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

```
_$25
Symb
----
IOC1
IO_C
IO_C
IO_D
IO_F
IO_S
KILL

KILL
KILL
LB_E
LB_D
LB_F
LB_H
LB_L
LOCA
LOCA
LOCK

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

MAKE
MAKE
MAP_
MAP_

MAP_
MARK
MARK
MARK
MARK
MARK
```

FILEID**FCPDEF

```
FFFFFFFFFF    CCCCCCCC  PPPPPPPP    DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF
FFFFFFFFFF    CCCCCCCC  PPPPPPPP    DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF
FF          CC          PP      PP  DD      DD  EE          FF
FF          CC          PP      PP  DD      DD  EE          FF
FF          CC          PP      PP  DD      DD  EE          FF
FF          CC          PP      PP  DD      DD  EE          FF
FFFFFFFF    CC          PPPPPPPP    DD      DD  EEEEEEEE    FFFFFFFF
FFFFFFFF    CC          PPPPPPPP    DD      DD  EEEEEEEE    FFFFFFFF
FF          CC          PP          DD      DD  EE          FF
FF          CC          PP          DD      DD  EE          FF
FF          CC          PP          DD      DD  EE          FF
FF          CC          PP          DD      DD  EE          FF
FF            CCCCCCCC  PP          DDDDDDDD    EEEEEEEEEE  FF
FF            CCCCCCCC  PP          DDDDDDDD    EEEEEEEEEE  FF

BBBBBBBB      333333      222222
BBBBBBBB      333333      222222
BB      BB  33      33  22      22
BB      BB  33      33  22      22
BB      BB          33          22
BB      BB          33          22
BBBBBBBB          33          22
BBBBBBBB          33          22
BB      BB          33      22
BB      BB          33      22
BB      BB  33      33    22
BB      BB  33      33    22
BBBBBBBB      333333    2222222222
BBBBBBBB      333333    2222222222
```

