705  175062                  MOV     FWFREE+2,R5     ;Get par mapping
003010                                  MAPTO   R5              ;Map to the wait element
003014  016467  0000006 175046          MOV     FW$WLK(R4),FWFREE ;Remove block from free list
003022  016467  0000026 175042          MOV     FW$WLK+2(R4),FWFREE+2
                                ;
                                ;  Set up information in the wait queue element
                                ;
003030  113764  0000006 0000006         MOVB    @#CORUSR,FW$UN(R4) ;Set job number
003036  010164  0000006                 MOV     R1,FW$DBN(R4)   ;Set # of block we are waiting for
003042  005064  0000006                 CLR     FW$WLK(R4)      ;Say we are end of linked list
003046  005064  0000026                 CLR     FW$WLK+2(R4)
                                ;
                                ;  Add entry to tail of wait list for file
                                ;
003052  010200                          MOV     R2,R0           ;Get pointer to FDB
003054  010301                          MOV     R3,R1
003056  062700  000000C                 ADD     #FF$FWD-FW$WLK,R0 ;Fake pointer so FDB looks like wait block
003062                          1$:     MAPTO   R1              ;Map to the next wait block (or FDB if 1st tm)
003066  005760  0000006                 TST     FW$WLK(R0)      ;Is this the last wait block on the list?
003072  001405                          BEQ     3$              ;Br if yes
003074  016001  0000026                 MOV     FW$WLK+2(R0),R1 ;Get pointer to next wait block on list
003100  016000  0000006                 MOV     FW$WLK(R0),R0
003104  000766                          BR      1$              ;Continue looking for end of list
003106  010460  0000006         3$:     MOV     R4,FW$WLK(R0)   ;Add our wait block to end of list
003112  010560  0000026                 MOV     R5,FW$WLK+2(R0)
                                ;
                                ;  Suspend job and wait for desired block to be unlocked
                                ;
```

```
     58 003116  012700  000000G                       MOV     #S$SFWT,R0      ;Get shared-file wait state code
     59 003122  004737  000000G                       CALL    @#QNSPND        ;Change job state and suspend execution
     60                                         ;
     61                                         ;  We have been restarted.
     62                                         ;  Either the desired block was unlocked or job was aborted.
     63                                         ;  Release the wait block we used.
     64                                         ;
     65 003126                                          MAPTO   R5              ;Map to the wait block
     66 003132  016400  000000G                       MOV     FW$WLK(R4),R0   ;Get pointer to following wait block
     67 003136  016401  000002G                       MOV     FW$WLK+2(R4),R1
     68 003142  062702  000000C                       ADD     #FF$FWD-FW$WLK,R2 ;Fake pointer so FDB looks like wait block
     69 003146                                  4$:     MAPTO   R3              ;Map to next wait block (or FDB if 1st time)
     70 003152  026204  000000G                       CMP     FW$WLK(R2),R4   ;Are we next entry in list?
     71 003156  001003                                BNE     6$              ;Br if not
     72 003160  026205  000002G                       CMP     FW$WLK+2(R2),R5
     73 003164  001405                                BEQ     7$              ;Br if yes
     74 003166  016203  000002G                 6$:   MOV     FW$WLK+2(R2),R3 ;Get pointer to next wait block for file
     75 003172  016202  000000G                       MOV     FW$WLK(R2),R2
     76 003176  000763                                BR      4$              ;Continue searching
     77 003200  010062  000000G                 7$:   MOV     R0,FW$WLK(R2)   ;Relink wait-block list around us
     78 003204  010162  000002G                       MOV     R1,FW$WLK+2(R2)
     79                                         ;
     80                                         ;  Return the wait block to the free list
     81                                         ;
     82 003210                                  5$:   MAPTO   R5              ;Map to wait block being freed
     83 003214  016764  174650  000000G               MOV     FWFREE,FW$WLK(R4) ;Put on free list
     84 003222  016764  174644  000002G               MOV     FWFREE+2,FW$WLK+2(R4)
     85 003230  010467  174634                        MOV     R4,FWFREE
     86 003234  010567  174632                        MOV     R5,FWFREE+2
     87                                         ;
     88                                         ;  Finished
     89                                         ;
     90 003240  012637  000000G                       MOV     (SP)+,@#KPAR5
     91 003244  012605                                MOV     (SP)+,R5
     92 003246  012604                                MOV     (SP)+,R4
     93 003250  012603                                MOV     (SP)+,R3
     94 003252  012602                                MOV     (SP)+,R2
     95 003254  012601                                MOV     (SP)+,R1
     96 003256  000207                                RETURN
```

```
                                        .SBTTL  CHNULK -- Unlock all blocks for a channel
                                ;-----------------------------------------------------------------------
                                ;  CHNULK is called to unlock all the blocks which are locked by a
                                ;  channel.
                                ;
                                ;  Inputs:
                                ;   R4,R5 = Pointer to CDB for channel.
                                ;
003260  010246                  CHNULK: MOV     R2,-(SP)
003262  010346                          MOV     R3,-(SP)
003264  013746  000000G                 MOV     @#KPAR5,-(SP)
                                ;
                                ;  See if we have the entire file locked
                                ;
003270                                  MAPTO   R5              ;Map to the CDB
003274  132764  000000G 000000G        BITB    #FL$EFL,FC$FLG(R4);Do we have the entire file locked?
003302  001404                          BEQ     1$              ;Br if not
003304  005000                          CLR     R0              ;Say we want to unlock the entire file
003306  004767  000042                  CALL    UNLOCK          ;Unlock the file
003312  000413                          BR      9$
                                ;
                                ;  We do not have the entire file locked.
                                ;  Unlock the specific blocks that we have locked
                                ;
003314  010402                  1$:     MOV     R4,R2           ;Get pointer to CDB
003316  062702  000000G                 ADD     #FC$LBN,R2      ;Point to list of block numbers
003322  013703  000000G                 MOV     @#VMLBLK,R3     ;Get # blocks in list
003326  011200                  2$:     MOV     (R2),R0         ;Get next block number
003330  001402                          BEQ     3$              ;Br if this entry not used
003332  004767  000016                  CALL    UNLOCK          ;Unlock that block
003336  005022                  3$:     CLR     (R2)+           ;Say this block no longer locked by us
003340  077306                          SOB     R3,2$           ;Loop if more to check
                                ;
                                ;  Finished
                                ;
003342  012637  000000G         9$:     MOV     (SP)+,@#KPAR5
003346  012603                          MOV     (SP)+,R3
003350  012602                          MOV     (SP)+,R2
003352  000207                          RETURN
```

```
                                        .SBTTL  UNLOCK -- Unlock a specific block
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                ;  Unlock a specific block for a channel.
                                ;  If there are any users waiting for this block, they are restarted.
                                ;
                                ;  Inputs:
                                ;   R0 = Biased block number to be unlocked (0==>Unlock entire file).
                                ;   R4,R5 = Pointer to CDB.
                                ;
003354  010146                  UNLOCK: MOV     R1,-(SP)
003356  010246                          MOV     R2,-(SP)
003360  010346                          MOV     R3,-(SP)
003362  010446                          MOV     R4,-(SP)
003364  010546                          MOV     R5,-(SP)
003366  013746  000000G                 MOV     @#KPAR5,-(SP)
003372  010001                          MOV     R0,R1           ;Carry block number in R1
                                ;
                                ;  Decrement block lock count in CDB and FDB
                                ;
003374                                  MAPTO   R5              ;Map to the CDB
003400  105364  000000G                 DECB    FC$NLB(R4)      ;Say channel has one fewer blocks locked
003404  142764  000000G 000000G         BICB    #FL$EFL,FC$FLG(R4);Say channel does not have whole file locked
003412  016402  000000G                 MOV     FC$FDB(R4),R2   ;Get pointer to FDB
003416  016403  000002G                 MOV     FC$FDB+2(R4),R3
003422                                  MAPTO   R3              ;Map to the FDB
003426  005362  000000G                 DEC     FF$NLB(R2)      ;Say one fewer locked blocks in file
003432  142762  000000G 000000G         BICB    #FT$EFL,FF$FLG(R2);Say entire file is not locked
                                ;
                                ;  Get pointer to first wait block for this file
                                ;
003440  016204  000000G                 MOV     FF$FWD(R2),R4   ;Get pointer to 1st wait block for file
003444  016205  000002G                 MOV     FF$FWD+2(R2),R5
003450  001451                          BEQ     9$              ;Br if no waiting users
                                ;
                                ;  There are some jobs waiting for blocks in this file.
                                ;  See if we are unlocking the entire file.
                                ;
003452  005701                          TST     R1              ;Are we unlocking entire file?
003454  001022                          BNE     1$              ;Br if not
003456                                  MAPTO   R5              ;Map to the first wait block
003462  005764  000000G                 TST     FW$DBN(R4)      ;1st user waiting for entire file?
003466  001440                          BEQ     8$              ;Br if yes -- Restart him only
003470  004767  000120          3$:     CALL    RESTRT          ;Restart this user
003474  016405  000002G         4$:     MOV     FW$WLK+2(R4),R5 ;Get pointer to next wait block
003500  016404  000000G                 MOV     FW$WLK(R4),R4
003504  001433                          BEQ     9$              ;Br if hit end of list
003506                                  MAPTO   R5              ;Map to next wait block
003512  005764  000000G                 TST     FW$DBN(R4)      ;Does this job want to lock whole file?
003516  001364                          BNE     3$              ;Br if not
003520  000765                          BR      4$              ;He can't have all since we started others
                                ;
                                ;  We are unlocking a specific block
                                ;
003522  016200  000000G         1$:     MOV     FF$NLB(R2),R0   ;Get # blocks remaining locked in file
003526                          5$:     MAPTO   R5              ;Map to wait block
003532  005764  000000G                 TST     FW$DBN(R4)      ;Waiting for entire file?
003536  001003                          BNE     7$              ;Br if not
```

