```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

F 7

FILEID**RMSINTSTR

```
RRRRRRRR    MM      MM    SSSSSSSS    IIIIII    NN      NN  TTTTTTTTTT    SSSSSSSS  TTTTTTTTTT  RRRRRRRR
RRRRRRRR    MM      MM    SSSSSSSS    IIIIII    NN      NN  TTTTTTTTTT    SSSSSSSS  TTTTTTTTTT  RRRRRRRR
RR      RR  MMMM  MMMM  SS              II      NN      NN      TT      SS              TT      RR      RR
RR      RR  MMMM  MMMM  SS              II      NN      NN      TT      SS              TT      RR      RR
RR      RR  MM  MM  MM  SS              II      NNNN    NN      TT      SS              TT      RR      RR
RR      RR  MM  MM  MM  SS              II      NNNN    NN      TT      SS              TT      RR      RR
RRRRRRRR    MM      MM    SSSSSS        II      NN  NN  NN      TT        SSSSSS        TT      RRRRRRRR
RRRRRRRR    MM      MM    SSSSSS        II      NN  NN  NN      TT        SSSSSS        TT      RRRRRRRR
RR  RR      MM      MM          SS      II      NN    NNNN      TT              SS      TT      RR  RR
RR  RR      MM      MM          SS      II      NN    NNNN      TT              SS      TT      RR  RR
RR    RR    MM      MM          SS      II      NN      NN      TT              SS      TT      RR    RR        ....
RR    RR    MM      MM          SS      II      NN      NN      TT              SS      TT      RR    RR        ....
RR      RR  MM      MM  SSSSSSSS      IIIIII    NN      NN      TT      SSSSSSSS        TT      RR      RR      ....
RR      RR  MM      MM  SSSSSSSS      IIIIII    NN      NN      TT      SSSSSSSS        TT      RR      RR      ....

  SSSSSSSS  DDDDDDDD    LL
  SSSSSSSS  DDDDDDDD    LL
SS          DD      DD  LL
SS          DD      DD  LL
SS          DD      DD  LL
SS          DD      DD  LL
  SSSSSS    DD      DD  LL
  SSSSSS    DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
SSSSSSSS    DDDDDDDD    LLLLLLLLLL
SSSSSSSS    DDDDDDDD    LLLLLLLLLL
```

```
{       $begin  rmsintstr,V04-000
{
{****************************************************************************
{*                                                                          *
{*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
{*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
{*  ALL RIGHTS RESERVED.                                                    *
{*                                                                          *
{*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
{*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
{*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
{*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
{*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
{*  TRANSFERRED.                                                            *
{*                                                                          *
{*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
{*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
{*  CORPORATION.                                                            *
{*                                                                          *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
{*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
{*                                                                          *
{*                                                                          *
{****************************************************************************
{
```

```
{
{       internal rms structure definitions
{
{ Modified By:
{
{       V03-070 JEJ0053         J E Johnson             30-Aug-1984
{               Add FTL code for an invalid EBK/HBK value.
{
{       V03-069 JEJ0022         J E Johnson             04-Apr-1984
{               Fix boken quadword alignment in previous change.
{
{       V03-068 JEJ0010         J E Johnson             19-Mar-1984
{               Add GBH$L_OUTBUFQUO to count the number of times that
{               a global section exceeds the GBLBUFQUO sysgen limit.
{
{       V03-067 DGB0005         Donald G. Blair         28-Feb-1984
{               Add IFB$B_AGENT_MODE.
{
{       V03-066 JWT0150         Jim Teague              02-Feb-1984
{               Add IFB$W_BUFFER_OFFSET.
{
{       V03-065 SHZ0005         Stephen H. Zalewski     06-Dec-1983
{               Add new FTL definition.
{
{       V03-064 JWT0141         Jim Teague              11-Nov-1983
{               Change IFB$V_RUM to IFB$V_ONLY_RU
{
{       V03-063 KBT0566         Keith B. Thompson       26-Jul-1983
{               Change the flag V_NWA to V_FILEFOUND
{
{       V03-062 SHZ0004         Stephen H. Zalewski     28-Jun-1983
{               Add several new FTL codes.
{
{       V03-061 KPL0004         Peter Lieberwirth       21-Jun-1983
{               Add new FTL code.  Last few edits had wrong ident.
{
{       V03-059 KPL0003         Peter Lieberwirth       20-Jun-1983
{               Mung JNLFLGs fields.
{
{       V03-058 SHZ0003         Stephen H. Zalewski     20-Jun-1983
{               Add new fields to IFBDEF, GBHDEF and GBDDEF for cluster
{               global buffers.  Also remove obsolete rms failure codes.
{
{       V03-057 KPL0022         Peter Lieberwirth       26-May-1983
{               Add more journaling flags{ a second byte in the IFB and
{               a byte in the IRB.  First use is a flag to indicate that
{               a valid AT journal entry exists for the IFB/IRB operation
{               and should be flushed.  Also, move IFB$C_BLN_SEQ past all
{               the common journaling structures.
{
{       V03-058 KPL0021         Peter Lieberwirth       13-May-1983
{               Increase size of ASB used for FAB operations.
{
{       V03-057 KPL0020         Peter Lieberwirth        1-May-1983
{               Align MJB.
{
```

```
{       V03-056 KPL0019         Peter Lieberwirth       30-Apr-1983
{               Add some MJB flags.
{
{       V03-055 KPL0018         Peter Lieberwirth       29-Apr-1983
{               Add MJB definition.  The Miscellaneous Journal Buffer
{               is used to write out misc. journal entries.  Add
{               individual fields for journal channels in RJB channel
{               quadword.
{
{       V03-054 KPL0017         Peter Lieberwirth       28-Apr-1983
{               Add pointer to audit trail journaling buffer (ATJNLBUF)
{               in IFB and IRB.  Add pointer to journaling buffer used
{               for EXTENDs in IFB (EXTJNLBUF).
{
{       V03-053 JWH0209         Jeffrey W. Horn         12-Apr-1983
{               Remove mapping sequence numbers from the RJB.
{               Also had to replace the source because of compare
{               buffer overflow.
{
{       V03-052 JWH0197         Jeffrey W. Horn         21-Mar-1983
{               Add FLB, the File Lock Block, to save the file
{               lock in the RULOCK list.
{
{       V03-051 RAS0130         Ron Schaefer            14-Mar-1983
{               Change BDB structure for more general space utilization{
{               add BDB$L_ALLOC_ADDR and BDB$W_ALLOC_SIZE fields.
{               Revise BDB journaling fields as well (JWH0184).
{
{       V03-050 DAS0003         David Solomon           18-Feb-1983
{               Add RLB fields for timeout on record lock.
{
{       V03-049 JWH0184         Jeffrey W. Horn         10-Feb-1983
{               Added BDB$L_BI_ADDR, BDB$L_AI_ADDR, BDB$W_BI_SIZE, and
{               BDB$W_AI_SIZE{ re-definitions of other BDB fields which
{               describe the isam journaling BDB.
{
{       V03-048 DAS0001         David Solomon           26-Jan-1983
{               Add IDX$C_SGNQUAD, IDX$C_UNSGNQUAD for 64-bit binary keys.
{
{       V03-047 RAS0120         Ron Schaefer            25-Jan-1983
{               Add support to echo SYS$INPUT to SYS$OUTPUT:
{               add bit IRB$V_PPF_ECHO and field IFB$W_ECHO_ISI.
{
{       V03-046 JWH0170         Jeffrey W. Horn         18-Jan-1983
{               Add the bit RLB$V_FAKE to the flag byte of the RLB.
{               Add the bit IFB$V_RU_RLK to IFB$B_JNLFLG.
{
{       V03-045 TMK0010         Todd M. Katz            14-Jan-1983
{               Add the bit IRB$V_NO_Q_WAIT to the bookkeeping bit field of
{               the IRAB.
{
{       V03-044 TMK0009         Todd M. Katz            12-Jan-1983
{               Add the bit IRB$V_RU_UPDATE to the bookkeeping bit field of
{               the IRAB.
{
{       V03-043 LJA0053         Laurie J. Anderson      12-Jan-1983
{               Add SHR field to IFB and add MBF field to IRB
{
{       V03-042 LJA0049         Laurie J. Anderson      10-Jan-1983
{               Fix LJA0045 to make ISI/IFI a word not byte.
{
{       V03-041 KBT0452         Keith B. Thompson       6-jan-1983
{               Make ifab longword aligned
{
{       V03-040 SHZ0002         Stephen H. Zalewski     5-Dec-1982
{               Moved ifb$l_hbk and ifb$l_ebk out of file header area of ifb
{               and replaced them with ifb$l_hbk_disk and ifb$l_ebk_disk.
{
{               Removed ebk0 and ebk2 subfields from ifb.
{
{       V03-039 KBT0444         Keith B. Thompson       4-Dec-1982
{               Increase size of the name buffer in the directory
{               in the cache node (DRC$)
{
{       V03-038 TMK0008         Todd M. Katz            22-Dec-1982
{               Add the bits IRB$V_RU_DELETE and IRB$V_RU_UNDEL to the
{               bookkeeping bit field of the IRAB. Also add the field
{               IRB$L_OLDBUF to the IRAB.
{
{               Add the field IFB$B_RECVRFLGS to the IFAB, and define several
{               of the bits within this field. Set the constant IFB$C_KBUFNUM
{               to 6 so that only six keybuffers will be allocated.
{
{       V03-037 LJA0045         Laurie J. Anderson      21-Dec-1982
{               Add IFI/ISI field to the IFB/IRB for context extraction $DISPLAY
{
{       V03-036 KBT0422         Keith B. Thompson       30-Nov-1982
{               Change ifb$w_devbufsiz to ifb$l_devbufsiz and ifb$w_asdevbsiz
{               to ifb$l_asdevbsiz
{
{       V03-035 LJA0041         Laurie J. Anderson      30-Nov-1982
{               Add ifb$c_kbufnum - As a constant of the number of key
{               buffers allocated.
{
{       V03-034 KBT0404         Keith B. Thompson       23-Nov-1982
{               Add fwa_ptr to ifab
{
{       V03-033 KBT0401         Keith B. Thompson       9-Nov-1982
{               Make ISAM ASB bigger again.
{
{       V03-032 KBT0399         Keith B. Thompson       4-Oct-1982
{               Remvove FWA definitions and put them in RMSFWADEF.MDL
{               and add 32 more bytes to the asb isam stack.
{
{       V03-031 MCN0009         Maria del C. Nasr       29-Oct-1982
{               KEY_COMPR flag in the index descriptor can be defined
{               for all keys.  Also eliminate COUNT_DUP, NORFA, and
{               PRG_D_RFA flags since they are not referenced.
{
{       V03-030 KBT0384         Keith B. Thompson       26-Oct-1982
{               Make asb$l_argcnt a byte field again
{
```