FCP

M
b

MAC

! S

MAC

LIT

! V

FIE

FIE

FIE

```
DEFINITION FILE FOR FCP COMPILATION

Version:        'V04-000'
```



```
!++
! FACILITY:  F11ACP Structure Level 2
!
! ABSTRACT:
!
!       These are the data stucture definitions and random macros
!       used to compile FCP.
!
! ENVIRONMENT:
!
!       STARLET operating system, including privileged system calls
!       and internal system subroutines.
!
!--
!
! AUTHOR:  Andrew C. Goldstein, CREATION DATE:  9-Dec-1976  10:53
!
! MODIFIED BY:
!
!       V03-033 CDS0020         Christian D. Saether    30-Aug-1984
!               Add flag to disable updating of FILE_HEADER by READ_HEADER.
!               Add cleanup flag to note refcnt up on primary fcb from
!               fid_to_spec routine.
!
!       V03-032 LMP0303         L. Mark Pilant,         21-Aug-1984  13:21
```

```
                Up the storage for the full file spec to accomodate a 16
                level directory tree.

        V03-031 CDS0019         Christian D. Saether    13-Aug-1984
                Add CLF_MARKFCBSTALE flag.
                Remove AV_MARKDEL flag.

        V03-030 CDS0018         Christian D. Saether     6-Aug-1984
                Add STS_HAD_LOCK and STS_KEEP_LOCK flags.

        V03-029 CDS0017         Christian D. Saether     4-Aug-1984
                Add SWITCHES NOSAFE because bliss generates cse's
                for values that cross routine calls that are modified.
                Add DIRINDX_TYPE buffer type.
                Add CACHE_HDR cell.  Remove obsolete cell.
                Modify base constant so that CLEANUP_FLAGS are at 0.
                Remove L_JSB_C linkage (same as L_JSB now).
                Add L_MAP_POINTER linkage.  Declare registers notused
                in the jsb linkages.

        V03-028 CDS0016         Christian D. Saether    15-July-1984
                Reflect the addition of another buffer pool.
                Add another level of BIND to BIND_COMMON.  This
                lets bliss realize that the contents of the base
                register is a constant.

        V03-027 CDS0015         Christian D. Saether    2-July-1984
                Add STS_DISKREAD flag and STSFLGS bitvector
                to indicate last buffer read from disk.

        V03-026 CDS0014         Christian D. Saether    9-May-1984
                Remove definition for VC_NOALLOC.

        V03-025 ACG0427         Andrew C. Goldstein,    8-May-1984  11:08
                Restructure saved audit info to save space

        V03-024 ACG0424         Andrew C. Goldstein,    1-May-1984  20:13
                Add flags to identify implicit SYSPRV to volume owner

        V03-023 CDS0013         Christian D. Saether    20-Apr-1984
                Rework various fields, linkages, and impure storage
                for file access arbitration changes.
                Eliminate intermediate BIND declaration in BIND_COMMON.
                Try word-relative references once again.

        V03-022 ACG0415         Andrew C. Goldstein,    12-Apr-1984  13:31
                Remove ACL handling cells

        V03-021 RSH0135         R. Scott Hanna          06-Mar-1984
                Add AUDIT_COUNT and AUDIT_ARGLIST to global storage.

        V03-020 ACG0408         Andrew C. Goldstein,    20-Mar-1984  16:06
                Reduce size of LOCAL_ARB; make APPLY_RVN and DEFAULT_RVN
                macros; add SURFACE_ERROR macro; redesign global storage macro

        V03-019 ACG0402         Andrew C. Goldstein,    14-Mar-1984  15:02
```

```
        Go back to default longword addressing - it's too big

V03-018 CDS0012         Christian D. Saether    13-Feb-1984
        Add ACB_ADDR to COMMON BIND statement.
        Add BFR_LIST, BFR_CREDITS, and BFRS_USED to COMMON BIND.
        Add VC_SEQNUM field.
        Add L_JSB_C and L_RELEASE_CACHE linkages.
        Replace NO_LCKCHK with CACHELOCK.

V03-017 LMP0186         L. Mark Pilant,         3-Feb-1984  11:53
        Add a new block type CHIP_TYPE for CHIP blocks.

V03-016 CDS0011         Christian D. Saether    19-Dec-1983
        Define BCR11 linkage to base common off register 11.
        Create a BIND definition of popular COMMON cells
        to minimize the number of external references in
        the modules that reference them.
        Remove ADDRESSING_MODE switch forcing longword
        references on all EXTERNAL declarations.

V03-015 CDS0010         Christian D. Saether    14-Oct-1983
        Add JSB_LINK linkage definition.

V03-014 CDS0009         Christian D. Saether    28-Sep-1983
        Add VC_FLAGS fields to include status flags.
        Increase number of lock blocks to 5.

V03-013 CDS0008         Christian D. Saether    21-Sep-1983
        Add definition for number of serial lock blocks.

V03-012 CDS0007         Christian D. Saether    14-Sep-1983
        Add file lock value block context fields.

V03-011 CDS0006         Christian D. Saether    12-Sep-1983
        Add volume lock value block fields.

V03-010 ACG0334         Andrew C. Goldstein,    6-May-1983  14:33
        Fix consistency in declaration of USER_STATUS

V03-009 CDS0005         Christian D. Saether    21-Apr-1983
        Add access lock value block flag DELAY_TRUNC and
        value TRUNC_VBN.
        Add linkage for TRUNC_CHECKS.

V03-008 CDS0004         Christian D. Saether     6-Apr-1983