```
     58 003540  005700                          TST     R0              ;Is entire file free?
     59 003542  001412                          BEQ     8$              ;Br if yes
     60 003544  000403                          BR      6$
     61 003546  020164  000000G          7$:     CMP     R1,FW$DBN(R4)   ;Is he waiting for block we are freeing?
     62 003552  001406                          BEQ     8$              ;Br if yes
     63 003554  016405  000002G          6$:     MOV     FW$WLK+2(R4),R5 ;Get par mapping for next wait block
     64 003560  016404  000000G                  MOV     FW$WLK(R4),R4   ;Get pointer to next wait block
     65 003564  001360                          BNE     5$              ;Br if more wait blocks to check
     66 003566  000402                          BR      9$              ;No waiting jobs to restart
     67                                ;
     68                                ;  Restart a job
     69                                ;
     70 003570  004767  000020         8$:     CALL    RESTRT          ;Restart this waiting job
     71                                ;
     72                                ;  Finished
     73                                ;
     74 003574  012637  000000G          9$:     MOV     (SP)+,@#KPAR5
     75 003600  012605                          MOV     (SP)+,R5
     76 003602  012604                          MOV     (SP)+,R4
     77 003604  012603                          MOV     (SP)+,R3
     78 003606  012602                          MOV     (SP)+,R2
     79 003610  012601                          MOV     (SP)+,R1
     80 003612  000207                          RETURN
```

```
                                        .SBTTL  RESTRT -- Restart a job that is waiting for a lock
                                ;-----------------------------------------------------------------------
                                ;  Restart a job that is waiting for a locked block.
                                ;
                                ;  Inputs:
                                ;   R4,R5 = Pointer to wait block for job being restarted.
                                ;
003614  010146                  RESTRT: MOV     R1,-(SP)
003616  010246                          MOV     R2,-(SP)
003620  013746  000000G                 MOV     @#KPAR5,-(SP)
                                ;
                                ;  Get number of job to restart
                                ;
003624                                  MAPTO   R5              ;Map to the wait block
003630  116401  000000G                 MOVB    FW$UN(R4),R1    ;Get # of job to restart
                                ;
                                ;  Place job in high-priority execution state
                                ;
003634  004737  000000G                 CALL    @#QHIPRI        ;Place job in high-prio state
                                ;
                                ;  If the job we are restarting has the same execution state as the
                                ;  currently executing job, requeue the currently executing job
                                ;  to the tail of the execution queue in order to give the restarted
                                ;  job an opportunity to get the block we released before our
                                ;  job relocks it.
                                ;
003640  113702  000000G                 MOVB    @#CORUSR,R2     ;Get # of currently running job
003644  016200  000000G                 MOV     LSTATE(R2),R0   ;Get our execution state
003650  020061  000000G                 CMP     R0,LSTATE(R1)   ;Same as job we restarted?
003654  001007                          BNE     9$              ;Br if not
003656  126261  000000G 000000G         CMPB    LPRI(R2),LPRI(R1);Same execution priority too?
003664  001003                          BNE     9$              ;Br if not
003666  010201                          MOV     R2,R1           ;Get our job index number
003670  004737  000000G                 CALL    @#ENQTL         ;Requeue our job at tail of execution list
                                ;
                                ;  Finished
                                ;
003674  012637  000000G          9$:    MOV     (SP)+,@#KPAR5
003700  012602                          MOV     (SP)+,R2
003702  012601                          MOV     (SP)+,R1
003704  000207                          RETURN
```

```
                                .SBTTL  CLSCDB -- Close a CDB
                        ;-----------------------------------------------------------------------
                        ;  CLSCDB is called to close a CDB.
                        ;  All locked blocks are freed, the CDB is freed, and the FDB is also
                        ;  freed if this is the last CDB associated with it.
                        ;
                        ;  Inputs:
                        ;   R4,R5 = Pointer to CDB being closed.
                        ;
003706  010146          CLSCDB: MOV     R1,-(SP)
003710  010246                  MOV     R2,-(SP)
003712  010346                  MOV     R3,-(SP)
003714  013746  000000G         MOV     @#KPAR5,-(SP)
                        ;
                        ;  Unlock all blocks for this channel
                        ;
003720  004767  177334          CALL    CHNULK          ;Unlock all blocks assoc with the CDB
                        ;
                        ;  Remove CDB from CDB list for this job
                        ;
003724                  MAPTO   R5              ;Map to CDB being released
003730  016400  000000G         MOV     FC$CLK(R4),R0   ;Get pointer to following CDB
003734  016401  000002G         MOV     FC$CLK+2(R4),R1
003740  012702  00000C          MOV     #JCDB-FC$CLK,R2 ;Get fake pointer to JCDB, make like CDB
003744  026204  000000G  1$:     CMP     FC$CLK(R2),R4   ;Are we the next CDB on the list?
003750  001003                  BNE     3$              ;Br if not
003752  026205  000002G         CMP     FC$CLK+2(R2),R5
003756  001407                  BEQ     4$              ;Br if yes
003760  016203  000002G  3$:     MOV     FC$CLK+2(R2),R3 ;Get pointer to next CDB for job
003764  016202  000000G         MOV     FC$CLK(R2),R2
003770                  MAPTO   R3              ;Map to the next CDB
003774  000763                  BR      1$              ;Continue searching
003776  010062  000000G  4$:     MOV     R0,FC$CLK(R2)   ;Relink chain around us
004002  010162  000002G         MOV     R1,FC$CLK+2(R2)
                        ;
                        ;  Return CDB to free list
                        ;
004006                  MAPTO   R5              ;Map to CDB being freed
004012  016764  174046  000000G         MOV     FCFREE,FC$CLK(R4);Add CDB to free list
004020  016764  174042  000002G         MOV     FCFREE+2,FC$CLK+2(R4)
004026  010467  174032                  MOV     R4,FCFREE       ;Put element at head of free list
004032  010567  174030                  MOV     R5,FCFREE+2
004036  005237  000000G                 INC     @#NUMCDB        ;One more free CDB
                        ;
                        ;  Remove CDB from list of CDB's for file
                        ;
004042  016400  000000G         MOV     FC$FLK(R4),R0   ;Get pointer to following CDB for file
004046  016401  000002G         MOV     FC$FLK+2(R4),R1
004052  016402  000000G         MOV     FC$FDB(R4),R2   ;Get pointer to FDB
004056  016403  000002G         MOV     FC$FDB+2(R4),R3
004062  001443                  BEQ     20$             ;Br if no FDB set up
004064  010246                  MOV     R2,-(SP)        ;Save the FDB pointer
004066  010346                  MOV     R3,-(SP)
004070  062702  00000C          ADD     #FF$CDB-FC$FLK,R2 ;Fake pointer so FDB looks like a CDB
004074                  5$:     MAPTO   R3              ;Map to the next CDB (or FDB if 1st time)
004100  026204  000000G         CMP     FC$FLK(R2),R4   ;Are we next CDB on list?
004104  001003                  BNE     6$              ;Br if not
```

CLSCDB -- Close a CDB

```
     58 004106  026205  000002G                 CMP     FC$FLK+2(R2),R5
     59 004112  001405                          BEQ     7$                  ;Br if yes
     60 004114  016203  000002G          6$:    MOV     FC$FLK+2(R2),R3 ;Get pointer to next CDB
     61 004120  016202  000000G                 MOV     FC$FLK(R2),R2
     62 004124  000763                          BR      5$                  ;Continue searching
     63 004126  010062  000000G          7$:    MOV     R0,FC$FLK(R2)   ;Relink chain around us
     64 004132  010162  000002G                 MOV     R1,FC$FLK+2(R2)
     65                                 ;
     66                                 ;  Release all data cache descriptors associated with this file
     67                                 ;
     68 004136  012603                          MOV     (SP)+,R3            ;Get back pointer to the FDB
     69 004140  012602                          MOV     (SP)+,R2
     70 004142  005737  000000G                 TST     @#NUMDCD            ;Are we doing data caching?
     71 004146  001402                          BEQ     8$                  ;Br if not
     72 004150  004767  001552                 CALL    CLSDCD              ;Release data cache entries for file
     73                                 ;
     74                                 ;  See if there are any CDB's left associated with this file.
     75                                 ;
     76 004154                          8$:    MAPTO   R3                  ;Map to the FDB
     77 004160  005762  000000G                 TST     FF$CDB(R2)          ;Are there any CDB's left assoc with file?
     78 004164  001002                          BNE     20$                 ;Br if yes
     79                                 ;
     80                                 ;  We just released the last CDB associated with this file.
     81                                 ;  Free the FDB.
     82                                 ;
     83 004166  004767  000014                 CALL    FREFDB              ;Free the FDB
     84                                 ;
     85                                 ;  Finished
     86                                 ;
     87 004172  012637  000000G          20$:   MOV     (SP)+,@#KPAR5
     88 004176  012603                          MOV     (SP)+,R3
     89 004200  012602                          MOV     (SP)+,R2
     90 004202  012601                          MOV     (SP)+,R1
     91 004204  000207                          RETURN
```