```
{       V03-029 KPL0016         Peter Lieberwirth       26-Oct-1982
{               Move RMS Journaling and Recovery structures RJR and RMSR
{               to RMSJNLSTR.MDL.
{
{       V03-028 KBT0368         Keith B. Thompson       14-Oct-1982
{               Add new asb fields
{
{       V03-027 JWH0111         Jeffrey W. Horn         29-Sep-1982
{               Backout JWH106, JWH0100.  Make FWA$T_xxJNLN to be
{               .ASCIC strings.  Implement new RJR format.  Add RMSR
{               definitions.  Standardize format of RJB.
{
{       V03-026 KBT0361         Keith B. Thompson       6-Oct-1982
{               Make asb$b_stksz a word field
{
{       V03-025 JWH0106         Jeffrey W. Horn         22-Sep-1982
{               Add IFB$V_MKC in IFB$B_JNLFLG to indicate file is
{               marked as 'closed'.
{
{       V03-024 KBT0342         Keith B. Thompson       22-Sep-1982
{               Make ASB 7 longwords bigger and backout JWH0102
{
{       V03-023 JWH0102         Jeffrey W. Horn         20-Sep-1982
{               Add the field IFB$L_RULOCK which points to a SFSB to hold
{               a lock-manager lock on files opened non-shared and which
{               can be recovery-unit journaled.
{
{       V03-022 TMK0007         Todd M. Katz            18-Sep-1982
{               Add the bit IRB$V_LAST_GT to the IRAB field IRB$W_SRCHFLAGS.
{
{       V03-021 KBT0326         Keith B. Thompson       17-Sep-1982
{               Remove frb_ptr and other related s0 shareing stuff and
{               add stall_lock flag
{
{       V03-020 JWH0100         Jeffrey W. Horn         16-Sep-1982
{               Re-arrange FWA$T_JNLACE to include FWA$Q_BIJNL, FWA$Q_AIJNL
{               and FWA$Q_ATJNL so that the journal name lengths get written
{               with the journal names.
{
{       V03-019 JWH0007         Jeffrey W. Horn         16-Sep-1982
{               Add support for Recovery Unit locking:
{                       1.  Add IRB$L_IDENT, a process-wide unique identifier
{                       for each IRB.
{                       2.  Change RLB$W_OWNER to RLB$L_OWNER, which will now
{                       contain the value of IRB$L_IDENT instead of the ISI.
{                       3.  Add RLB$V_CONV to indicate lock needs to be
{                       converted to new mode.
{                       4.  Add RLB$V_LV2 a flag to indicate "level-2" RU
{                       record locking consistancy.
{
{       V03-018 TMK0006         Todd M. Katz            08-Sep-1982
{               Clean up the definitions of some fields in the index descriptor
{               definition, as they pertain to prologue 3 SIDRs.
{
{               Make the field IRB$B_SRCHFLAGS a word, and shift some of the
{               other fields around. Making this field a word allows
{               IRB$V_DEL_SEEN to be given its own bit instead of redefining a
{               bit (TMK0001). Also define a bit IRB$V_DUP_KEY in this same
{               field.
{
{       V03-017 TMK0005         Todd M. Katz            19-Aug-1982
{               Eliminate the field IRB$B_DIFF_CHAR.
{
{       V03-016 SHZ0001         Stephen H. Zalewski,    11-Aug-1982  21:26
{               Add a pointer to the GBSB (Global Buffer Synchronization
{               Block) in the IFAB.  Made GBH quadword aligned.
{
{++
{*****
{
{       NOTE:   All blocks MUST be longword aligned.
{
{       NOTE:   All blocks that are allocated the buffer management routine
{               must have a byte block size field as byte 9 from the start
{               of the block.
{
{*****
```

```
RM

/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*

ag

en

en
```

```
{
{       IFB field definitions
{
{       Internal fab (ifb)
{
{       There is one ifab (internal file access block) per open file
{

module $IFBDEF;

/*
/*  NOTE:  The fields thru JNLBDB inclusive are common between the ifb and irb
/*


aggregate IFBDEF structure fill prefix IFB$;
    FILL_1_OVERLAY union fill;
        FILL_1 quadword fill prefix IFBDEF tag $$;      /* device characteristic and bookkeeping bit vectors
        FILL_1_BITS structure fill;
            FILL_2 bitfield length 32 fill prefix IFBDEF tag $$;/* bookkeeping bits start in longword 2
                                                        /* (but have definitions that allow them to
                                                        /*  be referenced from the start of the ifab)
                                                        /*++
                                                        /*  the following bits are defined in
                                                        /*  common with the irab
                                                        /*
            BUSY bitfield;                              /* stream busy
            EOF bitfield;                               /* file positioned at eof
            PPF_IMAGE bitfield;                         /* flag for indirect processing of process-
                                                        /* permanent files (restricts allowable operations)
            ASYNC bitfield;                             /* async i/o flag (must be zero for ifab)
            ASYNCWAIT bitfield;                         /* wait on async i/o (must be zero for ifab)
                                                        /*--
                                                        /*
                                                        /*  ifab specific bits
                                                        /*
            ACCESSED bitfield;                          /* file is accessed
            ANSI_D bitfield;                            /* ansi d variable records
            RWC bitfield;                               /* copy of fop bit from open
            DMO bitfield;                               /* copy of fop bit from open
            SPL bitfield;                               /* copy of fop bit from open
            SCF bitfield;                               /* copy of fop bit from open
            DLT bitfield;                               /* copy of fop bit from open
            DFW bitfield;                               /* deferred write (copy of fop bit from $open)
            SQO bitfield;                               /* sequential operations only
            PPF_INPUT bitfield;                         /* this is command 'input' stream
            NFS bitfield;                               /* non-file structured flag
            WRTACC bitfield;                            /* logical or of fac bits:
                                                        /*   put, upd, del, trn
            MSE bitfield;                               /* multi-streams enabled
            CREATE bitfield;                            /* set if doing create (may be "create if")
            NORECLK bitfield;                           /* record locking not required
                                                        /* (i.e., no shared access or multi-stream)
            RW_ATTR bitfield;                           /* set if file attributes must be re-written
            TMP bitfield;                               /* temporary file (i.e., no directory entry)
            TEF bitfield;                               /* truncate at eof due to large auto extend
            STALL_LOCK bitfield;                        /* RMS is stalled for file lock
            SEQFIL bitfield;                            /* this is really a sequential file being shared
            SEARCH bitfield;                            /* search ifab - left during wildcard operations
            RMS_STALL bitfield;                         /* RMS is stalled on this file operation
            RESTART bitfield;                           /* Reopen or recreate operation in progress
            FILEFOUND bitfield;                         /* A file was found on a search operation
            DAP_OPEN bitfield;                          /* open/create function was performed via dap
            DAP bitfield;                               /* data access protocol transmission
            NSP bitfield;                               /* network services protocol transmission
        end FILL_1_BITS;
        FILL_1_FIELDS structure fill;
            PRIM_DEV longword unsigned;                 /* device characteristics bits
                                                        /* (for primary device - bit encoding same as for fab)
            BKPBITS longword unsigned;                  /* bookkeeping bits
/*
        end FILL_1_FIELDS;
    end FILL_1_OVERLAY;
    BID byte unsigned;                                  /* block id
    constant BID        equals 11  prefix IFB tag $C;   /* ifab id code
    BLN byte unsigned;                                  /* block length in longwords
    MODE byte unsigned;                                 /* caller's mode
    EFN byte unsigned;                                  /* event flag used for synchronous qio
    IOS_OVERLAY union fill;
        IOS longword unsigned;                          /* internal i/o status block
        BWB_OVERLAY union fill;
            BWB longword unsigned;                      /* bucket wait block for inter stream waiting
            BWB_FIELDS structure fill;
                FILL_8 byte dimension 2 fill prefix IFBDEF tag $$;
                IOS2 word unsigned;                     /* high word of io status block
            end BWB_FIELDS;
        end BWB_OVERLAY;
    end IOS_OVERLAY;
    IOS4 longword unsigned;                             /* 2nd longword of io status block
    ASBADDR longword unsigned;                          /* address of asynchronous context block
    ARGLST longword unsigned;                           /* user call parameters addr
    IRAB_LNK longword unsigned;                         /* pointer to irab(s)
    CHNL word unsigned;                                 /* i/o channel number
    FAC_OVERLAY union fill;
        FAC byte unsigned;                              /* file access
        FAC_BITS structure fill;
            PUT bitfield mask;                          /* (same as in fab's fac field)
            GET bitfield mask;
            DEL bitfield mask;
            UPD bitfield mask;
            TRN bitfield mask;
            BIO bitfield mask;
            BRO bitfield mask;
            EXE bitfield mask;
        end FAC_BITS;
                                                        /* note: if both bio and bro set, implies block i/o
                                                        /*       access only allowed for this connect, resets
                                                        /*       to bro on disconnect (seq. file org. only).
                                                        /*
    end FAC_OVERLAY;
    ORGCASE byte unsigned;                              /* copy of org for case dispatching
    LAST_FAB longword unsigned;                         /* address of fab for last operation
    IFI word unsigned;                                      /* Internal file Identifier, the one we gave to the user
    ECHO_ISI word unsigned;                                 /* ISI of stream to echo records from SYS$INPUT
    ATJNLBUF longword unsigned;                             /* address of IFAB audit trail buffer
    JNLBDB longword unsigned;                               /* address of Journaling BDB for FAB operations

/*-----*****

    EXTJNLBUF longword unsigned;                            /* pointer to buffer to contain extend journal record
    FWA_PTR longword unsigned;                              /* pointer to file work area control block
    NWA_PTR longword unsigned;                              /* pointer to network work area control block
    BDB_FLNK longword unsigned;                             /* pointer to bdb(s)
    BDB_BLNK longword unsigned;                             /* bdb backward link
    DEVBUFSIZ longword unsigned;                            /* device default (or bls if mt) buff size
    RTDEQ word unsigned;                                    /* run-time default extend quantity
    SHR byte unsigned;                                      /* File sharing bits from users FAB
    AGENT_MODE byte unsigned;                               /* User's FAB$V_FILE_MODE field, maximized with mode of caller

/*
/***********
/*
/*  the following fields must remain as is since
/*  they correspond to the rms attributes stored
/*  in the file header
/*

    RFMORG_OVERLAY union fill;
        RFMORG byte unsigned;                               /* organization and record format
        RFMORG_BITS structure fill;
            RFM bitfield length 4;                          /* record format (n.b. constant values defined in rfm field of fab)
            ORG bitfield length 4;                          /* file organization
        end RFMORG_BITS;
        constant SEQ    equals 0  prefix IFB tag $C;        /* sequential
        constant REL    equals 1  prefix IFB tag $C;        /* relative
        constant IDX    equals 2  prefix IFB tag $C;        /* indexed
        constant DIR    equals 3  prefix IFB tag $C;        /* direct
        constant MAXORG equals 2  prefix IFB tag $C;        /* release 1.5 maximum
    end RFMORG_OVERLAY;
    RAT byte unsigned;                                      /* record attributes (n.b. bit offsets defined in rat field of fab)
    LRL word unsigned;                                      /* longest record's length (or fixed record length)
    HBK_DISK longword unsigned;                             /* hi vbn allocated (note: disk format!)
    EBK_DISK longword unsigned;                             /* eof vbn (note: disk format!)
    FFB word unsigned;                                      /* first free byte in eof block
    BKS byte unsigned;                                      /* bucket size (! vbns)
    FSZ byte unsigned;                                      /* record header size for vfc
    MRS word unsigned;                                      /* max record size allowable
    DEQ word unsigned;                                      /* default extend quantity
    GBC word unsigned;                                      /* global buffer count
    constant FHAEND equals . prefix IFB$ tag K;             /* end of file header attributes
    constant FHAEND equals . prefix IFB$ tag C;             /* end of file header attributes
    FILL_4 word fill prefix IFBDEF tag $$;