        Define linkage for LOCK_MODE routine.
        Define access lock value block flag MARKDEL.
        Define ERRCHK macro.

V03-007 STJ3068         Steven T. Jeffreys,     23-Mar-1983
        Defined literal values for erase on delete support.

V03-006 LMP0059         L. Mark Pilant,         27-Dec-1982  9:03
        Always create a FCB for a file header.  This eliminates a
        lot of special case FCB handling.
```

```
        V03-005 CDS0003         Christian D. Saether    15-Dec-1982
                Create PIC_DESC macro for runtime init of
                string descriptor (so it's pic).

        V03-004 CDS0002         C Saether               15-Oct-1982
                Define all event flags to use 30.

        V03-003 CDS0001         C Saetner                6-Oct-1982
                Redefine kernel_call macro to normal call.

        V03-002 LMP0036         L. Mark Pilant,         30-Jun-1982  14:50
                Add an additional block type ACL_TYPE for ACL data block.

        V03-001 LMP0037         L. Mark Pilant,         28-Jun-1982  14:56
                Change all external symbol referencing to be longword relative.

        V02-013 ACG0230         Andrew C. Goldstein,    29-Dec-1981  14:42
                Add expiration date maintenance

        V02-012 ACG0245         Andrew C. Goldstein,    23-Dec-1981  20:09
                Clean up handling of implicitly spooled files

        V02-011 LMP0003         L. Mark Pilant,          8-Dce-1981  11:30
                Add cleanup flag CLF_REMAP to force a rebuild of the files
                windows.  (This is necessary if an extend fails due to the
                user's byte limit quota being exceeded.)

        V02-010 ACG0208         Andrew C. Goldstein,    30-Oct-1981  19:12
                Add segmented directory record support

        V02-009 ACG0167         Andrew C. Goldstein,    16-Apr-1980  19:26
                Previous revision history moved to F11B.REV
**
```

```
SWITCHES NOSAFE;

!
! Declare PSECT usage to minimize page breakage.
!
PSECT
        OWN     = $LOCKEDD1$,
        GLOBAL  = $LOCKEDD1$,
        PLIT    = $CODE$ (EXECUTE);

!
! Declare VAX built in functions.
!
BUILTIN
        TESTBITSS,
        TESTBITSC,
        TESTBITCS,
        TESTBITCC,
        FFS,
        FFC,
        EMUL,
        EDIV,
        ROT,
        REMQUE,
        INSQUE,
        CHMU,
        MTPR,
        HALT;
!
! Structure declarations used for system defined structures to
! save typing.
!

STRUCTURE
        BBLOCK [O, P, S, E; N] =
            [N]
            (BBLOCK+O)<P,S,E>,

        BBLOCKVECTOR [I, O, P, S, E; N, BS] =
            [N*BS]
            ((BBLOCKVECTOR+I*BS)+O)<P,S,E>;
!
! Assorted macros used in FCP code
!

MACRO
                                        ! set processor IPL
        SET_IPL (LEVEL) = MTPR (%REF (LEVEL), PR$_IPL)%;
!
! Declare code that must be locked into the working set.
!

MACRO
        LOCK_CODE       =
                PSECT   CODE    = $LOCKEDC1$,
                        PLIT    = $LOCKEDC1$,
```

```
                        OWN     = $LOCKEDD1$,
                        GLOBAL  = $LOCKEDD1$;
                        %;

!
! ***** Note: The following two macros violate the Bliss language definition
! ***** in that they make use of the value of SP while building the arg list.
! ***** It is the opinion of the Bliss maintainers that this usage is safe
! ***** from planned future optimizations.
!
! Macro to call the change mode to kernel system service.
! Macro call format is "KERNL_CALL (ROUTINE, ARG1, ARG2, ... )".
!
MACRO
        KERNL_CALL (R) =
            BEGIN
            EXTERNAL ROUTINE SYS$CMKRNL : ADDRESSING_MODE (ABSOLUTE);
            BUILTIN SP;
            SYS$CMKRNL (R, .SP, %LENGTH-1
                        %IF %LENGTH GTR 1 %THEN ,%REMAINING %FI)
            END%;
!
! Macro to redefine the old kernel_call macro to a normal call.
!
MACRO
        KERNEL_CALL (R) =
            BEGIN
            R (%REMAINING )
            END%;
!
! Macro to call the change mode to exec system service.
! Macro call format is "EXEC_CALL (ROUTINE, ARG1, ARG2, ... )".
!
MACRO
        EXEC_CALL (R) =
            BEGIN
            EXTERNAL ROUTINE SYS$CMEXEC : ADDRESSING_MODE (ABSOLUTE);
            BUILTIN SP;
            SYS$CMEXEC (R, .SP, %LENGTH-1
                        %IF %LENGTH GTR 1 %THEN ,%REMAINING %FI)
            END%;
!
! Macro used to signal fatal errors (internal consistency checks).
!
MACRO
        BUG_CHECK (CODE, TYPE, MESSAGE) =
                BEGIN
                BUILTIN BUGW;
                EXTERNAL LITERAL %NAME('BUG$_',CODE);
                BUGW (%NAME('BUG$_',CODE) OR 4)
                END
                %;
!
! Macro to signal an error status and continue.
!
MACRO
        ERR_STATUS [CODE] =
```

```
            BEGIN
            %IF NOT %DECLARED (USER_STATUS)
            %THEN EXTERNAL USER_STATUS : WORD
            %ELSE MAP USER_STATUS : WORD
            %FI;
            IF .USER_STATUS
            THEN USER_STATUS = CODE;
            END%;

!
! Macro to signal an error status and exit.
! Implemented as a change mode to user instruction followed by a RET.
!
MACRO
        ERR_EXIT (CODE) =
            (CHMU (%REF (%IF %NULL (CODE) %THEN 0 %ELSE CODE %FI));
            RETURN (BUILTIN R0; .R0))
            %;