```
                        .SBTTL  FREFDB -- Free a FDB
                ;-----------------------------------------------------------------------
                ;  Release a File Descriptor Block (FDB).
                ;
                ;  Inputs:
                ;   R2,R3 = Pointer to FDB being freed.
                ;
004206  010146          FREFDB: MOV     R1,-(SP)
004210  010246                  MOV     R2,-(SP)
004212  010346                  MOV     R3,-(SP)
004214  013746  000000G         MOV     @#KPAR5,-(SP)
                ;
                ;  Release the FDB
                ;
004220                          MAPTO   R3              ;Map to the FDB
004224  016200  000000G         MOV     FF$FLK(R2),R0   ;Get pointer to FDB that follows one freed
004230  016201  000002G         MOV     FF$FLK+2(R2),R1
004234                          .ADDR   #FFHEAD,R4      ;Construct fake pointer to FFHEAD that makes
004242  162704  000000G         SUB     #FF$FLK,R4      ; FFHEAD look like link within a FDB
004246  026402  000000G 1$:     CMP     FF$FLK(R4),R2   ;Are we the next FDB in list?
004252  001003                  BNE     3$              ;Br if not
004254  026403  000002G         CMP     FF$FLK+2(R4),R3
004260  001407                  BEQ     6$              ;Br if yes
004262  016405  000002G 3$:     MOV     FF$FLK+2(R4),R5 ;Get pointer to next FDB in list
004266  016404  000000G         MOV     FF$FLK(R4),R4
004272                          MAPTO   R5              ;Map to the next FDB
004276  000763                  BR      1$              ;Continue search
004300  010064  000000G 6$:     MOV     R0,FF$FLK(R4)   ;Relink list around us
004304  010164  000002G         MOV     R1,FF$FLK+2(R4)
                ;
                ;  Return the FDB to the free list
                ;
004310                  2$:     MAPTO   R3              ;Map to FDB being freed
004314  016762  173534  000000G MOV     FFFREE,FF$FLK(R2);Put us on free list
004322  016762  173530  000002G MOV     FFFREE+2,FF$FLK+2(R2)
004330  010267  173520          MOV     R2,FFFREE
004334  010367  173516          MOV     R3,FFFREE+2
                ;
                ;  Finished
                ;
004340  012637  000000G         MOV     (SP)+,@#KPAR5
004344  012603                  MOV     (SP)+,R3
004346  012602                  MOV     (SP)+,R2
004350  012601                  MOV     (SP)+,R1
004352  000207                  RETURN
```

```
                                        .SBTTL  FNDCDB -- Locate a CDB for a channel
                                ;-----------------------------------------------------------------------
                                ;  FNDCDB is called to try to find an active CDB block for the current
                                ;  user and channel.
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> Could not find CDB for channel.
                                ;   C-flag cleared ==> Found CDB for channel.
                                ;   R4,R5 = Pointer to CDB found.
                                ;
004354  013746  000000G         FNDCDB: MOV     @#KPAR5,-(SP)
                                ;
                                ;  Get pointer to first CDB for job and see if job has any CDB's
                                ;
004360  013704  000000G                 MOV     @#JCDB,R4       ;Get pointer to 1st CDB for this job
004364  001422                          BEQ     7$              ;Br if there are none
004366  013705  000002G                 MOV     @#JCDB+2,R5     ;Get par mapping for 1st CDB
004372  113700  000000G                 MOVB    @#CHNNUM,R0     ;Get channel # we are searching for
                                ;
                                ;  Search CDB chain for job
                                ;
004376                          1$:     MAPTO   R5              ;Map to the next CDB for job
004402  132764  000000G 000000G        BITB    #FL$SPN,FC$FLG(R4) ;Is this a suspended CDB?
004410  001003                          BNE     2$              ;Br if suspended
004412  120064  000000G                 CMPB    R0,FC$CHN(R4)   ;Is this the correct CDB?
004416  001407                          BEQ     8$              ;Br if yes
004420  016405  000002G         2$:     MOV     FC$CLK+2(R4),R5 ;Get pointer to next CDB for job
004424  016404  000000G                 MOV     FC$CLK(R4),R4
004430  001362                          BNE     1$              ;Loop if more CDB's to check
                                ;
                                ;  There is no CDB for this channel
                                ;
004432  000261                 7$:     SEC                     ;Signal failure on return
004434  000401                          BR      9$
                                ;
                                ;  We found a CDB for this channel
                                ;
004436  000241                 8$:     CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
004440  012637  000000G         9$:     MOV     (SP)+,@#KPAR5
004444  000207                          RETURN
```

```
                                        .SBTTL  FNDFDB -- Locate a FDB for a file
                                ;-----------------------------------------------------------------------
                                ;  FNDFDB is called to try to locate a FDB corresponding to a particular file.
                                ;  The 2-word internal identification for the file is also computed.
                                ;
                                ;  Inputs:
                                ;   CHNADR = Pointer to 5 word channel status block for the channel that
                                ;            is open to the file.
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> Could not find FDB for file.
                                ;   C-flag cleared ==> Found FDB for file.
                                ;   R2,R3 = Pointer to FDB.
                                ;   CURFID = 2 word internal file identification.
                                ;
004446  010446                  FNDFDB: MOV     R4,-(SP)
004450  013746  000000G                 MOV     @#KPAR5,-(SP)
                                ;
                                ;  Build 2-word ID which uniquely identifies the file
                                ;
004454  013700  000000G                 MOV     @#CHNADR,R0     ;Point to channel status block
004460  016003  000000G                 MOV     C.CSW(R0),R3    ;Get Channel Status Word
004464  016004  000000G                 MOV     C.SBLK(R0),R4   ;Get file starting block number
004470  042703  000000C                 BIC     #^C<CS$NMX>,R3  ;Get device index number
004474  116002  000000G                 MOVB    C.DEVQ(R0),R2   ;Get unit number
004500  042702  177770                  BIC     #^C<7>,R2       ;Extract unit number
004504  020337  000000G                 CMP     R3,@#LDDEVX     ;Is device a logical disk?
004510  001006                          BNE     1$              ;Br if not
004512  006302                          ASL     R2              ;Convert unit number to word table index
004514  016203  000000G                 MOV     LDPDEV(R2),R3   ;Get real device # and unit #
004520  066204  000000G                 ADD     LDBASE(R2),R4   ;Bias starting block number by base of LD
004524  000402                          BR      7$
004526  000302                  1$:     SWAB    R2              ;Put unit # in upper byte
004530  050203                          BIS     R2,R3           ;Form composite device ID (dev # & unit #)
004532  010367  173350          7$:     MOV     R3,CURFID       ;1st word of file ID (device # and unit #)
004536  010467  173346                  MOV     R4,CURFID+2     ;2nd word of file ID (starting block number)
                                ;
                                ;  Search for a FDB which is open to this file
                                ;
004542  016702  173312                  MOV     FFHEAD,R2       ;Get pointer to 1st active FDB
004546  001421                          BEQ     2$              ;Br if there are no active FDB's
004550  016703  173306                  MOV     FFHEAD+2,R3     ;Get par mapping
004554                          4$:     MAPTO   R3              ;Map to the FDB
004560  026267  000000G 173320          CMP     FF$FID(R2),CURFID ;Compare 1st word of file ID
004566  001004                          BNE     3$              ;Br if not this file
004570  026267  000002G 173312          CMP     FF$FID+2(R2),CURFID+2 ;Compare 2nd word of file ID
004576  001407                          BEQ     6$              ;Br if found FDB for file
004600  016203  000002G          3$:     MOV     FF$FLK+2(R2),R3 ;Get pointer to next FDB
004604  016202  000000G                 MOV     FF$FLK(R2),R2
004610  001361                          BNE     4$              ;Loop if more FDB's to check
                                ;
                                ;  Cannot find a FDB for this file
                                ;
004612  000261                  2$:     SEC                     ;Signal failure on return
004614  000401                          BR      9$
                                ;
                                ;  Found FDB for this file
```

```
     58                                         ;
     59 004616  000241                          6$:     CLC                         ;Signal success on return
     60                                         ;
     61                                         ;  Finished
     62                                         ;
     63 004620  012637  0000000                 9$:     MOV     (SP)+,@#KPAR5
     64 004624  012604                                  MOV     (SP)+,R4
     65 004626  000207                                  RETURN
```