/*-----*****

    DRT_REHIT byte unsigned;                                /* hit count for local dirty buffers.
    GBL_REHIT byte unsigned;                                /* rehit count for gbl buffers.
    constant KBUFNUM    equals 6  prefix IFB tag $C;        /* constant - the number of key buffers allocated
    FILL_5 word fill prefix IFBDEF tag $$;
    RNS_LEN_OVERLAY union fill;
        RNS_LEN longword unsigned;                      /* resultant name string length (used as a temp field by $search)
        LOCK_BDB longword unsigned;                     /* lock bdb address (used by $extend for rel. file)
    end RNS_LEN_OVERLAY;
    HBK longword unsigned;                              /* hi vbn allocated.
    EBK longword unsigned;                              /* eof vbn.
    SFSB_PTR longword unsigned;                         /* pointer to shared file synchronization block
    GBSB_PTR longword unsigned;                         /* pointer to global buffer synchronization block.
    PAR_LOCK_ID longword unsigned;                      /* Parent lock ID for bucket locks (get from SFSB.)
    AVLCL word unsigned;                                /* local buffers available.
    AVGBPB word unsigned;                               /* gbl ptr blocks available.
    GBH_PTR longword unsigned;                          /* pointer to global header.
    AS_DEV longword unsigned;                           /* assigned device characteristics
    FILL_6 longword fill prefix IFBDEF tag $$;          /* (spare) (* DO NOT RE-USE, Garbaged when filling in
                                                        /*                AS_DEV and ASDEVBSIZ *)
    ASDEVBSIZ longword unsigned;                        /* assigned device buffer size
    BLBFLNK longword unsigned;                          /* Forward link to BLB chain.
    BLBBLNK longword unsigned;                          /* Back link to BLB chain.
    JNLFLG_OVERLAY union fill;
        JNLFLG byte unsigned;                           /* journaling attribute flags
        JNLFLG_BITS structure fill;
            ONLY_RU bitfield mask;                      /* Recovery Unit journaling, no access outside RU
            RU bitfield mask;                           /* Recovery Unit journaling
            BI bitfield mask;                           /* Before Image journaling
            AI bitfield mask;                           /* After Image jouraling
            AT bitfield mask;                           /* Audit Trail journaling
            NEVER_RU bitfield mask;                     /* never do RU journaling
        end JNLFLG_BITS;
    end JNLFLG_OVERLAY;
    RECVRFLGS_OVERLAY union fill;
        RECVRFLGS byte unsigned;                        /* Recovery flags
        RECVRFLGS_BITS structure fill;
            RU_RECVR bitfield mask;                     /* Recovery Unit Rollback in progress
            AI_RECVR bitfield mask;                     /* AI Roll Forward Recovery in progress
            BI_RECVR bitfield mask;                     /* BI Roll Backward Recovery in progress
        end RECVRFLGS_BITS;
    end RECVRFLGS_OVERLAY;
    JNLFLG2_OVERLAY union fill;
        JNLFLG2 byte unsigned;                          /* Secondary journaling flags (generally operation specific)
        JNLFLG2_BITS structure fill;
            VALID_AT bitfield mask;                     /* AT entry in IFB buffer is valid and should be written
            JNL bitfield mask;                          /* Journaling Initialized for this file
            RUP bitfield mask;                          /* Recovery Unit in progress
            RU_RLK bitfield mask;                       /* Fake record locking during recovery unit
            DONE_ASS_JNL bitfield mask;                 /* Journal channels already assigned
        end JNLFLG2_BITS;
    end JNLFLG2_OVERLAY;
    FILL_7 byte fill prefix IFBDEF tag $$;              /* spare
    RJB longword unsigned;                              /* RMS Journaling Block address
    BUFFER_OFFSET word unsigned;                        /* ANSI buffer offset
    FILL_11 word fill prefix IFBDEF tag $$;             /*  for alignment
    constant BLN_SEQ equals . prefix IFB$ tag K;
    constant BLN_SEQ equals . prefix IFB$ tag C;
/*--
/*
```

```
/*  organization-dependent fields
/*
/*  the following fields are used differently
/*  depending upon the file's organization
/*
/*++
/*
/*  relative org specific fields
/*
end IFBDEF;

aggregate IFBDEF1 structure fill prefix IFB$;
    FILL_9 byte dimension 172 fill prefix IFBDEF tag $$;
    MRN longword unsigned;                              /* (rel) max record number
    DVBN longword unsigned;                             /* (rel) first data bucket vbn
    constant BLN_REL equals . prefix IFB$ tag K;
    constant BLN_REL equals . prefix IFB$ tag C;
/*--

/*++
/*
/*  indexed org specific fields
/*
end IFBDEF1;

aggregate IFBDEF2 structure fill prefix IFB$;
    FILL_10 byte dimension 172 fill prefix IFBDEF tag $$;
    IDX_PTR longword unsigned;                          /* (idx) pointer to primary key index descriptor
    AVBN byte unsigned;                                 /* (idx) vbn of 1st area descriptor
    AMAX byte unsigned;                                 /* (idx) total number of area descriptors
    NUM_KEYS byte unsigned;                             /* (idx) ! of keys in file
    UBUFSZ byte unsigned;                               /* (idx) update buffer size for keys
    KBUFSZ word unsigned;                               /* (idx) key buffer size
    EXTRABUF byte unsigned;                             /* (idx) number of extra buffers for 'cache'ing
    PLG_VER byte unsigned;                              /* (idx) prologue version number
    constant BLN_IDX equals . prefix IFB$ tag K;
    constant BLN_IDX equals . prefix IFB$ tag C;
    constant BLN equals . prefix IFB$ tag K;            /* ifab length
    constant BLN equals . prefix IFB$ tag C;            /* ifab length
/*--
end IFBDEF2;

end_module $IFBDEF;

module $IRBDEF;
```

```
/*
/*        IRB field definitions
/*
/*        Internal rab (irb)
/*
/*        There is 1 irab per connected record access stream
/*


/*
/*  NOTE: The fields thru JNLBDB inclusive are common between the irb and ifb
/*


aggregate IRBDEF structure fill prefix IRB$;
    FILL_1_OVERLAY union fill;
        FILL_1 quadword fill prefix IRBDEF tag $$;      /* used to get bookkeeping bit definitions
                                                        /* to apply from start of irab
        FILL_1_BITS0 structure fill;
            FILL_2 bitfield length 32 fill prefix IRBDEF tag $$;/* bookkeeping bits start in longword 2
                                                        /*++
                                                        /*
                                                        /*  the following bits are defined in common
                                                        /*  with the ifab
                                                        /*
            BUSY bitfield;                              /* file busy
            EOF bitfield;                               /* stream positioned at eof
            PPF_IMAGE bitfield;                         /* flag for indirect processing of process-
                                                        /* permanent file
            ASYNC bitfield;                             /* asynchronous i/o request
            ASYNCWAIT bitfield;                         /* $wait issued for asynchronous i/o request
                                                        /*--
                                                        /*
                                                        /*   irab specific bits
                                                        /*
            FIND_LAST bitfield;                         /* last operation was a find
            PUTS_LAST bitfield;                         /* last operation was a put sequential
            BIO_LAST bitfield;                          /* this/last operation is/was a block i/o operation
                                                        /* note: this bit is set only if mixed block and record
                                                        /*       operations (bro access).  after call to rm$rset
                                                        /*        refers to the current operation and bro_sw gives
                                                        /*        type of last operation.
            BRO_SW bitfield;                            /* switched from record operation to block i/o operation
            FIND bitfield;                              /* operation is a find
            RAHWBH bitfield;                            /* read ahead or write behind processing
            SKIP_NEXT bitfield;                         /* skip to next record flag for index fo
            DUP bitfield;                               /* duplicate records seen
            UNLOCK_RP bitfield;                         /* release lock on current (rp) record
            PPF_EOF bitfield;                           /* give one-shot rms$_eof error on sys$input
            PPF_SKIP bitfield;                          /* skip sys$input record ($deck), redoing $get
                                                        /* or $find on next record
            PPF_FNDSV bitfield;                         /* save value for find bit when ppf_skip set
            IDX_ERR bitfield;                           /* index update error occurred
            RRV_ERR bitfield;                           /* rrv update error occurred
            UPDATE bitfield;                            /* operation is an update (indexed)
            UPDATE_IF bitfield;                         /* operation was a $PUT -> $UPDATE
```

```
         ABOVELCKD bitfield;                      /* level above was locked by search_tree
         GBLBUFF bitfield;                        /* global buffers are in use.
         CON_EOF bitfield;                        /* file positioned at EOF by $CONNECT (isam)
         NO_Q_WAIT bitfield;                      /* do not wait for enqueues on query locks
         PPF_ECHO bitfield;                       /* echo SYS$INPUT records to SYS$OUTPUT
         RMS_STALL bitfield;                      /* RMS is stalled on this record operation
         RESTART bitfield;                        /* Reconnect operation in progress
         DAP_CONN bitfield;                       /* connect function was performed via dap
         RU_DELETE bitfield;                      /* recovery unit deletion in progress
         RU_UNDEL bitfield;                       /* recovery unit un-deletion in progress
         RU_UPDATE bitfield;                      /* place new record is special RU UPDATE format
      end FILL_1_BITS0;
/*
/*  the following are alternate definitions for alternate
/*  (non-conflicting) use of the above bits
/*
      FILL_1_BITS1 structure fill;
         FILL_3A byte dimension 5 fill prefix IRBDEF tag $$;
         FILL_3B bitfield length 1 fill prefix IRBDEF tag $$;/* start re-use with find
         WRITE bitfield;                          /* operation is a write
      end FILL_1_BITS1;
      FILL_1_FIELDS2 structure fill;
         IFAB_LNK longword unsigned;              /* pointer to ifab
         BKPBITS longword unsigned;               /* bookkeeping status bits
/*
      end FILL_1_FIELDS2;
   end FILL_1_OVERLAY;
   BID byte unsigned;                             /* block id
   constant BID       equals 10  prefix IRB tag $C;   /* irab code
   BLN byte unsigned;                             /* block length in longwords
   MODE byte unsigned;                            /* caller's mode
   EFN byte unsigned;                             /* event flag for synchronous io
   IOS_OVERLAY union fill;
      IOS longword unsigned;                      /* internal i/o status block
      BWB_OVERLAY union fill;
         BWB longword unsigned;                   /* bucket wait block for inter stream locking
         BWB_FIELDS structure fill;
            FILL_10 byte dimension 2 fill prefix IRBDEF tag $$;
            IOS2 word unsigned;                   /* high word of io status block
         end BWB_FIELDS;
      end BWB_OVERLAY;
   end IOS_OVERLAY;
   IOS4 longword unsigned,                        /* io status block (2nd longword)
   ASBADDR longword unsigned;                     /* address of permanent asynchronous context block
   ARGLST longword unsigned;                      /* user arg list address
                                                  /* if async, points to copy at head
                                                  /* of async context block

   IRAB_LNK longword unsigned;                    /* pointer to next irab
   CURBDB longword unsigned;                      /* current bdb address
   LAST_RAB longword unsigned;                    /* address of rab for last operation
   ISI word unsigned;                             /* Internal stream Identifier, the one we gave to the user
   FILL_4 word fill prefix IRBDEF tag $$;         /* spare - longword align
   ATJNLBUF longword unsigned;                    /* address of IRAB audit trail journaling buffer
   JNLBDB longword unsigned;                      /* address of journaling BDB for RAB operations
/*---------*****
   "IDENT" longword unsigned;                     /* process unique identifier for the IRB
```