!
! Macro to exit with error if value of block is failure.
!
MACRO
        ERRCHK (CALL) =
            (LOCAL STS;
            STS = (CALL);
            IF NOT .STS
            THEN ERR_EXIT (.STS)
            ELSE .STS) %;
!
! Macro to generate a string with a descriptor.
!
MACRO
        DESCRIPTOR (STRING) =
                UPLIT (%CHARCOUNT (STRING), UPLIT BYTE (STRING))%;
!
! Macro to dynamically init a given descriptor for a given string.
! This avoids the non-pic code generated by the DESCRIPTOR macro above.
!
MACRO
        PIC_DESC (STRING, DESC) =
                DESC [0] = %CHARCOUNT (STRING);
                DESC [1] = UPLIT (STRING); %;
!
! Macro to generate a bitmask from a list of bit numbers.
!
MACRO
        BITLIST (ITEM) [] =
                %IF %COUNT NEQ 0 %THEN OR %FI 1^ITEM BITLIST (%REMAINING)
                %;
!
! Macro to return the number of actual parameters supplied to a routine
! call.
!
MACRO
        ACTUALCOUNT =
            BEGIN
            BUILTIN AP;
```

```
            .(.AP)<0,8>
            END
            %;

!
! Macro to check assumed values.
!
MACRO
        ASSUME (Q) =
                %IF NOT (Q)
                %THEN %WARN ('Assumption ', Q, ' is not true')
                %FI
                %;

!
! Macros to do quadword arithmetic. The bizarre coding of compare
! is used because evidently CASE is the only construct that the compiler
! flows correctly in conditionals.
!
MACRO
        SUBQ (SOURCE, DEST, DEST2) =
                BEGIN
                BUILTIN SUBM;
                SUBM (2,
                      SOURCE,
                      DEST,
                      %IF %NULL (DEST2) %THEN DEST %ELSE DEST2 %FI
                      )
                END
                %;
MACRO
        ADDQ (SOURCE, DEST, DEST2) =
                BEGIN
                BUILTIN ADDM;
                ADDM (2,
                      SOURCE,
                      DEST,
                      %IF %NULL (DEST2) %THEN DEST %ELSE DEST2 %FI
                      )
                END
                %;
MACRO
        CMPQ (SOURCE, REL, DEST) =
                BEGIN
                BUILTIN CMPM;
                CASE CMPM (2, SOURCE, DEST)
                FROM -1 TO 1 OF
                    SET
                    [-1]:   %STRING (REL) EQL 'LSS'
                         OR %STRING (REL) EQL 'LEQ'
                         OR %STRING (REL) EQL 'NEQ';
                    [0]:    %STRING (REL) EQL 'GEQ'
                         OR %STRING (REL) EQL 'LEQ'
                         OR %STRING (REL) EQL 'EQL';
                    [1]:    %STRING (REL) EQL 'GTR'
                         OR %STRING (REL) EQL 'GEQ'
                         OR %STRING (REL) EQL 'NEQ';
                    TES
```

```
                END
                %;

!
! Macros to apply the current RVN to a file ID from the file structure,
! and default the RVN to zero when it is the current one.
!
MACRO
        APPLY_RVN (RVN, CURRENT_RVN) =
            BEGIN
            IF .(RVN)<0,8> EQL 0
            THEN (RVN)<0,8> = CURRENT_RVN;
            IF .(RVN)<0,8> EQL 1
            AND CURRENT_RVN EQL 0
            THEN (RVN)<0,8> = 0;
            END
            %,

        DEFAULT_RVN (RVN, CURRENT_RVN) =
            BEGIN
            IF .(RVN)<0,8> EQL CURRENT_RVN
            THEN (RVN)<0,8> = 0;
            END
            %;


!
! Macro to evaluate a disk error status code as being a surface error
! (i.e., caused by the disk medium  as opposed to the controller).
!
MACRO
        SURFACE_ERROR (CODE) =
                    CODE EQL SS$_PARITY
                OR CODE EQL SS$_DATACHECK
                OR CODE EQL SS$_FORMAT
                OR CODE EQL SS$_FORCEDERROR
                %;
!
! File ID's that are known constants
!
LITERAL
        INDEX_FID       = 1,            ! index file
        BITMAP_FID      = 2,            ! storage map file
        BADBLK_FID      = 3,            ! bad block file
        MFD_FID         = 4,            ! MFD
        CORIMG_FID      = 5,            ! core image file
        VOLSET_FID      = 6,            ! volume set list file
        CONTIN_FID      = 7,            ! continuation file
        BACKUP_FID      = 8,            ! backup journal file
        BADLOG_FID      = 9;            ! bad block log file
!
! Constants used in protection checking
!
LITERAL
        SYSTEM_UIC      = 8,            ! highest UIC group of system UIC's

        READ_ACCESS     = 0,            ! file access modes
        WRITE_ACCESS    = 1,
```

```
        DELETE_ACCESS   = 2,
        CREATE_ACCESS   = 3,
        RDATT_ACCESS    = 4,
        WRATT_ACCESS    = 5,
        EXEC_ACCESS     = 6;

!
! Type codes used to identify blocks being read by READ_BLOCK.
! Note that READ_BLOCK contains a table indexed by these codes.
!
LITERAL
        HEADER_TYPE     = 0,            ! file header
        BITMAP_TYPE     = 1,            ! storage bitmap
        DIRECTORY_TYPE  = 2,            ! directory block
        INDEX_TYPE      = 3,            ! other index file blocks
        DATA_TYPE       = 4,            ! random data file blocks
        QUOTA_TYPE      = 5,            ! disk quota file blocks
        DIRINDX_TYPE    = 6;            ! directory index type blocks

!
! Type codes used to identify blocks of memory requested from the