```
                                        .SBTTL  SFWRIT -- Monitor writes to shared files
                                ;-----------------------------------------------------------------------
                                ;  SFWRIT is called each time a .WRITE EMT is executed to see if the
                                ;  write is being directed to a shared file.
                                ;  If the file is shared, the following things are done:
                                ;  1. A flag is set in the CDB's for all other users indicating the
                                ;     file has been modified.
                                ;
                                ;  Inputs:
                                ;   CHNNUM = Number of I/O channel
                                ;   EMTBLK+2 = File block number
                                ;   EMTBLK+4 = Base address of user's buffer
                                ;   EMTBLK+6 = Number of words being written
                                ;
004630  010246                 SFWRIT: MOV     R2,-(SP)
004632  010346                         MOV     R3,-(SP)
004634  010446                         MOV     R4,-(SP)
004636  010546                         MOV     R5,-(SP)
004640  013746  000000G                MOV     @#KPAR5,-(SP)
                                ;
                                ;  See if channel is opened to a shared file
                                ;
004644  004767  177504                 CALL    FNDCDB          ;See if channel is opened to a shared file
004650  103444                         BCS     9$              ;Br if not
                                ;
                                ;  This is a write to a shared file.
                                ;  Set file-modified flag in CDB's for all other jobs using this file.
                                ;  (R4,R5 = Pointer to CDB)
                                ;
004652                                  MAPTO   R5              ;Map to the CDB
004656  016402  000000G                MOV     FC$FDB(R4),R2   ;Get pointer to the FDB
004662                                  MAPTO   FC$FDB+2(R4)    ;Map to the FDB
004670  016203  000002G                MOV     FF$CDB+2(R2),R3 ;Get pointer to 1st CDB for the file
004674  016202  000000G                MOV     FF$CDB(R2),R2
004700                          3$:     MAPTO   R3              ;Map to the CDB
004704  020204                         CMP     R2,R4           ;Is the same CDB that is doing the write?
004706  001002                         BNE     1$              ;Br if not
004710  020305                         CMP     R3,R5
004712  001403                         BEQ     2$              ;Br if yes
004714  152762  000000G 000000G 1$:     BISB    #FL$ACT,FC$FLG(R2);Set flag saying file has been modified
004722  016203  000002G          2$:     MOV     FC$FLK+2(R2),R3 ;Get pointer to next CDB
004726  016202  000000G                MOV     FC$FLK(R2),R2
004732  001362                         BNE     3$              ;Br if there is another CDB
                                ;
                                ;  See if we need to check for updating the shared-file data cache
                                ;
004734  005737  000000G                TST     @#NUMDCD        ;Are we doing data caching?
004740  001410                         BEQ     9$              ;Br if not
004742                                  MAPTO   R5              ;Map to the CDB
004746  016402  000000G                MOV     FC$FDB(R4),R2   ;Get pointer to FDB
004752  016403  000002G                MOV     FC$FDB+2(R4),R3
004756  004767  000532                 CALL    DCWRT           ;Check for update to data cache
                                ;
                                ;  Finished
                                ;
004762  012637  000000G          9$:     MOV     (SP)+,@#KPAR5
004766  012605                         MOV     (SP)+,R5
```

```
     58 004770  012604                          MOV     (SP)+,R4
     59 004772  012603                          MOV     (SP)+,R3
     60 004774  012602                          MOV     (SP)+,R2
     61 004776  000207                          RETURN
```

```
                                        .SBTTL  ** Data Caching Routines **
                                ;
                                ;  Note: The code related to shared file data caching is not loaded
                                ;  unless data caching is wanted.
                                ;
005000                          DCCBAS:                         ;Start of data caching code
                                ;
                                        .SBTTL  DCRD1  -- Try to read data from cache
                                ;-----------------------------------------------------------------------
                                ;  DCRD1 is called from the .READ routine to attempt to obtain
                                ;  the data needed by the read from the data cache.
                                ;  If the desired data is in the data cache, it is transferred to the
                                ;  user's buffer.
                                ;
                                ;  Inputs:
                                ;   CHNNUM = # of I/O channel
                                ;   EMTBLK+2 = File block # where transfer starts
                                ;   EMTBLK+4 = Base address of user's buffer
                                ;   EMTBLK+6 = Number of words to be transferred
                                ;
                                ;  Outputs:
                                ;   C-flag cleared if data was found in cache
                                ;
005000  010146                  DCRD1:  MOV     R1,-(SP)
005002  010246                          MOV     R2,-(SP)
005004  010346                          MOV     R3,-(SP)
005006  010446                          MOV     R4,-(SP)
005010  010546                          MOV     R5,-(SP)
005012  013746  000000G                 MOV     @#KPAR5,-(SP)
                                ;
                                ;  See if this channel is opened to a shared file
                                ;
005016  004767  177332                  CALL    FNDCDB          ;Search for CDB for this channel
005022  103447                          BCS     9$              ;Br if this channel not opened to shared file
                                ;
                                ;  This channel is opened to a shared file.
                                ;  Get pointer to FDB for the file.
                                ;
005024                                  MAPTO   R5              ;Map to the CDB
005030  016402  000000G                 MOV     FC$FDB(R4),R2   ;Get pointer to the FDB
005034  016403  000002G                 MOV     FC$FDB+2(R4),R3
                                ;
                                ;  Set up information about the transfer
                                ;
005040  013767  000002G 173032          MOV     @#EMTBLK+2,CBLOCK;Get base block number
005046  013767  000004G 173026          MOV     @#EMTBLK+4,CBUF ;Get base address of user's buffer
005054  013767  000006G 173022          MOV     @#EMTBLK+6,CWORDS;Get # words to transfer
                                ;
                                ;  Try to find data block in the cache
                                ;
005062  005237  000000G         1$:     INC     @#DCTRD         ;Inc total # reads from shared files
005066  001002                          BNE     4$              ;Br if did not overflow
005070  004767  001510                  CALL    CLRST           ;Reset cache statistics
005074  004767  000540          4$:     CALL    FNDDCD          ;See if block is in cache
005100  103420                          BCS     9$              ;Br if block is not in cache
                                ;
                                ;  Found data block in cache -- Move data to user's buffer
```

```
                                        ;
005102  005237  000000G                 INC     @#DCCRD         ;Inc # reads satisfied by cache data
005106  004767  001304                  CALL    DCMVTU          ;Move data to user's buffer
                                        ;
                                        ;  Increment use count for cache entry
                                        ;
005112  004767  001172                  CALL    DCINCU          ;Increment use count for cache entry
                                        ;
                                        ;  See if we need more data
                                        ;
005116  005267  172756                  INC     CBLOCK          ;Increment file block number
005122  062767  001000  172752          ADD     #512.,CBUF      ;Advance buffer address
005130  162767  000400  172746          SUB     #256.,CWORDS    ;Dec # words left to be transferred
005136  003351                          BGT     1$              ;Br if more needed
                                        ;
                                        ;  Successful completion
                                        ;
005140  000241                          CLC                     ;Say read was satisfied from cache
                                        ;
                                        ;  Finished
                                        ;
005142  012637  000000G         7$:     MOV     (SP)+,@#KPAR5
005146  012605                          MOV     (SP)+,R5
005150  012604                          MOV     (SP)+,R4
005152  012603                          MOV     (SP)+,R3
005154  012602                          MOV     (SP)+,R2
005156  012601                          MOV     (SP)+,R1
005160  000207                          RETURN
```

```
                                        .SBTTL  DCRD2  -- Update cache following read
                                ;-----------------------------------------------------------------------
                                ;  DCRD2 is called following a .READ operation to see if the data
                                ;  just read should be placed in the data cache.
                                ;
                                ;  Inputs:
                                ;   CHNNUM = # of I/O channel
                                ;   EMTBLK+2 = File block number
                                ;   EMTBLK+4 = Base address of user's buffer
                                ;   EMTBLK+6 = Number of words transferred
                                ;
005162  010146                  DCRD2:  MOV     R1,-(SP)
005164  010246                          MOV     R2,-(SP)
005166  010346                          MOV     R3,-(SP)
005170  010446                          MOV     R4,-(SP)
005172  010546                          MOV     R5,-(SP)
005174  013746  0000000                 MOV     @#KPAR5,-(SP)
                                ;
                                ;  See if this channel is opened to a shared file
                                ;
005200  004767  177150                  CALL    FNDCDB          ;Search for CDB for this channel
005204  103533                          BCS     9$              ;Br if channel not open to shared file
                                ;
                                ;  See if data caching is being suppressed on this channel
                                ;
005206                                  MAPTO   R5              ;Map to the CDB
005212  132764  0000000 0000000       BITB    #FL$NDC,FC$FLG(R4);Is data caching being suppressed?
005220  001125                          BNE     9$              ;Br if yes
                                ;
                                ;  We are doing data caching for this file.
                                ;  Wait for .READ operation to finish.
                                ;
005222  004737  0000000               CALL    @#IOWAIT        ;Wait for .READ to finish
                                ;
                                ;  Set up data about the transfer
                                ;
005226  013767  0000020 172644        MOV     @#EMTBLK+2,CBLOCK;File block number
005234  013767  0000040 172640        MOV     @#EMTBLK+4,CBUF ;Buffer address
005242  013767  0000060 172634        MOV     @#EMTBLK+6,CWORDS;Number of words read
005250                                MAPTO   R5              ;Map to the CDB
005254  016402  0000000               MOV     FC$FDB(R4),R2   ;Get pointer to FDB for file
005260  016403  0000020               MOV     FC$FDB+2(R4),R3
                                ;
                                ;  See if data block is currently in cache
                                ;
005264  026727  172614  000400  3$:   CMP     CWORDS,#256.    ;Are there at least 256 words left?
005272  103453                        BLO     2$              ;Br if not -- Don't have a block left
005274  004767  000340                CALL    FNDDCD          ;See if this block is in the cache
005300  103037                        BCC     1$              ;Br if yes -- Don't need to put it there
                                ;
                                ;  This data block is not currently in the cache -- Put it there
                                ;
005302  004767  000646                CALL    NEWDCD          ;Get a new data cache descriptor block
005306  103445                        BCS     2$              ;Br if no free cache descriptor blocks
                                ;
                                ;  Initialize a new data cache descriptor block (pointer in R4,R5)
                                ;
```