```
    RLB_LNK longword unsigned;                          /* pointer to RLBs
    NXTBDB longword unsigned;                           /* next bdb address
    NRP_OVERLAY union fill;
        NRP longword unsigned;                          /* next record pointer (relative record number)
        NRP_VBN_OVERLAY union fill;
            NRP_VBN longword unsigned;                  /* next record pointer (relative)
            NRP_VBN_FIELDS structure fill;
                CACHEFLGS byte unsigned;                /* cacheflags for calls to getbkt,cache, etc. (indexed)
                STOPLEVEL byte unsigned;                /* level to stop at on tree search (indexed)
                SRCHFLAGS_OVERLAY union fill;
                    SRCHFLAGS word unsigned;            /* search flags (indexed)
                    SRCHFLAGS_BITS structure fill;
                        POSINSERT bitfield mask;        /* position for insert
                        SRCHGT bitfield mask;           /* approximate search gt
                        POSDELETE bitfield mask;        /* position for delete
                        NEW_IDX bitfield mask;          /* need to read in new idx dsc from file
                        SRCHGE bitfield mask;           /* approximate search ge
                        NORLS_RNF bitfield mask;        /* don't release bkt on rnf error, if set
                        FIRST_TIM bitfield mask;        /* flag to indicate 1st time for seq. processing
                        PRM bitfield mask;              /* flag to indicate that the permanence bit in the bdb
                                                        /*  should be set
                        DUP_KEY bitfield mask;          /* a duplicate key seen on scan of any data bucket
                        DEL_SEEN bitfield mask;         /* a deleted record has been encountered between current
                                                        /* and a next record during a $GET/$FIND
                        LAST_GT bitfield mask;          /* result of last search of compressed key bucket was GT
                    end SRCHFLAGS_BITS;
                end SRCHFLAGS_OVERLAY;
            end NRP_VBN_FIELDS;
        end NRP_VBN_OVERLAY;
    end NRP_OVERLAY;
    NRP_OFF_OVERLAY union fill;
        NRP_OFF longword unsigned;                      /* next record pointer offset (relative)
        CURVBN_OVERLAY union fill;
            CURVBN longword unsigned;                   /* vbn of current record (relative)
            NRP_OFF_OVERLAY1 union fill;
                NRP_OFF word unsigned;                  /* "
                SPL_BITS_OVERLAY union fill;
                    SPL_BITS byte unsigned;             /* bits for splitting (indexed)
                    FILL_5_OVERLAY union fill;
                        FILL_5 byte fill prefix IRBDEF tag $$;/* redefine bits
                        FILL_6_OVERLAY union fill;
                            FILL_6 byte fill prefix IRBDEF tag $$;
                            FILL_6_BITS structure fill;
                                BKT_NO_LO bitfield mask;/* low bit of bucket number processing
                                NEW_BKTS bitfield mask length 2;/* number of new buckets (0-3)
                                REC_W_LO bitfield mask; /* if splitting at pos_insert than rec goes w/ lo
                                CONT_BKT bitfield mask; /* middle bucket is a continuation bkt
                                CONT_R bitfield mask;   /* right bucket is a continuation bkt
                                EMPTY_BKT bitfield mask;/* bucket contains no data records
                                DUPS_SEEN bitfield mask;/* dups seen on scan of bucket, any key
                            end FILL_6_BITS;
                            FILL_5_BITS structure fill;
                                BKT_NO bitfield mask length 2;
                                BIG_SPLIT bitfield mask;
                            end FILL_5_BITS;
                            FILL_6_BITS structure fill;
```

```
                                SPL_IDX bitfield mask;  /* split up new index record and swing pointer
                                EMPT_SEEN bitfield mask;/* empty bucket passed over on posinsert
                            end FILL_6_BITS;
                        end FILL_6_OVERLAY;
                    end FILL_5_OVERLAY;
                end SPL_BITS_OVERLAY;
            end NRP_OFF_OVERLAY1;
        end CURVBN_OVERLAY;
    end NRP_OFF_OVERLAY;
    RP_OVERLAY union fill;
        RP longword unsigned;                           /* record pointer (relative record !)
        RP_VBN_OVERLAY union fill;
            RP_VBN longword unsigned;                   /* record pointer (relative)
            RP_VBN_FIELDS structure fill;
                POS_INS word unsigned;                  /* offset for position for insert for put (indexed)
                SPLIT word unsigned;                    /* first split point (indexed)
            end RP_VBN_FIELDS;
        end RP_VBN_OVERLAY;
    end RP_OVERLAY;
    RP_OFF_OVERLAY union fill;
        RP_OFF longword unsigned;                       /* record pointer offset
        LST_REC_OVERLAY union fill;
            LST_REC longword unsigned;                  /* last record address (indexed)
            PTR_VBN_OVERLAY union fill;
                PTR_VBN longword unsigned;              /* pointer vbn used by find_by_rrv (indexed)
                PTR_VBN_FIELDS structure fill;
                    RP_OFF_OVERLAY1 union fill;
                        RP_OFF word unsigned;           /* record pointer offset
                        SPLIT_1 word unsigned;          /* second split point -- 3-bkt split (indexed)
                    end RP_OFF_OVERLAY1;
                    SPLIT_2 word unsigned;              /* third split point -- 4-bkt split (indexed)
                end PTR_VBN_FIELDS;
            end PTR_VBN_OVERLAY;
        end LST_REC_OVERLAY;
    end RP_OFF_OVERLAY;
    OWNER_ID_OVERLAY union fill;
        OWNER_ID longword unsigned;                     /* owner id used for record locks
        OWNER_ID_FIELDS structure fill;
            OWN_ID word unsigned;                       /* index part of process id (pid)
            OWN_ISI_OVERLAY union fill;
                OWN_ISI word unsigned;                  /* isi value for this irab
                PPF_ISI byte unsigned;                  /* isi value for this process-permanent irab
            end OWN_ISI_OVERLAY;
        end OWNER_ID_FIELDS;
    end OWNER_ID_OVERLAY;
    BCNT byte unsigned;                                 /* i/o buffer count
    MBC byte unsigned;                                  /* multi-block count
    RSZ word unsigned;                                  /* record size from user
    RBF longword unsigned;                              /* user record buffer address
    MBF byte unsigned;                                  /* Multi-buffer count from user's RAB
    JNLFLG3_OVERLAY union fill;
        JNLFLG3 byte unsigned;                          /* IRB journaling flags
        JNLFLG3_BITS structure fill;
            VALID_AT bitfield mask;                     /* IRB MJB contains valid AT entry to write
        end JNLFLG3_BITS;
    end JNLFLG3_OVERLAY;
    FILL_7 word fill prefix IRBDEF tag $$;              /* spare to longword align
/*++
/*
/*  start of organization dependent fields
/*
/*++
/*
/* used by sequential and relative files
/*
    FILL_8 word fill prefix IRBDEF tag $$;              /* pad so longwords align

    CSIZ word unsigned;                                 /* current record size (seq)

/*++
/*
/*  relative org specific fields
/*
        constant BLN_REL equals . prefix IRB$ tag K;
        constant BLN_REL equals . prefix IRB$ tag C;

/*++
/*
/*  sequential org specific fields
/*
    TEMPO_OVERLAY union fill;
        TEMPO longword unsigned;
        TEMPO_FIELDS structure fill;
            ROVHDSZ_OVERLAY union fill;
                ROVHDSZ word unsigned;                  /* overhead size for record
                ROVHDSZ_FIELDS structure fill;
                    PRE_CCTL byte unsigned;             /* 'pre' carriage control
                    POST_CCTL byte unsigned;            /* 'post' carriage control
                end ROVHDSZ_FIELDS;
            end ROVHDSZ_OVERLAY;
            RTOTLSZ word unsigned;                      /* total size for record
        end TEMPO_FIELDS;
    end TEMPO_OVERLAY;
    TEMP1_OVERLAY union fill;
        TEMP1 longword unsigned;
        constant BLN_SEQ equals . prefix IRB$ tag K;
        constant BLN_SEQ equals . prefix IRB$ tag C;
        NVBNS byte unsigned;                            /* number of vbns transferred (nxtblk1)
/*
/* indexed org specific fields
/*
    end TEMP1_OVERLAY;
end IRBDEF;

aggregate IRBDEF1 structure fill prefix IRB$;
    FILL_11 byte dimension 96 fill prefix IRBDEF tag $$;
    KEYBUF longword unsigned;                           /* address of internal key buffer & update buffer
    UPDBUF longword unsigned;                           /* address of internal update buffer
    RECBUF longword unsigned;                           /* address of internal record buffer
    OLDBUF longword unsigned;                           /* address of internal old record buffer (updates only)
    RFA_VBN_OVERLAY union fill;
        RFA_VBN longword unsigned;                      /* save record vbn for nrp data
        UPD_BDB_OVERLAY union fill;
            UPD_BDB longword unsigned;                  /* save current bdb during insert operation
            LAST_VBN longword unsigned;                 /* last vbn at data level for update
        end UPD_BDB_OVERLAY;
    end RFA_VBN_OVERLAY;
    RFA_ID_OVERLAY union fill;
        RFA_ID word unsigned;                           /* save record id for search data
        LAST_ID word unsigned;                          /* id for udr during update (plg 3)
    end RFA_ID_OVERLAY;
    SAVE_POS word unsigned;                             /* save duplicate position for nrp data
    NEXT_VBN_OVERLAY union fill;
        NEXT_VBN longword unsigned;                     /* save next user data record VBN for nrp data
        PUTUP_VBN longword unsigned;                    /* RFA VBN of $PUT/$UPDATE record
    end NEXT_VBN_OVERLAY;
    FIRST_VBN longword unsigned;                        /* save SIDR first element VBN for search NRP data
    NEXT_ID_OVERLAY union fill;
        NEXT_ID word unsigned;                          /* save next user data record ID for nrp data
        PUTUP_ID word unsigned;                         /* ID of $PUT/$UPDATE record
    end NEXT_ID_OVERLAY;
    FIRST_ID word unsigned;                             /* save SIDR first element ID for search NRP data
    LOCK_BDB longword unsigned;                         /* lock bdb addr of level below on splits
    VBN_LEFT_OVERLAY union fill;
        VBN_LEFT longword unsigned;                     /* left vbn of split
        MIDX_TMP1 longword unsigned;                    /* temporary one for make index
    end VBN_LEFT_OVERLAY;
    VBN_RIGHT_OVERLAY union fill;
        VBN_RIGHT longword unsigned;                    /* right vbn of split
        MIDX_TMP2 longword unsigned;                    /* temporary two for make index
    end VBN_RIGHT_OVERLAY;
    VBN_MID_OVERLAY union fill;
        VBN_MID longword unsigned;                      /* middle vbn of split
        MIDX_TMP3_OVERLAY union fill;
            MIDX_TMP3 longword unsigned;                /* temporary three for make index
            NEXT_DOWN longword unsigned;                /* used by search_tree
        end MIDX_TMP3_OVERLAY;
    end VBN_MID_OVERLAY;
    REC_COUNT longword unsigned;                        /* number of current record in this bucket (plg 3)
    LST_NCMP longword unsigned;                         /* address of last key with zero front compression (plg 3)
    SPL_COUNT longword unsigned;                        /* number of the first record to be moved into new bucket
                                                        /*  when splitting indexes and SIDRs
    NID_RIGHT word unsigned;                            /* Next record ID of the right bucket
    NID_MID word unsigned;                              /* Next record ID of the middle bucket
    RFA_NID word unsigned;                              /* Next record ID of the RFA bucket
    KEYSZ byte unsigned;                                /* size of key in keybuffer !2
    FILL_9 byte fill prefix IRBDEF tag $$;              /* spare byte
    CUR_VBN longword unsigned;                          /* VBN of current record (primary/SIDR)
    POS_VBN longword unsigned;                          /* VBN of primary data record for NRP positioning
    UDR_VBN longword unsigned;                          /* VBN of current primary data record
    SIDR_VBN longword unsigned;                         /* SIDR array first element VBN of current record (SIDR)
    CUR_ID word unsigned;                               /* ID of current record (primary)
    POS_ID word unsigned;                               /* ID of primary data record for NRP positioning
    UDR_ID word unsigned;                               /* ID of current primary data record
    SIDR_ID word unsigned;                              /* SIDR array first element ID of current record (SIDR)
    CUR_COUNT word unsigned;                            /* SIDR array count of current record (SIDR)
    RP_KREF byte unsigned;                              /* Key of reference by which next record is retrieved
    CUR_KREF byte unsigned;                             /* Key of reference of current record (primary/SIDR)
    constant BLN_IDX equals . prefix IRB$ tag K;
    constant BLN_IDX equals . prefix IRB$ tag C;
end IRBDEF1;