! allocator. Note that these codes index into a table in ALLOCATE.
!
LITERAL
        FCB_TYPE        = 0,            ! file control block
        WCB_TYPE        = 1,            ! window block
        VCB_TYPE        = 2,            ! volume control block
        RVT_TYPE        = 3,            ! relative volume table
        MVL_TYPE        = 4,            ! magtape volume list
        AQB_TYPE        = 5,            ! ACP queue control block
        CACHE_TYPE      = 6,            ! cache data block
        ACL_TYPE        = 7,            ! Access Control List block
        CHIP_TYPE       = 8;            ! $CHKPRO internal interface block

!
! Mode codes for the bad block log file scan routine
!
LITERAL
        REMOVE_BADBLOCK = 0,            ! remove log entry
        ENTER_READERR   = 1,            ! log read error
        ENTER_WRITERR   = 2;            ! log write error

!
! Mode flags for the routine CHARGE_QUOTA.
!
LITERAL
        QUOTA_CHECK     = 0,            ! check space requested against quota
        QUOTA_CHARGE    = 1;            ! charge the space to the quota file

!
! Index codes for the subfunctions in the performance measurement data base.
!
LITERAL
        PMS_FIND        = 6,            ! directory searches
        PMS_ENTER       = 7,            ! directory entries
        PMS_ALLOC       = 8,            ! storage map allocation and deallocation
        PMS_RWATT       = 9;            ! read/write attributes

!
! Random constants.
!
LITERAL
```

```
        LB_NUM          = 5,            ! number of serial lock blocks.
        EFN             = 30,           ! event flag for I/O
        MBX_EFN         = 30,           ! event flag for asynchronous mailbox I/O
        TIMER_EFN       = 30,           ! EFN for timers
        MAILBOX_EFN     = 4,            ! EFN for job controller reply mailbox
        FILENAME_LENGTH = 80,           ! maximum file name length
        MIN_WINDOW      = 1,            ! minimum window size
        MAX_WINDOW      = 80,           ! maximum window size (in pointers)
        MAX_ACL_SIZE    = 512;          ! Maximum size of an (in core) ACL

!
! Modes to call TRUNCATE routine.
!
LITERAL
        ERASE_POINTERS  = 1,            ! erase retrieval pointers removed
        DEALLOC_BLOCKS  = 1;            ! deallocate the blocks

!
! Normal termination cleanup flags
!
LITERAL
        CLF_FIXFCB      = 1,            ! update FCB from header
        CLF_DOSPOOL     = 2,            ! send file to print queue
        CLF_INVWINDOW   = 4,            ! invalidate all windows
        CLF_SUPERSEDE   = 5,            ! supersede old file
        CLF_DIRECTORY   = 6,            ! directory operation enabled
        CLF_SPOOLFILE   = 7,            ! operation is on spool file
        CLF_SYSPRV      = 8,            ! user has SYSTEM privilege or equivalent
        CLF_CLEANUP     = 9,            ! cleanup is in progress
        CLF_INCOMPLETE  = 10,           ! file is not completely mapped
        CLF_NOBUILD     = 11,           ! don't get ACL info from header
        CLF_VOLOWNER    = 12,           ! SYSPRV implied by volume ownership
        CLF_GRPOWNER    = 13,           ! SYSPRV implied by GRPPRV and above
        CLF_MARKFCBSTALE= 14,           ! Mark primary_fcb stale clusterwide.
        CLF_PFCB_REF_UP = 15,           ! Primary_fcb refcnt is up.
!
! Error termination cleanup flags
!
        CLF_DEACCESS    = 16,           ! deaccess file
        CLF_ZCHANNEL    = 17,           ! clean out user's channel
        CLF_TRUNCATE    = 18,           ! undo extend operation
        CLF_FLUSHFID    = 19,           ! flush file ID cache
        CLF_DELFID      = 20,           ! delete file ID
        CLF_DELFILE     = 21,           ! delete complete file
        CLF_REMOVE      = 22,           ! remove directory entry
        CLF_REENTER     = 23,           ! put directory entry back
        CLF_CLOSEFILE   = 24,           ! close internal file
        CLF_DEACCQFILE  = 25,           ! deaccess quota file
        CLF_DELWINDOW   = 26,           ! deallocate window
        CLF_HDRNOTCHG   = 27,           ! file header not charged to user
        CLF_DELEXTFID   = 28,           ! delete extension header
        CLF_NOTCHARGED  = 29,           ! disk blocks not charged to user yet
        CLF_FIXLINK     = 30,           ! restore old file back link
        CLF_REMAP       = 31;           ! remap the file to fix up the windows
!
! Cleanup actions that modify the disk, and are to be turned off in case
! of a write error.
```

```
!
LITERAL
        CLF_M_WRITEDISK =
                1^CLF_SUPERSEDE         ! supersede old file
                OR 1^CLF_TRUNCATE       ! undo extend operation
                OR 1^CLF_DELFID         ! delete file ID
                OR 1^CLF_DELFILE        ! delete complete file
                OR 1^CLF_REMOVE         ! remove directory entry
                OR 1^CLF_REENTER        ! put directory entry back
                OR 1^CLF_DELEXTFID;     ! delete extension header

!
! Various internal status flags for the STSFLGS bitvector.
!

LITERAL
        STS_DISKREAD    = 0,            ! last buffer read was from disk, not cache
        STS_HAD_LOCK    = 1,            ! already held lock.
        STS_KEEP_LOCK   = 2,            ! keep open_file lock
        STS_LEAVE_FILEHDR = 3;          ! Don't update FILE_HEADER cell.