```
     58 005310                                          MAPTO   R3                      ;Map to the FDB
     59 005314  016200  0000000                        MOV     FF$DCD(R2),R0           ;Get pointer to 1st DCD for file
     60 005320  016201  0000020                        MOV     FF$DCD+2(R2),R1
     61 005324  010462  0000000                        MOV     R4,FF$DCD(R2)           ;Set us as the 1st DCD for file
     62 005330  010562  0000020                        MOV     R5,FF$DCD+2(R2)
     63 005334                                          MAPTO   R5                      ;Map to the new DCD
     64 005340  010064  0000000                        MOV     R0,DC$LNK(R4)           ;Add us to chain for this FDB
     65 005344  010164  0000020                        MOV     R1,DC$LNK+2(R4)
     66 005350  010264  0000000                        MOV     R2,DC$FDB(R4)           ;Make DCD point to FDB
     67 005354  010364  0000020                        MOV     R3,DC$FDB+2(R4)
     68 005360  016764  172514  0000000               MOV     CBLOCK,DC$BLK(R4);Remember the file block number
     69 005366  012764  177777  0000000               MOV     #-1,DC$USE(R4)  ;Temporarily set use count very high
     70                                         ;
     71                                         ;  Move data from user's buffer to cache
     72                                         ;
     73 005374  004767  001114                          CALL    DCMVTC                  ;Move data to cache
     74                                         ;
     75                                         ;  See if there is more data to be moved to cache
     76                                         ;
     77 005400  005267  172474                  1$:     INC     CBLOCK                  ;Advance file block number
     78 005404  062767  001000  172470                  ADD     #512.,CBUF              ;Advance buffer address
     79 005412  162767  000400  172464                  SUB     #256.,CWORDS            ;Reduce # words left to transfer
     80 005420  003321                                  BGT     3$                      ;Br if more data remaining
     81                                         ;
     82                                         ;  Reset use counters for cache blocks just added
     83                                         ;
     84 005422                                  2$:     MAPTO   R3                      ;Map to the FDB
     85 005426  016204  0000000                        MOV     FF$DCD(R2),R4           ;Get pointer to 1st DCD for file
     86 005432  016205  0000020                        MOV     FF$DCD+2(R2),R5
     87 005436  001416                                  BEQ     9$                      ;Br if not DCD's for file
     88 005440                                  4$:     MAPTO   R5                      ;Map to the DCD
     89 005444  026427  0000000 177777                 CMP     DC$USE(R4),#-1  ;Is this one that we just added?
     90 005452  001010                                  BNE     9$                      ;Br if not
     91 005454  012764  000004  0000000               MOV     #4,DC$USE(R4)           ;Set initial use count
     92 005462  016405  0000020                        MOV     DC$LNK+2(R4),R5 ;Chain to next DCD
     93 005466  016404  0000000                        MOV     DC$LNK(R4),R4
     94 005472  001362                                  BNE     4$                      ;Br if more DCD's to check
     95                                         ;
     96                                         ;  Finished
     97                                         ;
     98 005474  012637  0000000                 9$:     MOV     (SP)+,@#KPAR5
     99 005500  012605                                  MOV     (SP)+,R5
    100 005502  012604                                  MOV     (SP)+,R4
    101 005504  012603                                  MOV     (SP)+,R3
    102 005506  012602                                  MOV     (SP)+,R2
    103 005510  012601                                  MOV     (SP)+,R1
    104 005512  000207                                  RETURN
```

```
                                        .SBTTL  DCWRT  -- Check for writes to shared file data cache
                                ;-----------------------------------------------------------------------
                                ;  DCWRT is called each time a .WRITE operation is done to a shared file.
                                ;  It checks to see if the data being written needs to be placed in the
                                ;  shared file data cache.
                                ;
                                ;  Inputs:
                                ;   R2,R3 = Pointer to FDB for file being written to.
                                ;   CHNNUM = Number of I/O channel.
                                ;   EMTBLK+2 = File block number
                                ;   EMTBLK+4 = Base address of user's buffer
                                ;   EMTBLK+6 = Number of words being written
                                ;
005514  010446                  DCWRT:  MOV     R4,-(SP)
005516  010546                          MOV     R5,-(SP)
                                ;
                                ;  Set up information about the transfer
                                ;
005520  013767  0000020 172352          MOV     @#EMTBLK+2,CBLOCK;Block number
005526  013767  0000040 172346          MOV     @#EMTBLK+4,CBUF ;Buffer address
005534  013767  0000060 172342          MOV     @#EMTBLK+6,CWORDS;Number of words being written
                                ;
                                ;  See if data block is in cache now
                                ;
005542  005237  0000000         6$:     INC     @#DCTWR         ;Inc total # writes to shared files
005546  001002                          BNE     7$              ;Br if count did not overflow
005550  004767  001030                  CALL    CLRST           ;Reset statistics
005554  004767  000060          7$:     CALL    FNDDCD          ;Is this block in the cache now?
005560  103413                          BCS     5$              ;Br if not
                                ;
                                ;  Block is in cache -- Update it
                                ;
005562  005237  0000000                 INC     @#DCCWR         ;Inc # writes that update cache
005566  026727  172312  000400         CMP     CWORDS,#256.    ;Do we have a full block of data?
005574  103003                          BHIS    4$              ;Br if yes
005576  004767  000222                  CALL    RELDCD          ;Remove the cache entry if short write
005602  000402                          BR      5$
005604  004767  000704          4$:     CALL    DCMVTC          ;Move data from user's buffer to cache
                                ;
                                ;  See if there are more blocks to update
                                ;
005610  005267  172264          5$:     INC     CBLOCK          ;Increment block number
005614  062767  001000  172260         ADD     #512.,CBUF      ;Advance buffer address
005622  162767  000400  172254         SUB     #256.,CWORDS    ;Reduce # words left
005630  003344                          BGT     6$              ;Br if some data left
                                ;
                                ;  Finished
                                ;
005632  012605                          MOV     (SP)+,R5
005634  012604                          MOV     (SP)+,R4
005636  000207                          RETURN
```

```
                                        .SBTTL  FNDDCD -- Try to find data cache descriptor for data block
                                ;-----------------------------------------------------------------------------
                                ;  FNDDCD is called to try to find a data cache descriptor for a disk
                                ;  data block that is being held by the shared file data cache.
                                ;
                                ;  Inputs:
                                ;   R2,R3 = Pointer to FDB for shared file.
                                ;   CBLOCK = Block number
                                ;
                                ;  Outputs:
                                ;   C-flag cleared if block is currently in cache.
                                ;   R4,R5 = Pointer to data cache descriptor block for entry if found.
                                ;
005640  013746  0000000         FNDDCD: MOV     @#KPAR5,-(SP)
                                ;
                                ;  Get pointer to 1st DCD for this file
                                ;
005644                                  MAPTO   R3              ;Map to the FDB
005650  016204  0000000                 MOV     FF$DCD(R2),R4   ;Get pointer to 1st DCD for file
005654  001416                          BEQ     2$              ;Br if file has no DCD's
005656  016205  0000020                 MOV     FF$DCD+2(R2),R5
005662  016700  172212                  MOV     CBLOCK,R0       ;Get file block number
                                ;
                                ;  Search for DCD for block of interest
                                ;
005666                          1$:     MAPTO   R5              ;Map to the DCD
005672  020064  0000000                 CMP     R0,DC$BLK(R4)   ;Is this DCD for the right block?
005676  001407                          BEQ     3$              ;Br if yes
005700  016405  0000020                 MOV     DC$LNK+2(R4),R5 ;Get pointer to next DCD
005704  016404  0000000                 MOV     DC$LNK(R4),R4
005710  001366                          BNE     1$              ;Br if more DCD's for this file
                                ;
                                ;  Could not find data cache entry for this block
                                ;
005712  000261                 2$:     SEC                     ;Signal failure on return
005714  000401                          BR      9$
                                ;
                                ;  Found data cache entry for block
                                ;
005716  000241                 3$:     CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
005720  012637  0000000         9$:     MOV     (SP)+,@#KPAR5
005724  000207                          RETURN
```