end_module $IRBDEF;

module $ASBDEF;
```

```
/*
/*        ASB field definitions
/*
/*        Asynchronous context block (asb)
/*
/*        There is one asb per irab pointed to by irb$l_asbaddr allocated at
/*        connect and one per ifab which is dynamicaly allocated at stall
/*
/*        The asb$l_arglst is pointed to by the arglst field of the
/*        irab if the irb$v_async bookkeeping bit is set
/*
/*        All of the asb$c_bln_xxx must be longword aligned
/*


aggregate ASBDEF structure fill prefix ASB$;
    STKLEN word unsigned;                               /* save stack length (must be first word in block)
                                                        /*  STKLEN = BLN_org - BLN_FIX

    STKSIZ word unsigned;                               /* size of saved stack in bytes
    FILL_1 longword fill prefix ASBDEF tag $$;          /* spare
    BID byte unsigned;                                  /* block id
    constant BID        equals 13  prefix ASB tag $C;   /* asb id = 13
    BLN byte unsigned;                                  /* block length in longwords
    FILL_2 byte dimension 2 fill prefix ASBDEF tag $$;  /* spare
    ARGLST_OVERLAY union fill;
        ARGLST longword unsigned dimension 4;           /* saved argument list on async irab operations
        ARGLST_FIELDS structure fill;
            ARGCNT byte unsigned;                       /* argument count
                                                        /* value will be 0, 1, 2, or 3

            FILL_6 byte dimension 3 fill prefix ASBDEF tag $$;
            FABRAB longword unsigned;                   /* fab or rab address
            ERR longword unsigned;                      /* err routine addr
            SUC longword unsigned;                      /* suc routine addr
        end ARGLST_FIELDS;
    end ARGLST_OVERLAY;
    REGS longword unsigned dimension 5;                 /* save register area for regs 6, 7, 8, 10 and 11
    constant BLN_FIX equals . prefix ASB$ tag K;        /* block length of fixed asb
    constant BLN_FIX equals . prefix ASB$ tag C;        /* block length of fixed asb
                                                        /*  regs 4 and 5 are saved on stack
    STK longword unsigned dimension 35;                 /* saved stack area
    constant BLN_SEQ equals . prefix ASB$ tag K;        /* block length for seq org irab operations
    constant BLN_SEQ equals . prefix ASB$ tag C;        /* block length for seq org irab operations
    FILL_3 longword fill prefix ASBDEF tag $$;          /* additional space for relative org
    constant BLN_REL equals . prefix ASB$ tag K;        /* block length for rel org irab operations
    constant BLN_REL equals . prefix ASB$ tag C;        /* block length for rel org irab operations
    FILL_4 longword dimension 40 fill prefix ASBDEF tag $$;/* additional space for indexed org and FAB-related
    constant BLN_FAB equals . prefix ASB$ tag K;        /* block length for fab-related operations
    constant BLN_FAB equals . prefix ASB$ tag C;        /* block length for fab-related operations
    FILL_5 longword dimension 40 fill prefix ASBDEF tag $$;/* additional space for indexed org
    constant BLN_IDX equals . prefix ASB$ tag K;
    constant BLN_IDX equals . prefix ASB$ tag C;
end ASBDEF;

end_module $ASBDEF;
```

```
module $BDBDEF;
```

```
RM

/*
/*
/*
/*
/*
/*

co

en

mo
```

```
/*
/*       BDB field definitions
/*
/*      buffer descriptor block (bdb)
/*
/*      there is one bdb per i/o buffer
/*      ( the i/o buffers exist in separate pages, page aligned)
/*


aggregate BDBDEF structure fill prefix BDB$;
    FLINK longword unsigned;                        /* forward link
    BLINK longword unsigned;                        /* backward link
    BID byte unsigned;                              /* block id
    constant BID        equals 12  prefix BDB tag $C;  /* bdb id code
    BLN byte unsigned;                              /* block length in longwords
    FLGS_OVERLAY union fill;
        FLGS byte unsigned;                         /* bdb flags
        FLGS_BITS structure fill;
            VAL bitfield mask;                      /* buffer contents valid
            DRT bitfield mask;                      /* buffer content dirty
            IOP bitfield mask;                      /* buffer has i/o in progress
            PRM bitfield mask;                      /* buffer has permanence factor
            NOLOCATE bitfield mask;                 /* buffer shared - no locate mode
                                                    /* (set/cleared by rm$cache)
            WFO bitfield mask;                      /* other streams awaiting
                                                    /* the releasing of this bdb
            AST_DCL bitfield mask;                  /* ast has been declared for
                                                    /* waiting stream
        end FLGS_BITS;
    end FLGS_OVERLAY;
    CACHE_VAL_OVERLAY union fill;
        CACHE_VAL byte unsigned;                    /* relative value of buffer in cache
        VERTYP byte unsigned;                       /* version type (1 = wild)
    end CACHE_VAL_OVERLAY;
    USERS word unsigned;                            /* number of streams referencing this buffer
    BUFF_ID word unsigned;                          /* buffer identification number
    BLB_PTR longword unsigned;                      /* pointer to BLB chain for this BDB
    NUMB_OVERLAY union fill;
        NUMB word unsigned;                         /* ! of bytes of buffer in use
        DIRSEQ word unsigned;                       /* UCB$W_DIRSEQ at directory read time
    end NUMB_OVERLAY;
    ˉIZE word unsigned;                             /* ! bytes in buffer
    ADDR longword unsigned;                         /* address of buffer
    VBN longword unsigned;                          /* 1st vbn in buffer
    VBNSEQNO_OVERLAY union fill;
        VBNSEQNO longword unsigned;                 /* vbn seq number of validity check vs. bcb copy
        LAST longword unsigned;                     /* address of last directory record
    end VBNSEQNO_OVERLAY;
    WAIT_OVERLAY union fill;
        WAIT longword unsigned;                     /* wait thread (irab addr)
                                                    /* (for inter-stream intra-
                                                    /*  process locking only)
        VERCOUNT longword unsigned;                 /* negative count of version entries scanned
    end WAIT_OVERLAY;
    ALLOC_ADDR longword unsigned;                          /* buffer allocation addr
    ALLOC_SIZE word unsigned;                              /* buffer allocation size
    FILL_T word fill prefix BDBDEF tag $$;                 /* spare
    BI_BDB longword unsigned;                              /* address of isam/block i/o bi journaling BDB
    AI_BDB longword unsigned;                              /* address of isam/block i/o ai journaling BDB
    JNLSEQ character length 16;                            /* Journaling Sequence Number Block
    WK1_OVERLAY union fill;
        WK1 longword unsigned;                             /* work area
        WK1_FIELDS structure fill;
            REL_VBN byte unsigned;                         /* current vbn rel to start of buffer
            VAL_VBNS byte unsigned;                        /* ! of valid vbns in buffer
            PRE_CCTL byte unsigned;                        /* unit record carriage control byte ('pre')
            POST_CCTL byte unsigned;                       /* unit record carriage control byte ('post')
        end WK1_FIELDS;
    end WK1_OVERLAY;
    CURBUFADR longword unsigned;                           /* current buffer addr
end BDBDEF;

aggregate BDBDEF1 structure fill prefix BDB$;
    FILL_2 byte dimension 72 fill prefix BDBDEF tag $$;
    IOSB_OVERLAY union fill;
        IOSB longword unsigned dimension 2;                /* i/o status block for buffer
        constant BLN equals . prefix BDB$ tag K;           /* length of bdb block
        constant BLN equals . prefix BDB$ tag C;           /* length of bdb block
        IOSB_FIELDS structure fill;
            VERSION longword unsigned;                     /* addr of current/next directory version entry
            RECORD longword unsigned;                      /* address of current/next directory record
        end IOSB_FIELDS;
    end IOSB_OVERLAY;
end BDBDEF1;

end_module $BDBDEF;

module $GBPBDEF;
```

```
/*
/*        GBPB field definitions
/*
/*        Global Buffer Pointer Block (GBPB)
/*
/*        The GBPB is the process local structure used in conjunction with
/*        shared global i/o buffers.  In order to  inimize the impact of
/*        global buffers on existing code, the GBPB is identical to a BDB
/*        in those fields which are referenced outside of the RMSCACHE and
/*        RMSRELEASE routines.
/*


aggregate GBPBDEF structure fill prefix GBPB$;
    FLINK longword unsigned;                                /* forward link
    BLINK longword unsigned;                                /* backward link
    BID byte unsigned;                                      /* block id
    constant BID        equals 21  prefix GBPB tag $C;      /* gbpb id code
    BLN byte unsigned;                                      /* block length in longwords
    FLGS byte unsigned;                                     /* gbpb flags (use BDB flgs definitions)
    CACHE_VL byte unsigned;                                 /* relative cache value of this buffer
    USERS word unsigned;                                    /* number of streams referencing this buffer
    BUFF_ID word unsigned;                                  /* buffer identification number
    BLB_PTR longword unsigned;                              /* pointer to BLB chain for this GBPB
    NUMB word unsigned;                                     /* ! of bytes of buffer in use
    SIZE word unsigned;                                     /* ! bytes in buffer
    ADDR longword unsigned;                                 /* address of buffer
    VBN longword unsigned;                                  /* 1st vbn in buffer
    VBNSEQNO longword unsigned;                             /* sequence number field.
    GBD_PTR longword unsigned;                              /* Pointer to the GBD for this buffer.
    constant BLN equals . prefix GBPB$ tag K;               /* Length of GBPB block
    constant BLN equals . prefix GBPB$ tag C;               /* Length of GBPB block
end GBPBDEF;

end_module $GBPBDEF;