!
! Structure definitions for the file name descriptor block.
!

MACRO
        FND_FLAGS       = 0, 0, 16, 0%, ! file name flag bits
        FND_WILD_NAME   = 0, $BITPOSITION (FIB$V_ALLNAM), 1, 0%,        ! wild card name
        FND_WILD_TYPE   = 0, $BITPOSITION (FIB$V_ALLTYP), 1, 0%,        ! wild card type
        FND_WILD_VER    = 0, $BITPOSITION (FIB$V_ALLVER), 1, 0%,        ! wild card version
        FND_WILD        = 0, $BITPOSITION (FIB$V_WILD),   1, 0%,        ! wild card in name
        FND_MAX_VER     = 0, $BITPOSITION (FIB$V_NEWVER), 1, 0%,        ! maximize version
        FND_FIND_FID    = 0, $BITPOSITION (FIB$V_FINDFID), 1, 0%,       ! search for file ID
        FND_COUNT       = 4, 0, 32, 0%, ! name string length
        FND_STRING      = 8, 0, 32, 0%, ! name string address
        FND_VERSION     = 12,0, 16, 1%; ! version number

LITERAL
        FND_LENGTH      = 16;           ! length of filename descriptor

! Structure of directory scan context block.
!

MACRO
        DCX_VBN         = 0, 0, 32, 0%, ! directory VBN
        DCX_BUFFER      = 4, 0, 32, 0%, ! address of current buffer
        DCX_ENTRY       = 8, 0, 32, 0%, ! address of current record
        DCX_VERSION     = 12,0, 32, 0%, ! address of current version
        DCX_END         = 16,0, 32, 0%, ! address of end of data
        DCX_PRED        = 20,0, 32, 0%, ! address of predecessor record
        DCX_VERLIMIT    = 24,0, 16, 0%, ! version limit of current name
        DCX_VERCOUNT    = 26,0, 16, 0%, ! number of versions traversed
        DCX_NAME        = 28,0, 00, 0%; ! name string of prev. entry

LITERAL
        DCX_LENGTH      = 28+FILENAME_LENGTH+1+3 AND NOT 3;
```

```
                                        ! length of context block

! Macro to define direct access names for the standard directory context
! block.
!

MACRO
        DIR_CONTEXT_DEF =
            BIND
                DIR_VBN         = DIR_CONTEXT[DCX_VBN],
                DIR_BUFFER      = DIR_CONTEXT[DCX_BUFFER]       : REF BBLOCK,
                DIR_ENTRY       = DIR_CONTEXT[DCX_ENTRY]        : REF BBLOCK,
                DIR_VERSION     = DIR_CONTEXT[DCX_VERSION]      : REF BBLOCK,
                DIR_END         = DIR_CONTEXT[DCX_END]          : REF BBLOCK,
                DIR_PRED        = DIR_CONTEXT[DCX_PRED]         : REF BBLOCK,
                VERSION_LIMIT   = DIR_CONTEXT[DCX_VERLIMIT]     : WORD,
                VERSION_COUNT   = DIR_CONTEXT[DCX_VERCOUNT]     : WORD,
                LAST_ENTRY      = DIR_CONTEXT[DCX_NAME]         : VECTOR [,BYTE]
            %;
!
! Structure of the saved audit block (in AUDIT_ARGLIST).
!

MACRO
        AUDIT_TYPE      = 0, 0, 8, 0 %, ! audit record flags
        AUDIT_SUCCESS   = 1, 0, 1, 0 %, ! successful file access
        AUDIT_FID       = 2, 0, 0, 0 %, ! file ID of file
        AUDIT_ACCESS    = 8, 0,32, 0 %, ! access mask
        AUDIT_PRIVS     =12, 0,32, 0 %; ! privileges used

LITERAL
        AUDIT_LENGTH    = 16,           ! length of audit block
        MAX_AUDIT_COUNT = 4;            ! max number of auditable entries

! Various field definitions.
!

FIELD
    AV =
        SET
        AV_DELAYTRNC            = [0,1,1,0],    ! Delay truncation operation
        AV_TRUNCVBN             = [4,0,32,0]    ! VBN to truncate.
        TES;

FIELD
    FC =
        SET
        FC_HDRSEQ               = [0,0,32,0],
        FC_DATASEQ              = [4,0,32,0],
        FC_FILESIZE             = [8,0,32,0]
        TES;

FIELD
    VC =
        SET
        VC_FLAGS                = [0,0,16,0],
```

```
        VC_NOTFIRST_MNT         = [0,0,1,0],
        VC_IBMAPVBN             = [2,0,8,0],
        VC_SBMAPVBN             = [3,0,8,0],
        VC_VOLFREE              = [4,0,32,0],
        VC_IDXFILEOF            = [8,0,32,0],
        VC_SEQNUM               = [12,0,32,0]
        TES;

FIELD
    DIRC =
        SET
        DIRC$W_INUSE            = [0,0,16,0],
        DIRC$W_TOTALCELLS       = [2,0,16,0],
        DIRC$W_CELLSIZE         = [4,0,16,0],
        DIRC$W_BLKSPERCELL      = [6,0,16,0],
        DIRC$L_DATASEQ          = [8,0,32,0],
        DIRC$T_FIRSTCELL        = [12,0,0,0]
        TES;

! Define linkages here.
!

LINKAGE
        L_NORM          = CALL  : GLOBAL (BASE = 10),

        L_MAP_POINTER   = JSB :
                          GLOBAL (COUNT = 6, LBN = 7, MAP_POINTER = 8)
                                NOTUSED (2,3,4,5,9,10,11),

        L_JSB           = JSB   : GLOBAL (BASE = 10)
                                NOTUSED (4,5,6,7,8,9,11),

        L_JSB_1ARG      = JSB (REGISTER = 0)
                                : GLOBAL (BASE = 10)
                                NOTUSED (4,5,6,7,8,9,11),

        L_JSB_2ARGS     = JSB (REGISTER=0, REGISTER=1)
                                : GLOBAL (BASE = 10)
                                NOTUSED (4,5,6,7,8,9,11),

        L_R1OUT         = CALL (;REGISTER=1)
                                : GLOBAL (BASE = 10) ;

! Boolean literals for erase on delete support.  They are used to make
! the code more readable.
!

LITERAL
        ERASE_THE_DATA  = 1,                    ! Erase the extent
        DO_NOT_ERASE    = 0;                    ! Do not erase the extent
```

```
!
! We haven't figured out yet how to get the length of CONTEXT_SAVE to
! track automatically yet in the local compile. The value below is checked