```
                                        .SBTTL  CLSDCD -- Release all data cache entries for a file
                                ;-----------------------------------------------------------------------
                                ;  CLSDCD is called to release all data cache entries associated with
                                ;  a shared file.
                                ;
                                ;  Inputs:
                                ;   R2,R3 = Pointer to FDB for the file
                                ;
005726  010446                  CLSDCD: MOV     R4,-(SP)
005730  010546                          MOV     R5,-(SP)
005732  013746  0000000G              MOV     @#KPAR5,-(SP)
                                ;
                                ;  Release all data cache blocks associated with the file
                                ;
005736                                  MAPTO   R3                  ;Map to the FDB
005742  016204  0000000G                MOV     FF$DCD(R2),R4       ;Point to 1st DCD
005746  016205  0000020G                MOV     FF$DCD+2(R2),R5
005752  001417                          BEQ     5$                  ;Br if no data cache descriptors for file
005754  005062  0000000G                CLR     FF$DCD(R2)          ;Say FDB has no associated cache descriptors
005760  005062  0000020G                CLR     FF$DCD+2(R2)
005764                          4$:     MAPTO   R5                  ;Map to data cache descriptor
005770  005064  0000000G                CLR     DC$FDB(R4)          ;Say data cache descriptor is free
005774  005064  0000020G                CLR     DC$FDB+2(R4)
006000  016405  0000020G                MOV     DC$LNK+2(R4),R5     ;Get pointer to next DCD
006004  016404  0000000G                MOV     DC$LNK(R4),R4
006010  001365                          BNE     4$                  ;Br if more DCD's to free
                                ;
                                ;  Finished
                                ;
006012  012637  0000000G        5$:     MOV     (SP)+,@#KPAR5
006016  012605                          MOV     (SP)+,R5
006020  012604                          MOV     (SP)+,R4
006022  000207                          RETURN
```

```
                                        .SBTTL  RELDCD -- Release a data cache entry
                                ;-----------------------------------------------------------------------
                                ;  RELDCD is called to release (deallocate) a data cache entry and
                                ;  mark it as free.
                                ;
                                ;  Inputs:
                                ;   R4,R5 = Pointer to DCD to be released.
                                ;
006024  010146                  RELDCD: MOV     R1,-(SP)
006026  010246                          MOV     R2,-(SP)
006030  010346                          MOV     R3,-(SP)
006032  013746  000000G                 MOV     @#KPAR5,-(SP)
                                ;
                                ;  Unlink DCD from linked list associated with shared file
                                ;
006036                                  MAPTO   R5              ;Map to the DCD
006042  016400  000000G                 MOV     DC$LNK(R4),R0   ;Get pointer to DCD that follows our one
006046  016401  000020G                 MOV     DC$LNK+2(R4),R1
006052  016402  000000G                 MOV     DC$FDB(R4),R2   ;Get pointer to FDB
006056  016403  000020G                 MOV     DC$FDB+2(R4),R3
006062  005064  000000G                 CLR     DC$FDB(R4)      ;Mark the DCD as free
006066  005064  000020G                 CLR     DC$FDB+2(R4)
006072  062702  000000C                 ADD     #FF$DCD-DC$LNK,R2 ;Fake pointer to link cell
006076                          1$:     MAPTO   R3              ;Map to the next DCD (or FDB if 1st time)
006102  026204  000000G                 CMP     DC$LNK(R2),R4   ;Are we next DCD on list?
006106  001003                          BNE     3$              ;Br if not
006110  026205  000020G                 CMP     DC$LNK+2(R2),R5
006114  001405                          BEQ     2$              ;Br if yes
006116  016203  000020G          3$:     MOV     DC$LNK+2(R2),R3 ;Get pointer to next DCD
006122  016202  000000G                 MOV     DC$LNK(R2),R2
006126  000763                          BR      1$              ;Go check it
006130  010062  000000G          2$:     MOV     R0,DC$LNK(R2)   ;Relink chain around us
006134  010162  000020G                 MOV     R1,DC$LNK+2(R2)
                                ;
                                ;  Finished
                                ;
006140  012637  000000G                 MOV     (SP)+,@#KPAR5
006144  012603                          MOV     (SP)+,R3
006146  012602                          MOV     (SP)+,R2
006150  012601                          MOV     (SP)+,R1
006152  000207                          RETURN
```

```
                                        .SBTTL  NEWDCD -- Locate a free data cache descriptor
                                ;-----------------------------------------------------------------------
                                ;  NEWDCD is called to find a data cache descriptor to be used for a
                                ;  new cache entry.  It finds a free entry or the least used non-free entry.
                                ;
                                ;  Outputs:
                                ;   C-flag set on return if there are no free entries.
                                ;   R4,R5 = Pointer to new entry if one was found.
                                ;
 006154  010146                 NEWDCD: MOV     R1,-(SP)
 006156  010246                         MOV     R2,-(SP)
 006160  010346                         MOV     R3,-(SP)
 006162  013746  000000G                MOV     @#KPAR5,-(SP)
                                ;
                                ;  Search for a free or least-used data cache descriptor
                                ;
 006166  013701  000000G                MOV     @#NUMDCD,R1     ;Max # cache entries we are allowed to use
 006172  016704  171676                 MOV     DCDBAS,R4       ;Point to 1st data cache descriptor
 006176  016705  171674                 MOV     DCDBAS+2,R5
 006202  012700  177777                 MOV     #-1,R0          ;Initialize use count
 006206                          1$:    MAPTO   R5              ;Map to the DCD
 006212  005764  000000G                TST     DC$FDB(R4)      ;Is this cache entry free?
 006216  001425                         BEQ     4$              ;Br if yes -- Use this entry
 006220  026400  000000G                CMP     DC$USE(R4),R0   ;Is this entry less used than any found so far
 006224  103004                         BHIS    2$              ;Br if not
 006226  010402                         MOV     R4,R2           ;Save pointer to least used entry
 006230  010503                         MOV     R5,R3
 006232  016400  000000G                MOV     DC$USE(R4),R0   ;Save associated use count
 006236  016405  000002G          2$:   MOV     DC$NXT+2(R4),R5 ;Get pointer to next DCD
 006242  016404  000000G                MOV     DC$NXT(R4),R4
 006246  077121                         SOB     R1,1$           ;Loop if we are allowed to use more
                                ;
                                ;  Finished searching through all cache descriptors.
                                ;  See if we found one we can use.
                                ;
 006250  020027  177777                 CMP     R0,#-1          ;Did we find an available one?
 006254  001002                         BNE     3$              ;Br if yes
 006256  000261                         SEC                     ;Signal failure on return
 006260  000405                         BR      9$
                                ;
                                ;  We found a cache entry that we can use
                                ;
 006262  010204                 3$:     MOV     R2,R4           ;Get pointer to entry to use
 006264  010305                         MOV     R3,R5
 006266  004767  177532                 CALL    RELDCD          ;Free the entry
 006272  000241                 4$:     CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
 006274  012637  000000G        9$:     MOV     (SP)+,@#KPAR5
 006300  012603                         MOV     (SP)+,R3
 006302  012602                         MOV     (SP)+,R2
 006304  012601                         MOV     (SP)+,R1
 006306  000207                         RETURN
```

```
                                        .SBTTL  DCINCU -- Increment use count for data cache entry
                                ;-----------------------------------------------------------------------
                                ;  DCINCU is called to increment the use counter for a data cache entry.
                                ;  If the total number of increments done for all entires reaches
                                ;  DCAGE, all of the use counts are divided by two.
                                ;
                                ;  Inputs:
                                ;   R4,R5 = Pointer to data cache descriptor which is to be incremented.
                                ;
006310  010246                 DCINCU: MOV     R2,-(SP)
006312  010346                         MOV     R3,-(SP)
006314  013746  000000G                MOV     @#KPAR5,-(SP)
                                ;
                                ;  See if it is time to divide all use counts
                                ;
006320  005237  000000G                INC     @#DCTOTU        ;Increment total use count
006324  023727  000000G 000000G        CMP     @#DCTOTU,#DCAGE ;Is it time to divide all use counts?
006332  103417                         BLO     2$              ;Br if not
                                ;
                                ;  Divide all use counts by two
                                ;
006334  005037  000000G                CLR     @#DCTOTU        ;Reset total use counter
006340  016702  171530                 MOV     DCDBAS,R2       ;Get pointer to 1st DCD
006344  016703  171526                 MOV     DCDBAS+2,R3
006350                          1$:    MAPTO   R3              ;Map to a DCD
006354  006262  000000G                ASR     DC$USE(R2)      ;Divide the use count by 2
006360  016203  000002G                MOV     DC$NXT+2(R2),R3 ;Get pointer to next DCD
006364  016202  000000G                MOV     DC$NXT(R2),R2
006370  001367                         BNE     1$              ;Loop if more to do
                                ;
                                ;  Increment use count for specified entry
                                ;
006372                          2$:    MAPTO   R5              ;Map to the entry
006376  062764  000002  000000G        ADD     #2,DC$USE(R4)   ;Increment use count
                                ;
                                ;  Finished
                                ;
006404  012637  000000G                MOV     (SP)+,@#KPAR5
006410  012603                         MOV     (SP)+,R3
006412  012602                         MOV     (SP)+,R2
006414  000207                         RETURN
```