module $RLBDEF;
```

```
/*
/*        RLB field definitions
/*
/*        record lock block (rlb)
/*
/*        The rlb describes one locked record for a particular
/*        process-record stream (rab/irab). if the owner field
/*        is 0 then the rlb is available for use. otherwise, it
/*        describes a locked record. note: when owner is 0 the
/*        record rfa fields are zeroed (0).
/*
/*
/*                +----------------------------------------+
/*        rlb:     !                  link                  !
/*                +-----------------+----------------------+
/*                !                  !         rfa4  id           !
/*                +---------+--------+----------+----------+
/*                !  flags  !reserved!   bln    ,   bid    !
/*                +---------+--------+----------+----------+
/*                ,                  rfa0                  !
/*                +----------------------------------------+
/*                 !                  owner                              !
/*                 +----------------------------------------+
/*          sb:    ! Still to be def- !  VMS status code    !
/*                ! ined status bits !                      !
/*                +----------------------------------------+
/*                !  Lock Id. (Returned for new locks,     !
/*                !             input for conversions)     !
/*                +----------------------------------------+
/*


aggregate RLBDEF structure fill prefix RLB$;
    LNK longword unsigned;                                /* link to next rlb
    MISC_OVERLAY union fill;
        MISC longword unsigned;                           /* longword definition to optimize clearing field
        MISC_FIELDS structure fill;
            FLAGS2_OVERLAY union fill;
                FLAGS2 word unsigned;                     /* more flag bits
                FLAGS2_BITS structure fill;
                    TIMER_INPROG bitfield mask;           /* Timer queued.
                end FLAGS2_BITS;
            end FLAGS2_OVERLAY;
            RFA4_OVERLAY union fill;
                RFA4 word unsigned;                       /* 3'rd word of records rfa
                                                          /* offset for seq f.o. (bits 0:14)
                                                          /* always 0 for rel f.o. (bits 0:14)
                ID word unsigned;                         /* id for idx f.o.
            end RFA4_OVERLAY;
        end MISC_FIELDS;
    end MISC_OVERLAY;
    BID byte unsigned;                                    /* block id
    constant BID        equals 14  prefix RLB tag $C;     /* rlb code
    BLN byte unsigned;                                    /* block length in longwords
    TMO byte unsigned;                                    /* propagation of ROP TMO field
    FLAGS_OVERLAY union fill;
        FLAGS byte unsigned;                            /* various locking flags
        FLAGS_BITS structure fill;
            WAIT bitfield mask;                         /* propagation of ROP WAT bit
            CR bitfield mask;                           /* defines lock manager mode "concurrent read"
                                                        /* used to query lock databa.e for records
            PW bitfield mask;                           /* allow reader access to locked record flag
                                                        /* indicate "lock for write, allow readers"
            PR bitfield mask;                           /* used to query lock database
            CONV bitfield mask;                         /* defines lock manager option "convert"
            LV2 bitfield mask;                          /* sets lock as "level 2" RU consistancy
            FAKE bitfield mask;                         /* this RLB contains no lock.
            TMO bitfield mask;                          /* propagation of ROP TMO bit
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    RFAO longword unsigned;                             /* 1'st and 2'nd words of record's rfa
                                                        /* seq f.o. vbn
                                                        /* rel f.o. relative record number
                                                        /* idx f.o. start vbn
    OWNER longword unsigned;                            /* identification of owning stream
    LKSB_OVERLAY union fill;
        LKSB longword unsigned;                         /* first longword of lock status block
        LKSB_FIELDS structure fill;
            STATUS word unsigned;                       /* VMS status code
            S_BITS word unsigned;                       /* various status bits
        end LKSB_FIELDS;
    end LKSB_OVERLAY;
    LOCK_ID longword unsigned;                          /* second longword of lksb is lock_id
    constant BLN equals . prefix RLB$ tag K;            /* length of rlb
    constant BLN equals . prefix RLB$ tag C;            /* length of rlb
end RLBDEF;

end_module $RLBDEF;

module $FLBDEF;
```

```
/*
/*  file lock block definitions
/*

aggregate FLBDEF structure fill prefix FLB$;
    FLB_LNK longword unsigned;                          /* pointer to next FLB
    RLB_LNK longword unsigned;                          /* pointer to RLBs
    BID byte unsigned;                                  /* block id
    constant BID        equals 23  prefix FLB tag $C;
    BLN byte unsigned;                                  /* block length
    FILL_1 word fill prefix FLBDEF tag $$;              /* spare
    IFB_PTR longword unsigned;                          /* IFAB address
    LOCK_ID longword unsigned;                          /* lock id
    constant BLN equals . prefix FLB$ tag K;
    constant BLN equals . prefix FLB$ tag C;
end FLBDEF;

end_module $FLBDEF;

module $DRCDEF;
```

```
/*
/*  directory cache node definitions
/*


aggregate DRCDEF structure fill prefix DRC$;
    NXTFLNK longword unsigned;                          /* link to next entry, this level
    NXTBLNK longword unsigned;                          /* link to previous entry, this level
    LVLFLNK longword unsigned;                          /* link to first entry, next lower level
    LVLBLNK longword unsigned;                          /* link to last entry, next lower level
                                                        /* note: the links are maintained in lru order
    NAME character length 40;                           /* directory name or device and unit
                                                        /* note: stored as counted string counting count itself
    DID_OVERLAY union fill;
        DID word unsigned dimension 3;                  /* file id for directory
        constant BLN equals . prefix DRC$ tag K;        /* length of directory cache node
        constant BLN equals . prefix DRC$ tag C;        /* length of directory cache node
        DID_FIELDS structure fill;
            FILL_1 byte dimension 2 fill prefix DRCDEF tag $$;
            DIRSEQ word unsigned;                       /* directory sequence ! for device node
        end DID_FIELDS;
    end DID_OVERLAY;
end DRCDEF;

end_module $DRCDEF;

module $RLSDEF;
```

```
/*
/*                release option flag definitions
/*


aggregate RLSDEF  union fill prefix RLS$;
    RLSDEF_BITS structure fill;
        RETURN bitfield mask;                         /* return buffer and bdb to free space lists
        WRT_THRU bitfield mask;                       /* write buffer if dirty
        KEEP_LOCK bitfield mask;                      /* keep bdb locked
        DEQ bitfield mask;                            /* always release lock
    end RLSDEF_BITS;
end RLSDEF;

end_module $RLSDEF;

module $CSHDEF;
```

```
/*
/*               cache option flag definitions
/*


aggregate CSHDEF  union fill prefix CSH$;
    CSHDEF_BITS structure fill;
        LOCK bitfield mask;                              /* obtain exclusive access to block
        NOWAIT bitfield mask;                            /* do not wait for block on access interlock
                                                         /* collision
        NOREAD bitfield mask;                            /* do not read in block
        NOBUFFER bitfield mask;                          /* obtain access to block but don't allocate
                                                         /* a buffer for it and don't read it
    end CSHDEF_BITS;
end CSHDEF;

end_module $CSHDEF;

module $PIODEF;
```

```
/*
/*
/*  rms overall status bit definitions
/*


aggregate PIODEF  union fill prefix PIO$;
    PIODEF_BITS structure fill;
        INHAST bitfield;                                /* set if asts implicitly inhibited
                                                        /* if reset by disabled ast, ast must be re-
                                                        /* enabled
        EOD bitfield;                                   /* set if searching for 'eod' string on 'input'
        SYNC1 bitfield;                                 /* sync stalled operation using efn 27
        SYNC2 bitfield;                                 /* sync stalled operation using efn 28
    end PIODEF_BITS;
end PIODEF;

end_module $PIODEF;

module $FTLDEF;
```

```
/*
/*        definitions for rms debug failure codes
/*
/*
/*  the following codes are for temporary bug check tests, and are
/*  internal to rms.  all of the codes are negative, implying that they
/*  do not return to the caller, probably killing the process (if not
/*  the entire system).
/*

constant SETPRTFAIL     equals -1  prefix FTL tag $;    /* set protection system service failed (rm0bufmgr)
constant STKTOOBIG      equals -2  prefix FTL tag $;    /* stack too big for asb (rm0stall)
constant BADIFAB        equals -3  prefix FTL tag $;    /* invalid ifab or irab (rm0fset,rm0conn,rm0rset,rm0prflnm)
constant GTCHNFAIL      equals -4  prefix FTL tag $;    /* get channel system service failure (rm0prflnm)
constant BADORGCASE     equals -5  prefix FTL tag $;    /* invalid orgcase value for dispatch (all rms$
                                                        /* level routines execept open and create)
constant BADBDB         equals -6  prefix FTL tag $;    /* block not a bdb (rm0bufmgr)
constant ASBALLFAIL     equals -7  prefix FTL tag $;    /* couldn't allocate an asb (rm0stall)
constant BADASTPRM      equals -8  prefix FTL tag $;    /* ast parameter not a valid ifab/irab addr (rm0stall)
constant CANTDOAST      equals -9  prefix FTL tag $;    /* couldn't redeclare ast (insf. mem.) (rm0stall)
constant NOSTRUCT       equals -10  prefix FTL tag $;   /* rab or fab not same on ast (rm0stall)
constant NOASB          equals -11  prefix FTL tag $;   /* asb not allocated or stream not busy on ast (rm0stall)
constant NONXTBDB       equals -12  prefix FTL tag $;   /* no next bdb available (rm1seqxfr)
constant BADBUFSIZ      equals -13  prefix FTL tag $;   /* disk buffer size not = 512 (rm1conn)
constant ENQDEQFAIL     equals -14  prefix FTL tag $;   /* enq or deq service failed (rm0reclck)
constant NOCURBDB       equals -15  prefix FTL tag $;   /* no current bdb before calling rm$release (rm0reclck)
constant NOPARENT       equals -16  prefix FTL tag $;   /* no parent lock available for global buffer section lock (rm0share)
constant DEALLERR       equals -17  prefix FTL tag $;   /* ifab deallocation attempted with other block(s)
                                                        /* still allocated (rms0close)
constant IORNDN         equals -18  prefix FTL tag $;   /* i/o rundown inconsistency (either ifab or irab
                                                        /* table entries not zeroed) (rms0rndwn)
constant XFERSIZE       equals -19  prefix FTL tag $;   /* size of requested transfer not equal to
                                                        /* or less than the current number of bytes
                                                        /* in use for the bdb (rm0cache)
constant NOTLOCKED      equals -20  prefix FTL tag $;   /* bdb not locked and a keep lock request
                                                        /* was made on a release request.
constant NODIDORFID     equals -21  prefix FTL tag $;   /* neither a fid nor a did was set upon exit from
                                                        /* rm$setdid (rms0erase)
constant RELEASFAIL     equals -22  prefix FTL tag $;   /* release of non-dirty bdb failed (rm0xtnd23,rms0extend)
constant NOLOCKBDB      equals -23  prefix FTL tag $;   /* no lock bdb found (rm0xtnd23)
constant NONETWORK      equals -24  prefix FTL tag $;   /* network routine entered but no network support in rms
constant LOCKFAILED     equals -25  prefix FTL tag $;   /* failed to lock prolog (rm2create)
constant BADLEVEL       equals -26  prefix FTL tag $;   /* to search by id, structure level must be 0
constant ASTDECERR      equals -27  prefix FTL tag $;   /* ast declaration for file sharing failed
constant BADGBLCNT      equals -28  prefix FTL tag $;   /* Zero global buffer count found when not expected (rm1conn)
constant ACCNTOVFLO     equals -29  prefix FTL tag $;   /* access count overflow (rm0share)
constant BDBAVAIL       equals -30  prefix FTL tag $;   /* BDB was available and shouldn't have been.
constant GBLNOLK        equals -31  prefix FTL tag $;   /* Record locking was not set with global buffers.
constant LCKFND         equals -32  prefix FTL tag $;   /* A lock was found and we don't know what to do.
constant NOBLB          equals -33  prefix FTL tag $;   /* No BLB was found and there should have been one.
constant NOGBPB         equals -34  prefix FTL tag $;   /* No GBPB was found and should have been.
constant NOLCLBUF       equals -35  prefix FTL tag $;   /* Should have found a local buffer.
constant NORDNOTSET     equals -36  prefix FTL tag $;   /* NOREAD not set when NOBUFFER was.
constant NOTGBPB        equals -37  prefix FTL tag $;   /* Found an illegit BDB.
constant NOSFSB         equals -38  prefix FTL tag $;   /* No SFSB when allocating BLB.
constant LOCKHELD       equals -39  prefix FTL tag $;    /* Attempted to return a BLB with lock_id neq 0
constant RLSDRT         equals -40  prefix FTL tag $;    /* Dirty buffer found in releasall.
constant BADBLB         equals -41  prefix FTL tag $;    /* Bad BLB found in blocking AST routine.
constant BADOWNER       equals -42  prefix FTL tag $;    /* Owner field in BLB is bad in blocking AST routine.
constant GETLKIFAIL     equals -43  prefix FTL tag $;    /* $GETLKIW failed in last chance (rms0lstch).
constant BADEBKHBK      equals -44  prefix FTL tag $;    /* tried to store an invalid EBK/HBK (rm0share).