! with an assume in COMMON.B32.
!
LITERAL
        CONTEXT_SIZE    = 54;

!
! File system global storage. The following macro defines the cells
! in the global storage region.
!
MACRO GLOBAL_STORAGE =

        STORAGE_START,  VECTOR [0],     ! start of global storage
!
! The cells bracketed by L_DATA_START and L_DATA_END delimit the data
! in pages that are locked in the working set.
!
! Also note that any changes in the number and/or size of cells between
! here and the CONTEXT_START (aka CLEANUP_FLAGS) cell should adjust
! the internal ptr defined by the INIT_BASE macro below such that
! the value of CONTEXT_START computes to zero (compile COMMON.B32 and
! look in the listing to see whether it is correct, and if not, what
! the correct adjustment is).
!

        L_DATA_START,   VECTOR [0],     ! beginning of locked down data
        XQP_STACK,      VECTOR [5*512, BYTE], ! 5 page xqp kernel stack
        XQP_QUEUE,      VECTOR [2],     ! XQP queue head.
        XQP_DISPATCHER, LONG,           ! address of XQP dispatch routine
        CODE_SIZE,      LONG,           ! length of code
        CODE_ADDRESS,   LONG,           ! base address of code
        DATA_SIZE,      LONG,           ! length of data area
        DATA_ADDRESS,   LONG,           ! base address of data area
        PREV_FP,        LONG,           ! saved frame pointer
        PREV_STKLIM,    VECTOR [2],     ! saved kernel stack limits
        XQP_STKLIM,     VECTOR [2],     ! XQP kernel stack limits
        XQP_SAVFP,      LONG,           ! saved XQP frame pointer
        IO_CCB,         REF BBLOCK,     ! CCB of IO_CHANNEL.
        IO_CHANNEL,     LONG,           ! channel number for I/O
        BLOCK_LOCKID,   LONG,           ! activity block lock held.

! The remaining locations are initialized to known values (mainly zero)
! by the per request initialization routine.
!

        IMPURE_START,   VECTOR [0],
        USER_STATUS,    VECTOR [2],     ! I/O status to be returned to user
        IO_STATUS,      VECTOR [2],     ! status block for FCP I/O
        IO_PACKET,      REF BBLOCK,     ! address of current I/O request packet
        CURRENT_UCB,    REF BBLOCK,     ! address of UCB of current request
        CURRENT_VCB,    REF BBLOCK,     ! address of VCB of current request
        CURRENT_RVT,    REF BBLOCK,     ! RVT of current volume set, or UCB
        CURRENT_RVN,    LONG,           ! RVN of current volume
        SAVE_VC_FLAGS,  WORD,           ! save volume context flags.
```

```
        STSFLGS,        BITVECTOR [8],  ! various internal status flags
        BLOCK_CHECK,    BYTE,           ! make operation blocking check
        NEW_FID,        LONG,           ! file number of unrecorded file ID
        NEW_FID_RVN,    LONG,           ! RVN of above
        HEADER_LBN,     LONG,           ! LBN of last file header read
        BITMAP_VBN,     LONG,           ! VBN of current storage map block
        BITMAP_RVN,     LONG,           ! RVN of current storage map block
        BITMAP_BUFFER,  REF BBLOCK,     ! address of current storage map block
        SAVE_STATUS,    LONG,           ! saved status during DELETE's header read
        PRIVS_USED,     BBLOCK [4],     ! Privileges used to gain access
        ACB_ADDR,       REF BBLOCK,     ! address of ACB for cross process asts
        BFR_LIST,       BLOCKVECTOR [4,8,BYTE],
                                        ! listheads for in-process buffers
        BFR_CREDITS,    VECTOR [4,WORD], ! buffers credited to this process
        BFRS_USED,      VECTOR [4,WORD], ! buffers actually in-process
        CACHE_HDR,      REF BBLOCK,     ! Address of buffer cache header

! See the comment above at the L_DATA_START cell regarding the compiletime
! pointer in INIT_BASE if any cells to this point are added, deleted,
! or change size.
!
                                        ! The following locations are the re-enterable
                                        ! context area and must be saved when an
                                        ! secondary operation is performed.
        CONTEXT_START,  VECTOR [0],     ! ***** The next item must be CLEANUP_FLAGS
        CLEANUP_FLAGS,  BITVECTOR [32], ! cleanup action flags
        FILE_HEADER,    REF BBLOCK,     ! address of current file header
        PRIMARY_FCB,    REF BBLOCK,     ! address of primary file FCB
        CURRENT_WINDOW, REF BBLOCK,     ! address of file window
        CURRENT_FIB,    REF BBLOCK,     ! pointer to FIB currently in use
        CURR_LCKINDX,   LONG,           ! Current file header lock index.
        PRIM_LCKINDX,   LONG,           ! Primary file lock basis index.
        LOC_RVN,        LONG,           ! RVN specified by placement data
        LOC_LBN,        LONG,           ! LBN specified by placement data
        UNREC_LBN,      LONG,           ! start LBN of unrecorded blocks
        UNREC_COUNT,    LONG,           ! count of unrecorded blocks
        UNREC_RVN,      LONG,           ! RVN containing unrecorded blocks
        PREV_LINK,      BBLOCK [FID$C_LENGTH], ! old back link of file
        CONTEXT_END,    VECTOR [0],

        CONTEXT_SAVE,   VECTOR [CONTEXT_SIZE, BYTE],
                                        ! area to save primary context
        CONTEXT_SAVE_END, VECTOR [0],   ! end of above
        LB_LOCKID,      VECTOR [LB_NUM], ! serial lock ids.