```
                                        .SBTTL  DCMVTU -- Move data from cache to user's buffer
                                ;-----------------------------------------------------------------------
                                ;  DCMVTU is called to move data from a cache entry to the user's buffer.
                                ;
                                ;  Inputs:
                                ;   R4,R5 = Pointer to data cache descriptor
                                ;   CBUF = Virtual address of user's data buffer
                                ;   CWORDS = Number of words to move
                                ;
006416  010246                  DCMVTU: MOV     R2,-(SP)
006420  010346                          MOV     R3,-(SP)
006422  013746  000000G                 MOV     @#KPAR5,-(SP)
                                ;
                                ;  Validate the buffer address
                                ;
006426  016703  171450                  MOV     CBUF,R3         ;Get the buffer address
006432  010300                          MOV     R3,R0
006434  004737  000000G                 CALL    @#VALADW        ;Validate it
                                ;
                                ;  Map par 5 to the cache buffer
                                ;
006440                                  MAPTO   R5              ;Map to the DCD
006444                                  MAPTO   DC$PAR(R4)      ;Map par 5 to cache buffer
                                ;
                                ;  Move data from cache to user's buffer
                                ;
006452  012702  000000G                 MOV     #VPAR5,R2       ;Get virtual address of cache buffer
006456  012700  000400                  MOV     #256.,R0        ;Get max # words we can move
006462  026700  171416                  CMP     CWORDS,R0       ;Do we have that much to move?
006466  103002                          BHIS    1$              ;Br if yes
006470  016700  171410                  MOV     CWORDS,R0       ;Get # words to move
006474  012246                  1$:     MOV     (R2)+,-(SP)     ;Get word from cache buffer
006476  106623                          MTPD    (R3)+           ;Move to user's buffer
006500  077003                          SOB     R0,1$           ;Loop if more words to move
                                ;
                                ;  Finished
                                ;
006502  012637  000000G                 MOV     (SP)+,@#KPAR5
006506  012603                          MOV     (SP)+,R3
006510  012602                          MOV     (SP)+,R2
006512  000207                          RETURN
```

```
                                        .SBTTL  DCMVTC -- Move data from user's buffer to cache
                                ;-----------------------------------------------------------------------
                                ;  DCMVTC is called to move a block of data (256 words) from the user's
                                ;  buffer to a cache buffer.
                                ;
                                ;  Inputs:
                                ;   R4,R5 = Pointer to cache descriptor block
                                ;   CBUF = Virtual address of user's buffer
                                ;
006514  010246                  DCMVTC: MOV     R2,-(SP)
006516  010346                          MOV     R3,-(SP)
006520  013746  000000G                 MOV     @#KPAR5,-(SP)
                                ;
                                ;  Validate the buffer address
                                ;
006524  016703  171352                  MOV     CBUF,R3         ;Get user's buffer address
006530  010300                          MOV     R3,R0
006532  004737  000000G                 CALL    @#VALADW        ;Validate it
                                ;
                                ;  Map par 5 to the cache buffer
                                ;
006536                                  MAPTO   R5              ;Map to the DCD
006542                                  MAPTO   DC$PAR(R4)      ;Map to the cache buffer
                                ;
                                ;  Move data from user's buffer to the cache buffer
                                ;
006550  012702  000000G                 MOV     #VPAR5,R2       ;Get virtual address of cache buffer
006554  012700  000200                  MOV     #128.,R0        ;Move 128 double-words
006560  106523                  1$:     MFPD    (R3)+           ;Get a word from user's buffer
006562  012622                          MOV     (SP)+,(R2)+     ;Move to cache buffer
006564  106523                          MFPD    (R3)+           ;Get another word from user's buffer
006566  012622                          MOV     (SP)+,(R2)+     ;Move to cache buffer
006570  077005                          SOB     R0,1$           ;Loop if more to move
                                ;
                                ;  Finished
                                ;
006572  012637  000000G                 MOV     (SP)+,@#KPAR5
006576  012603                          MOV     (SP)+,R3
006600  012602                          MOV     (SP)+,R2
006602  000207                          RETURN
```

```
      1                                          .SBTTL  CLRST  -- Reset data cache statistics
      2                                ;-----------------------------------------------------------------
      3                                ;  CLRST is called to reset the usage statistics counters that
      4                                ;  accumulate data regarding data caching.
      5                                ;
      6 006604  005037  0000006       CLRST:  CLR     @#DCTRD         ;Total number of reads from shared files
      7 006610  005037  0000006               CLR     @#DCCRD         ;Number of reads satisfied by cache
      8 006614  005037  0000006               CLR     @#DCTWR         ;Total number of writes to shared files
      9 006620  005037  0000006               CLR     @#DCCWR         ;Number of writes that update cache data
     10                                ;
     11                                ;  Finished
     12                                ;
     13 006624  000207                          RETURN
     14                                ;
     15                                ;  Compute size of code related to data caching
     16                                ;
     17          001626                DCCSIZ  =       .-DCCBAS        ;Size of data caching code
     18
     19          000001                          .END
Errors detected:  0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  8510 Words  ( 34 Pages)
Size of core pool:  17920 Words  ( 70 Pages)
Operating  system:  RT-11

Elapsed time:  00:00:33.32
DK:TSLOCK,LP:TSLOCK=DK:TSLOCK/C/N:SYM
```

Cross reference table (CREF V05.05)

```
...V2    16-29    16-29    16-29    16-29    16-29    16-29    16-29    16-29    16-29    16-29    16-29    16-29
         16-29    16-29    16-29    16-29    16-29#   16-29#   16-29#   17-21    17-21    17-21    17-21    17-21
         17-21    17-21    17-21    17-21    17-21    17-21    17-21    17-21    17-21    17-21    17-21    17-21#
         17-21#   17-21#   26-18    26-18    26-18    26-18    26-18    26-18    26-18    26-18    26-18    26-18
         26-18    26-18    26-18    26-18    26-18    26-18    26-18#   26-18#   26-18#
AC$EI     3-11#    3-23
AC$EU     3-10#    3-24
AC$PI     3-9#     3-25     3-35     3-37
AC$PU     3-8#     3-26     3-37
AC$SI     3-7#     3-27     3-35     3-36     3-37     3-38
AC$SU     3-6#     3-28     3-37     3-38
ACCBIT    3-23#    17-21    17-21
ACCMSK    3-33#    16-29    16-29
ALLOC     4-29     4-43     4-59     4-74     5-24     6-16#
ALLOCL    4-25     4-39     4-55     4-70     5-15#
C.CSW     1-42     8-22     28-22
C.DEVQ    1-42     28-25
C.SBLK    1-42     28-23
CBLOCK    3-48#    30-45*   30-68*   31-37*   31-68    31-77*   32-19*   32-42*   33-22
CBUF      3-49#    30-46*   30-69*   31-38*   31-78*   32-20*   32-43*   38-16    39-16
CHKABT    1-38     21-25
CHKACC    8-94     17-14#
CHKLOK   19-71     20-14#
CHNADR    1-34     8-21     28-21
CHNNUM    1-35     10-31    10-59    16-24    27-18
CHNULK   13-17     14-29    19-33    22-9#    25-17
CLRST    30-53     32-27    40-6#
CLSCDB    1-70     8-17     8-61     8-103    18-18    25-10#
CLSDCD    8-98     25-72    34-9#
CORUSR    1-34     10-25    16-23    20-27    21-37    24-27
CS$NMX    1-43     28-24
CS$OPN    1-43     8-22
CURFID    3-54#    8-86     8-87     28-35*   28-36*   28-44    28-46
CWORDS    3-50#    30-47*   30-70*   31-39*   31-46    31-79*   32-21*   32-34    32-44*   38-29    38-31
DC$$SZ    1-34     4-67
DC$BLK    1-30     31-68*   33-27
DC$FDB    1-37     31-66*   31-67*   34-22*   34-23*   35-19    35-20    35-21*   35-22*   36-22
DC$LNK    1-37     31-64*   31-65*   31-92    31-93    33-29    33-30    34-24    34-25    35-17    35-18    35-23
          35-25    35-27    35-29    35-30    35-32*   35-33*
DC$NXT    1-36     4-71*    4-72*    4-84     4-85     36-29    36-30    37-27    37-28
DC$PAR    1-36     4-82*    38-23    39-23
DC$USE    1-37     31-69*   31-89    31-91*   36-24    36-28    37-26*   37-34*
DCAGE     1-31     37-17
DCCBAS   30-6#     40-17
DCCRD     1-30     30-59*   40-7*
DCCSIZ    1-23     40-17#
DCCWR     1-31     32-33*   40-9*
DCDBAS    3-47#    4-65*    4-66*    4-78     4-79     36-18    36-19    37-23    37-24
DCINCU   30-64     37-10#
DCMVTC   31-73     32-38    39-10#
DCMVTU   30-60     38-10#
DCRD1     1-71     30-24#
DCRD2     1-72     31-12#
DCTOTU    1-31     37-16*   37-17    37-22*
DCTRD     1-30     30-51*   40-6*
DCTWR     1-30     32-25*   40-8*
```

Cross reference table (CREF V05.05)