end_module $FTLDEF;

module $BUGDEF;
```

```
/*
/*  the following internal codes are for non-fatal bug check reporting.
/*  these codes are positive byte values.  they trigger a reporting action
/*  and return to the caller with r0 set to rms$_bug+<8*the bug code>,
/*  which is an externally documented rms error code.
/*

constant BADDFLTDIR     equals 1  prefix BUG tag $;     /*DEFAULT DIRECTORY STRING INVALID (RMOXPFN)


end_mouule $BUGDEF;

module $IDXDEF;
```

```
/*
/*        IDX field definitions
/*
/*        index descriptor definition
/*
/*        An index descriptor block exists for each key of reference in use.
/*        they are not necessarily contiguous in memory.
/*


aggregate IDXDEF structure fill prefix IDX$;
    IDXFL longword unsigned;                            /* forward link to next index descriptor
    FILL_1 longword fill prefix IDXDEF tag $$;          /* spare
    BID byte unsigned;                                  /* block id
    constant BID        equals 15  prefix IDX tag $C;   /* id for index descriptor block
    BLN byte unsigned;                                  /* length of block
    VBN longword unsigned;                              /* VBN where the descriptor came from
    OFFSET word unsigned;                               /* Offset into the block (VBN) of the descriptor
    DESC_NO byte unsigned;                              /* Descriptor number (index into update buffer)
    FILL_2 byte fill prefix IDXDEF tag $$;              /* spare
    IANUM byte unsigned;                                /* area number for index buckets
    LANUM byte unsigned;                                /* area number for lower index buckets
    DANUM byte unsigned;                                /* area number for data buckets
    ROOTLEV byte unsigned;                              /* level of root
    IDXBKTSZ byte unsigned;                             /* size of index bucket in vbn's
    DATBKTSZ byte unsigned;                             /* size of data bucket in vbn's
    ROOTVBN longword unsigned;                          /* start vbn of root bucket
    FLAGS_OVERLAY union fill;
        FLAGS byte unsigned;                            /* index/key flags
        FLAGS_BITS0 structure fill;
            DUPKEYS bitfield mask;                      /* duplicate keys allowed
            CHGKEYS bitfield mask;                      /* keys can change values
            NULKEYS bitfield mask;                      /* null key value allowed
            IDX_COMPR bitfield mask;                    /* index is compressed
            INITIDX bitfield mask;                      /* index is not initialized
            FILL_3 bitfield fill prefix IDXDEF tag $$;  /* spare
            KEY_COMPR bitfield mask;                    /* key has been compressed
        end FLAGS_BITS0;

        FLAGS_BITS1 structure fill;
            FILL_4 bitfield fill prefix IDXDEF tag $$;  /* space over dupkeys
            FILL_5 bitfield length 2 fill prefix IDXDEF tag $$;
            FILL_6 bitfield fill prefix IDXDEF tag $$;  /* space over idx_compr
            FILL_7 bitfield fill prefix IDXDEF tag $$;  /* space over initidx
            FILL_8 bitfield fill prefix IDXDEF tag $$;  /* spare
            FILL_9 bitfield fill prefix IDXDEF tag $$;  /* space over key_compr
            REC_COMPR bitfield mask;                    /* data record is in compressed form
        end FLAGS_BITS1;

    end FLAGS_OVERLAY;
    DATATYPE byte unsigned;                             /* data type of key field
    constant STRING     equals 0  prefix IDX tag $C;    /* string data type
    constant SGNWORD    equals 1  prefix IDX tag $C;    /* signed binary word
    constant UNSGNWORD  equals 2  prefix IDX tag $C;    /* unsigned binary word
    constant SGNLONG    equals 3  prefix IDX tag $C;    /* signed binary long word
    constant UNSGNLONG  equals 4  prefix IDX tag $C;     /* unsigned binary long word
    constant PACKED     equals 5  prefix IDX tag $C;     /* packed decimal
    constant SGNQUAD    equals 6  prefix IDX tag $C;     /* signed binary quadword
    constant UNSGNQUAD  equals 7  prefix IDX tag $C;     /* unsigned binary quadword
    SEGMENTS byte unsigned;                              /* number of key field segments
    NULLCHAR byte unsigned;                              /* null character
    KEYSZ byte unsigned;                                 /* total key size
    KEYREF byte unsigned;                                /* key of reference(0-primary)
    MINRECSZ word unsigned;                              /* minimum record size
    IDXFILL word unsigned;                               /* index fill
    DATFILL word unsigned;                               /* data fill
    IDXBKTYP byte unsigned;                              /* PLG3 - type of index bucket and SIDR bucket
    constant V2_BKT     equals 0  prefix IDX tag $C;     /* Prologue two bucket
    constant CMPIDX     equals 1  prefix IDX tag $C;     /* Prologue 3, index keys are compressed
    constant NCMPIDX    equals 2  prefix IDX tag $C;     /* Prologue 3, index keys are not compressed
    DATBKTYP byte unsigned;                              /* PLG3 - type of primary data bucket
    constant CMPCMP     equals 3  prefix IDX tag $C;     /* Prologue 3, primary key is compressed, data
                                                         /*  is compressed
                                                         /* Prologue 3, SIDR key is compressed
    constant CMPNCMP    equals 4  prefix IDX tag $C;     /* Prologue 3, primary key is compressed,
                                                         /*  data is not compressed
    constant NCMPCMP    equals 5  prefix IDX tag $C;     /* Prologue 3, primary key is not compressed
                                                         /*  data is compressed
    constant NCMPNCMP   equals 6  prefix IDX tag $C;     /* Prologue 3, primary key is not compressed
                                                         /*  data is not compressed
                                                         /* Prologue 3, SIDR key is compressed
    FILL_10 word fill prefix IDXDEF tag $$;              /* spare
    constant FIXED_BLN equals . prefix IDX$ tag K;
    constant FIXED_BLN equals . prefix IDX$ tag C;
/*
/* the following is the length of the fixed part of the index descriptor
/*

/*
/* the following is repeated for each key segment
/*
    POSITION word unsigned;                              /* key segment position
    SIZE byte unsigned;                                  /* key segment size (plg 3)
    TYPE byte unsigned;                                  /* key segment datatype (plg 3)
end IDXDEF;

end_module $IDXDEF;

module $UPDDEF;
```

```
/*
/* update buffer flags
/*


aggregate UPDDEF union fill prefix UPD$;
    FLAGS byte unsigned;
        FLAGS_BITS structure fill;
            INS_NEW bitfield mask;                          /* alternate key to be inserted from record buffer
            OLD_DEL bitfield mask;                          /* delete this key value using old record
        end FLAGS_BITS;
end UPDDEF;

end_module $UPDDEF;

module $GBHDEF;
```

```
/*
/*      GBH field definitions
/*
/*      Global Buffer Header (GBH)
/*
/*      There is a Global Buffer Header for every file's global buffer section.
/*
/*      ***  WARNING - THIS STRUCTURE MUST BE QUADWORD ALIGNED  ***
/*


aggregate GBHDEF structure fill prefix GBH$;
    GBD_FLNK longword unsigned;                         /* Self relative queue header for GBD's
    GBD_BLNK longword unsigned;
    BID byte unsigned;                                  /* Block ID
    constant BID        equals 17  prefix GBH tag $C;   /* Block ID code for GBH
    BLN byte unsigned;                                  /* Length of GBH in longwords
    TRC_FLGS_OVERLAY union fill;
        TRC_FLGS word unsigned;                         /* Trace flags (set to trace given function)
        TRC_FLGS_BITS structure fill;
            CACHE_IN bitfield mask;                     /* Cache inputs
            CACHE_OUT bitfield mask;                    /* Cache outputs
            RLS_IN bitfield mask;                       /* Release inputs
            RLS_OUT bitfield mask;                      /* Release outputs
            QIO_START bitfield mask;                    /* Qio inputs
            QIO_DONE bitfield mask;                     /* Qio outputs
            STALL bitfield mask;                        /* Stall inputs
            THREADGO bitfield mask;                     /* Stall outputs
            BLB_ENQ bitfield mask;                      /* Bucket lock ENQ inputs
            BLB_GRANT bitfield mask;                    /* Bucket lock grant status
            BLB_DEQ bitfield mask;                      /* Bucket lock DEQ request
            BLB_BLOCK bitfield mask;                    /* Blocking AST received
            F1 bitfield mask;
            F2 bitfield mask;
            F3 bitfield mask;
            F4 bitfield mask;
        end TRC_FLGS_BITS;

    end TRC_FLGS_OVERLAY;
    HI_VBN longword unsigned;                           /* Highest possible VBN value (FFFFFFFF).
    GS_SIZE longword unsigned;                          /* Size of total section in bytes.
    LOCK_ID longword unsigned;                          /* Lock ID of system file lock.
    GS_LOCK_ID longword unsigned;                       /* Lock ID of system global section lock.
    USECNT longword unsigned;                           /* Accessor count for section.
    TRC_FLNK longword unsigned;                         /* Trace blocks forward link
    TRC_BLNK longword unsigned;                         /* Trace blocks back link
    GBD_START longword unsigned;                        /* Offset to first GBD.
    GBD_END longword unsigned;                          /* Offset to last GBD.
    GBD_NEXT longword unsigned;                         /* Offset to next cache victim GBD.
    SCAN_NUM longword unsigned;                         /* Number of GBD's to scan for victim.
/*
/* Global buffer statistics section
/*
    HIT longword unsigned;                              /* Buffer found in global cache
    MISS longword unsigned;                             /* Buffer not found in global cache
    READ longword unsigned;                             /* Buffer read from disk into cache
    WRITE longword unsigned;                            /* Buffer written from cache to disk
    DFW_WRITE longword unsigned;                        /* Deferred writeback from cache to disk
    CROSS_HIT longword unsigned;                        /* Cross process hit count.
    OUTBUFQUO longword unsigned;                        /* Count of times GBLBUFQUO limit was hit.
    FILL_1 longword unsigned;                           /* Force quadword alignment
    constant BLN equals . prefix GBH$ tag K;            /* Length of global buffer header structure
    constant BLN equals . prefix GBH$ tag C;            /* Length of global buffer header structure
end GBHDEF;