        LB_BASIS,       VECTOR [LB_NUM], ! lock name bases.
        LB_HDRSEQ,      VECTOR [LB_NUM], ! file header cache sequence numbers.
        LB_DATASEQ,     VECTOR [LB_NUM], ! file data block cache sequence number.
        LB_FILESIZE,    VECTOR [LB_NUM], ! value block file size.

        DIR_FCB,        REF BBLOCK,     ! FCB of directory file
        DIR_LCKINDX,    LONG,           ! Directory lock basis index.
        DIR_RECORD,     LONG,           ! record number of found directory entry
        DIR_CONTEXT,    BBLOCK [DCX_LENGTH], ! current directory context
        OLD_VERSION_FID, BBLOCK [FID$C_LENGTH], ! Old version's FID
        PREV_VERSION,   LONG,           ! version number of previous directory entry
        PREV_NAME,      VECTOR [FILENAME_LENGTH+1, BYTE], ! name of previous entry
```

```
        PADDING_0,      VECTOR [1, BYTE],
        PREV_INAME,     VECTOR [FILENAME_LENGTH+6, BYTE], ! previous internal file name
        SUPER_FID,      BBLOCK [FID$C_LENGTH], ! file ID of superseded file
        LOCAL_FIB,      BBLOCK [FIB$C_LENGTH], ! primary FIB of this operation
        SECOND_FIB,     BBLOCK [FIB$C_LENGTH], ! FIB for secondary file operation
        LOCAL_ARB,      BBLOCK [ARB$C_HEADER], ! local copy of caller's ARB

        L_DATA_END,     VECTOR [0],            ! end of locked down data area.

        QUOTA_RECORD,   LONG,          ! record number of quota file entry
        FREE_QUOTA,     LONG,          ! record number of free quota file entry
        REAL_Q_REC,     REF BBLOCK,    ! buffer address of quota record read
        QUOTA_INDEX,    LONG,          ! cache index of cache entry found
        DUMMY_REC,      BBLOCK [DQF$C_LENGTH], ! dummy quota record for cache contents
        AUDIT_COUNT,    LONG,                  ! number of argument lists in AUDIT_ARGLIST

        IMPURE_END,     VECTOR [0],            ! end of initialized impure area

        MATCHING_ACE,   BBLOCK [ATR$S_READACL], ! Matching ACE storage

! The following two items must be adjacent.

        FILE_SPEC_LEN,  VECTOR [1, WORD],      ! Full file spec length

! Note that the size of the full file specification storage must
! track the definition in the routine FID_TO_SPEC.

        FULL_FILE_SPEC, VECTOR [1022, BYTE],   ! Full spec storage

! The preceeding two items must be adjacent.


! The following cells are used by PMS.

        PMS_TOT_READ,   LONG,          ! total disk reads
        PMS_TOT_WRITE,  LONG,          ! total disk writes
        PMS_TOT_CACHE,  LONG,          ! total cache reads

        PMS_FNC_READ,   LONG,
        PMS_FNC_WRITE,  LONG,
        PMS_FNC_CACHE,  LONG,
        PMS_FNC_CPU,    LONG,
        PMS_FNC_PFA,    LONG,

! Base values of parameters at start of current subfunction.

        PMS_SUB_NEST,   LONG,          ! nested subfunction flag

        PMS_SUB_FUNC,   LONG,          ! subfunction code
        PMS_SUB_READ,   LONG,
        PMS_SUB_WRITE,  LONG,
        PMS_SUB_CACHE,  LONG,
        PMS_SUB_CPU,    LONG,
        PMS_SUB_PFA,    LONG,

        AUDIT_ARGLIST,  BBLOCK [AUDIT_LENGTH*MAX_AUDIT_COUNT], ! security auditing argument lists
```

```
        STORAGE_END,    VECTOR [0],     ! end of global storage
        %;

!
! Define the base offset for the global common area. This is set up so
! that CONTEXT_START (CLEANUP_FLAGS) is at offset zero. When storage is
! added or removed before this cell, the base offset should be adjusted
! accordingly.
!
MACRO
        INIT_BASE =
                COMPILETIME $PTR = -2752
                %;

!
! Macro to declare global storage locally for the current compilation.
! This macro is invoked by most file system routines to link to the
! global common area.
!
MACRO BIND_COMMON =
                INIT_BASE;
                EXTERNAL REGISTER BASE = 10;
                BIND BR = .BASE;
                DEFINE_LOCAL (GLOBAL_STORAGE)
                %;

MACRO DEFINE_LOCAL [A, B] =
                BIND A = BR + $PTR : B
                %ASSIGN ($PTR, $PTR + %SIZE (%IF %IDENTICAL (B, LONG)
                                              OR %IDENTICAL (B, WORD)
                                              OR %IDENTICAL (B, BYTE)
                                             %THEN VECTOR [1, B]
                                             %ELSE B
                                             %FI))
                %;

!
! Macro to declare global storage globally for the entire file system.
!
MACRO GLOBAL_COMMON =
                INIT_BASE;
                DEFINE_GLOBAL (GLOBAL_STORAGE)
                %;

MACRO DEFINE_GLOBAL [A, B] =
                GLOBAL LITERAL A = $PTR
                %ASSIGN ($PTR, $PTR + %SIZE (%IF %IDENTICAL (B, LONG)
                                              OR %IDENTICAL (B, WORD)
                                              OR %IDENTICAL (B, BYTE)
                                             %THEN VECTOR [1, B]
                                             %ELSE B
                                             %FI))
                %;
```

```
!
! Macro to declare common base register external when full bind is not
! necessary.
!

MACRO   BASE_REGISTER =

EXTERNAL REGISTER
        BASE = 10; %;
```

FCPDEF
B32

ACLCNTRL
LIS

ACLSUBR
LIS

SNDERL
LIS

ACPCNTRL
LIS

F11X

F11BXQP
MAP

ACCESS
LIS

TRUNC
LIS

FILESERV
MAP

WITURN
LIS

SNDSMB
LIS