```
DCWRT    29-52     32-14#
DOCOPN    1-60      8-6#
DOCULK    1-65     13-6#
DOOPAP    1-59      7-14#
DORLK     1-63     11-6#
DOSFCK    1-67     15-10#
DOTLK     1-64     12-6#
DOULK1    1-66     14-6#
ECO       2-4#
EC1       2-5#      8-27     11-15    12-12    13-12    14-12
EC2       2-6#      8-35     15-26    19-107
EC3       2-7#      8-62     12-30
EC4       2-8#      8-104    12-36
EMTBLK    1-34      7-14     11-21    12-17    14-23    30-45    30-46    30-47    31-37    31-38    31-39    32-19
         32-20     32-21
EMTXIT    1-43      8-123    11-40    12-24    13-21    14-46    15-31
ENQTL     1-27     24-34
FC$$SS    1-43
FC$ACC    1-29     16-31*   17-36
FC$CHN    1-28     10-31    10-59*   16-24*   27-25
FC$CLK    1-29      4-56*    4-57*    8-40     8-41    16-35*   16-36*   25-22    25-23    25-24    25-25    25-27
         25-29     25-30    25-33*   25-34*   25-39*   25-40*   27-27    27-28
FC$FDB    1-28      8-118*   8-119*   12-32    12-33    19-49    19-50    21-20    21-21    23-23    23-24    25-49
         25-50     29-31    29-32    29-50    29-51    30-40    30-41    31-41    31-42
FC$FLG    1-39      9-20*    10-27    10-45    10-60*   14-18    15-20    15-25*   16-27*   19-25    19-86*   22-16
         23-22*    27-23    29-40*   31-27
FC$FLK    1-28      8-116*   8-117*   10-33    10-34    10-49    10-50    17-41    17-42    20-45    20-46    25-47
         25-48     25-54    25-56    25-58    25-60    25-61    25-63*   25-64*   29-41    29-42
FC$LBN    1-29      1-36     4-51     14-35    16-20    19-41    19-96    20-37    22-26
FC$NLB    1-43     19-38    19-87*   19-93    19-100*  20-32    23-21*
FC$UN     1-29     10-29    10-47    16-23*   20-34
FCFREE    3-45#     4-47*    4-48*    8-33     8-37     8-40*    8-41*   25-39    25-40    25-41*   25-42*
FF$$SZ    1-36      4-35     8-73
FF$CDB    1-32      8-111    8-112    8-113*   8-114*   10-22    10-23    10-42    10-43    17-29    17-30    20-24
         20-26     25-54    25-77    29-33    29-34
FF$DCD    1-40     31-59    31-60    31-61*   31-62*   31-85    31-86    33-19    33-21    34-16    34-17    34-19*
         34-20*    35-23
FF$FID    1-32      8-86*    8-87*    28-44    28-46
FF$FLG    1-39     12-34    19-64    19-83*   23-27*
FF$FLK    1-32      4-40*    4-41*    8-70     8-71     8-79*    8-80*   26-16    26-17    26-19    26-20    26-22
         26-24     26-25    26-28*   26-29*   26-34*   26-35*   28-48    28-49
FF$FWD    1-32     21-46    21-68    23-31    23-32
FF$NLB    1-39     19-56    19-84*   19-102*  23-26*   23-54
FFFREE    3-43#     4-33*    4-34*    8-56     8-68     8-70*    8-71*   26-34    26-35    26-36*   26-37*
FFHEAD    3-44#     8-79     8-80     8-81*    8-82*   26-18    26-18    28-40    28-42
FL$ACT    1-29     15-20    15-25    29-40
FL$EFL    1-39     14-18    19-25    19-86    22-16    23-22
FL$NDC    1-30     16-27    31-27
FL$SPN    1-29      9-20    10-27    10-45    10-60    27-23
FNDCDB    8-11      9-13    11-10    12-10    13-10    14-10    15-14    18-12    27-11#   29-23    30-33    31-21
FNDDCD   30-54     31-48    32-28    33-14#
FNDFDB    8-50     10-14    28-16#
FREFDB   25-83     26-8#
FT$EFL    1-39     12-34    19-64    19-83    23-27
FW$$SZ    1-36      4-20
FW$DBN    1-33     21-38*   23-41    23-48    23-56    23-61
```

Cross reference table (CREF V05.05)

```
FW$UN     1-33     21-37*   24-15
FW$WLK    1-33      4-26*    4-27*   21-32   21-33   21-39*  21-40*  21-46   21-48   21-50   21-51   21-53*
         21-54*   21-66    21-67    21-68   21-70   21-72   21-74   21-75   21-77*  21-78*  21-83*  21-84*
         23-44    23-45    23-63    23-64
FWFREE    3-46#    4-21*    4-22*   21-29   21-30   21-32*  21-33*  21-83   21-84   21-85*  21-86*
INICDB    8-46    16-11#
IOWAIT    1-30    31-33
JCDB      1-35    16-35    16-36    16-37*  16-38*  25-24   27-15   27-17
KPAR5     1-43     4-11     4-81*    4-90*   5-28*   6-22*   8-39*   8-69*   8-110*  8-115*  9-9     9-19*
          9-24*   10-10    10-21*   10-26*  10-41*  10-44*  10-64*  12-31*  12-33*  14-17*  15-19*  16-13
         16-17*   16-42*   17-17    17-28*  17-35*  17-56*  18-8    18-22*  19-19   19-24*  19-55*  19-82*
         19-85*   19-92*   19-101*  19-124* 20-19   20-23*  20-31*  20-60*  21-15   21-19*  21-31*  21-47*
         21-65*   21-69*   21-82*   21-90*  22-11   22-15*  22-36*  23-15   23-20*  23-25*  23-40*  23-47*
         23-55*   23-74*   24-10    24-14*  24-38*  25-13   25-21*  25-31*  25-38*  25-55*  25-76*  25-87*
         26-11    26-15*   26-26*   26-33*  26-41*  27-11   27-22*  27-42*  28-17   28-43*  28-63*  29-19
         29-30*   29-32*   29-35*   29-49*  29-56*  30-29   30-39*  30-79*  31-17   31-26*  31-40*  31-58*
         31-63*   31-84*   31-88*   31-98*  33-14   33-18*  33-26*  33-44*  34-11   34-15*  34-21*  34-30*
         35-12    35-16*   35-24*   35-37*  36-13   36-21*  36-50*  37-12   37-25*  37-33*  37-38*  38-12
         38-22*   38-23*   38-38*   39-12   39-22*  39-23*  39-37*
LDBASE    1-37    28-31
LDDEVX    1-41    28-27
LDPDEV    1-41    28-30
LKWAIT   11-35    21-10#
LOKCSH    1-35     4-80
LOKINI    1-58     4-6#
LOKMEM    1-37     4-16
LPRI      1-27    24-31    24-31
LSTATE    1-27    24-28    24-29
NEWDCD   31-53    36-10#
NLINES    1-33     4-23
NUMCDB    1-31     8-38*   25-43#
NUMDCD    1-35     8-96    25-70   29-47   36-17
OPNCOM    7-15     8-11#
QHIPRI    1-38    24-19
QNSPND    1-38    21-59
RELDCD   32-36    35-9#    36-45
RESTRT   23-43    23-70    24-8#
S$SFWT    1-38    21-58
SETERR    1-43     8-28     8-36    8-63    8-105   11-16   11-30   12-13   12-37   13-13   14-13   15-27
SFCLS     1-68    18-6#
SFRSST    1-62    10-6#
SFSVST    1-61     9-7#
SFWRIT    1-69    29-15#
TRYLK    11-26    12-19    19-17#
TSLOCK    1-23     1-54#    1-58    1-59    1-60    1-61    1-62    1-63    1-64    1-65    1-66    1-67
          1-68     1-69     1-70    1-71    1-72
UNLOCK   14-41    22-19    22-30   23-10#
VALADW    1-31    38-18    39-18
VMLBLK    1-34     4-49    14-36   16-19   19-42   19-93   20-38   22-27
VMXSF     1-33     4-36
VMXSFC    1-33     4-52
VNUMDC    1-34     4-63
VPAR5     1-37     4-15     6-28    6-33    6-38   38-27   39-27
```

Cross reference table (CREF V05.05)

.ADDR	1-19#	16-29	17-21	26-18								
MAPTO	1-47#	4-81	5-28	6-22	8-39	8-69	8-110	8-115	9-19	10-21	10-26	10-41
	10-44	12-31	12-33	14-17	15-19	16-17	17-28	17-35	19-24	19-55	19-82	19-85
	19-92	19-101	20-23	20-31	21-19	21-31	21-47	21-65	21-69	21-82	22-15	23-20
	23-25	23-40	23-47	23-55	24-14	25-21	25-31	25-38	25-55	25-76	26-15	26-26
	26-33	27-22	28-43	29-30	29-32	29-35	29-49	30-39	31-26	31-40	31-58	31-63
	31-84	31-88	33-18	33-26	34-15	34-21	35-16	35-24	36-21	37-25	37-33	38-22
	38-23	39-22	39-23									