end_module $GBHDEF;

module $TRCDEF;
```

**F

```
/*
/*      TRC field definitions
/*
/*      Trace block structure (TRC)
/*
/*      Tracing saves at specific points in the RMS code for debugging and
/*      algorithm analysis purposes.
/*
/*      ***  WARNING - THIS STRUCTURE MUST BE QUADWORD ALIGNED  ***
/*


aggregate TRCDEF structure fill prefix TRC$;
    FLNK longword unsigned;                             /* Trace block forward link
    BLNK longword unsigned;                             /* Trace block back link
    BID byte unsigned;                                  /* Block ID
    constant BID        equals 18  prefix TRC tag $C;   /* Trace block code
    BLN byte unsigned;                                  /* Length of block in longwords
    FUNCTION word unsigned;                             /* Function code (see GBH definitions)
    "STRUCTURE" longword unsigned;                      /* ifab/irab address.
    PID word unsigned;                                  /* Process ID
    SEQNUM word unsigned;                               /* Sequence number.
    VBN longword unsigned;                              /* VBN requested.
    RETURN1 longword unsigned;                          /* Address of caller.
    RETURN2 longword unsigned;                          /* Caller's caller.
    ARGS_OVERLAY union fill;
        ARGS longword unsigned dimension 8;             /* Function specific arguments
        constant BLN equals . prefix TRC$ tag K;        /* NOTE: should be quadwords multiple to
        constant BLN equals . prefix TRC$ tag C;        /* NOTE: should be quadwords multiple to
        ARGS_FIELDS structure fill;
            ARG_FLG longword unsigned;                  /* Argument flags (R3).
            BDB_ADDR longword unsigned;                 /* BDB address.
            BDB_USERS word unsigned;                    /* Use count from BDB.
            BDB_BUFF word unsigned;                     /* BDB buffer ID.
            BDB_CACHE byte unsigned;                    /* BDB cache value.
            BDB_FLAGS byte unsigned;                    /* Status flags from BDB.
            BDB_SEQ longword unsigned;                  /* Sequence number from BDB.
            BLB_MODE byte unsigned;                     /* Mode held in BLB.
            BLB_FLAGS byte unsigned;                    /* Flags from BLB.
            BLB_ADDR longword unsigned;                 /* Address of BLB.
            BLB_LOCK longword unsigned;                 /* Lock ID from BLB.
            BLB_SEQ longword unsigned;                  /* Sequence number from BLB.
                                                        /* maintain quad alignment on header
        end ARGS_FIELDS;
    end ARGS_OVERLAY;
end TRCDEF;

end_module $TRCDEF;

module $GBDDEF;
```

```
/*
/*       GBD structure definitions
/*
/*       Global Buffer Descriptor (GBD)
/*
/*       There is a single GBD for every buffer in a global buffer
/*       section (used only with shared files).  The GBD's themselves
/*       are in the section also and linked from a queue header in
/*       the Global Buffer Header (GBH).
/*
/*       ***  WARNING - THIS STRUCTURE MUST BE QUADWORD ALIGNED  ***
/*


aggregate GBDDEF structure fill prefix GBD$;
    FLINK longword unsigned;                          /* Forward link - Note: This is a self relative queue
    BLINK longword unsigned;                          /* Back link
    BID byte unsigned;                                /* Block ID
    constant BID        equals 19  prefix GBD tag $C;  /* Block ID code for GBD
    BLN byte unsigned;                                /* Block length of GBD
    FLAGS_OVERLAY union fill;
        FLAGS byte unsigned;                          /* Buffer status flags
        FLAGS_BITS structure fill;
            VALID bitfield mask;                      /* Buffer is valid.
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    CACHE_VAL byte unsigned;                          /* Cache value of this bucket
    VBN longword unsigned;                            /* VBN of bucket the buffer describes
    VBNSEQNUM longword unsigned;                      /* VBN sequence number validity check
    LOCK_ID longword unsigned;                        /* Lock ID of system lock.
    NUMB word unsigned;                               /* Number of bytes in use
    SIZE word unsigned;                               /* Size of buffer in bytes
    REL_ADDR longword unsigned;                       /* Address of buffer relative to GBH
    USECNT word unsigned;                             /* Accessor count for bucket
    REHIT_RD byte unsigned;                           /* Rehit by same process count.
    REHIT_LK byte unsigned;                           /* Rehit by same locker process.
    FILL_1 longword fill prefix GBDDEF tag $$;        /* SPARE to maintain QUAD alignment.
    constant BLN equals . prefix GBD$ tag K;          /* Length of Global Buffer Descriptor structure.
    constant BLN equals . prefix GBD$ tag C;          /* Length of Global Buffer Descriptor structure.
end GBDDEF;

end_module $GBDDEF;

module $BLBDEF;
```

```
/*
/*        BLB field definitions
/*
/*        Bucket Lock Block (BLB)
/*
/*        The BLB contains the argument list for the SYS$ENQ system service
/*        as well a pointer to the BDB it relates to and other status.
/*


aggregate BLBDEF structure fill prefix BLB$;
    FLNK longword unsigned;                                /* Link to next BLB
    BLNK longword unsigned;                                /* Back link
    BID byte unsigned;                                     /* Block ID
    constant BID        equals 16  prefix BLB tag $C;      /* BLB code
    BLN byte unsigned;                                     /* Block length
    BLBFLGS_OVERLAY union fill;
        BLBFLGS byte unsigned;                             /* Control flags for BLB
        BLBFLGS_BITS structure fill;
            LOCK bitfield mask;                            /* Corresponds to CSH$V_LOCK
            NOWAIT bitfield mask;                          /* Same as CSH$V_NOWAIT
            NOREAD bitfield mask;                          /* Same as CSH$V_NOREAD
            NOBUFFER bitfield mask;                        /* Same as CSH$V_NOBUFFER
            IOLOCK bitfield mask;                          /* Lock mode for read/write
            DFW bitfield mask;                             /* This is lock for deferred write buffer
            WRITEBACK bitfield mask;                       /* The associated buffer must be written back
        end BLBFLGS_BITS;
    end BLBFLGS_OVERLAY;
    MODEHELD byte unsigned;                                /* Mode of current lock held.
    BDB_ADDR longword unsigned;                            /* BDB for which this lock is held
    OWNER longword unsigned;                               /* Address of stream owning this lock
    VBN longword unsigned;                                 /* VBN of bucket lock (resource name)
    RESDSC longword unsigned dimension 2;                  /* Resource name descriptor
    LKSTS word unsigned;                                   /* Lock status word
    FILL_1 word fill prefix BLBDEF tag $$;                 /* reserved
    LOCK_ID longword unsigned;                             /* Lock ID
    VALBLK_OVERLAY union fill;
        VALBLK longword unsigned dimension 4;              /* Lock value block
        constant BLN equals . prefix BLB$ tag K;           /* Length of BLB
        constant BLN equals . prefix BLB$ tag C;           /* Length of BLB
        VALSEQNO longword unsigned;                        /* Sequence number part of value block
    end VALBLK_OVERLAY;
end BLBDEF;

end_module $BLBDEF;

module $RJBDEF;
```

```
/*
/*        RJB Definitions
/*
/*        RMS Journaling Block (RJB)
/*
/*        This block contains the necessary control information to keep
/*        track of the state of journaling on this file
/*


aggregate RJBDEF structure fill prefix RJB$;
    CHAN_OVERLAY union fill;
        CHAN quadword unsigned;                         /*Channel Block
        CHAN_FIELDS structure fill;
            RUCHAN word unsigned;                       /* channel for recovery unit journal
            BICHAN word unsigned;                       /* channel for before image journal
            AICHAN word unsigned;                       /* channel for after image journal
            ATCHAN word unsigned;                       /* channel for audit trail journal
        end CHAN_FIELDS;
    end CHAN_OVERLAY;
    BID byte unsigned;                                  /*Block Id
    constant BID        equals 22  prefix RJB tag $C;
    BLN byte unsigned;                                  /*Block Length
    FLAGS_OVERLAY union fill;
        FLAGS word unsigned;                            /*Flags word
        constant BLN equals . prefix RJB$ tag K;        /*Length of RJB
        constant BLN equals . prefix RJB$ tag C;        /*Length of RJB
        FLAGS_BITS structure fill;
            RU bitfield mask;                           /*Set to indicate RU channel open
            BI bitfield mask;                           /*Set to indicate BI channel open
            AI bitfield mask;                           /*Set to indicate AI channel open
            AT bitfield mask;                           /*Set to indicate AT channel open
            OPEN bitfield mask;                         /*Indicates $OPEN mapping entry written
        end FLAGS_BITS;

    end FLAGS_OVERLAY;
end RJBDEF;

end_module $RJBDEF;

module $MJBDEF;
```

```
/*
/*      MJB field definitions
/*
/*        Miscellaneous Journaling Buffer
/*
/*      The MJB is used for writing miscellaneous journal entries,
/*      for example, extend entries or audit-trail entries.
/*



aggregate MJBDEF structure fill prefix MJB$;
    FILL_1 longword dimension 2 fill prefix MJBDEF tag $$;/* spare
    BID byte unsigned;                                  /* block id
    constant BID        equals 24  prefix MJB tag $C;
    BLN byte unsigned;                                  /* block length in longwords
    FLAGS_OVERLAY union fill;
        FLAGS word unsigned;                            /* flags
        FLAGS_BITS structure fill;
            INIT bitfield mask;                         /* set if RJR overhead is initialized
            FORCE bitfield mask;                        /* set if RJR is to be written thru to journal
                                                        /* and not buffered by CJF (input to WRITE_MJB)
            FILE bitfield mask;                         /* set if file operation to journal
            SYNCH_SHARE bitfield mask;                  /* set if file lock can't be released during
                                                        /* STALL
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    JNL byte unsigned;                                  /* set to CJF$_"jnl type" as input to WRITE_MJB
    FILL_2 byte dimension 3 fill prefix MJBDEF tag $$;  /* spare

    DESC_OVERLAY union fill;
        DESC quadword unsigned;                         /* RJR descriptor used in $WRITEJNL service
        DESC_FIELDS structure fill;
            SIZE word unsigned;                         /* size of RJR to write
            FILL_3 byte dimension 2 fill prefix MJBDEF tag $$;
            POINTER longword unsigned;                  /* pointer to RJR
        end DESC_FIELDS;
    end DESC_OVERLAY;
    IOSB_OVERLAY union fill;
        IOSB quadword unsigned;                         /* IOSB to use in $WRITEJNL
        IOSB_FIELDS structure fill;
            FILL_4 byte dimension 8 fill prefix MJBDEF tag $$;
            RJR character length 0 tag T;               /* the journal record begins here
            constant BLN equals . prefix MJB$ tag K;
            constant BLN equals . prefix MJB$ tag C;
        end IOSB_FIELDS;
    end IOSB_OVERLAY;
end MJBDEF;

end_module $MJBDEF;
```

0313 AH-BT13A-SE DIGITAL EQUIPMENT CORPORATION
VAX/VMS V4.0 CONFIDENTIAL AND PROPRIETARY

RMSFILSTR
SDL

RMSMAC
REQ

RMSINTSTR
SDL

RMSUSR
SDL

NWADEF
MDL

RMSFWADEF
SDL

RMSSHR
SDL