```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSSS
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  SSSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSSS
```

D 9

FILEID**RMS

```
RRRRRRRR    MM      MM    SSSSSSSS
RRRRRRRR    MM      MM    SSSSSSSS
RR      RR  MMMM  MMMM  SS
RR      RR  MMMM  MMMM  SS
RR      RR  MM  MM  MM  SS
RR      RR  MM  MM  MM  SS
RRRRRRRR    MM      MM    SSSSSS
RRRRRRRR    MM      MM    SSSSSS
RR  RR      MM      MM          SS
RR  RR      MM      MM          SS
RR    RR    MM      MM          SS      ....
RR    RR    MM      MM          SS      ....
RR      RR  MM      MM  SSSSSSSS        ....
RR      RR  MM      MM  SSSSSSSS        ....

LL            SSSSSSSS  TTTTTTTTTT
LL            SSSSSSSS  TTTTTTTTTT
LL          SS              TT
LL          SS              TT
LL          SS              TT
LL          SS              TT
LL            SSSSSS        TT
LL            SSSSSS        TT
LL                  SS      TT
LL                  SS      TT
LL                  SS      TT
LL                  SS      TT
LLLLLLLLLL  SSSSSSSS        TT
LLLLLLLLLL  SSSSSSSS        TT
```

```
!++
!
! UTLDEF.B32 - UTILITY DEFINITION MACROS FOR BLISS PROCESSING
!              OF STARLET DEFINITION MACROS.
!       Version '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.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************
!
!++
!
! MODIFIED BY:
!
!       V02-001 REFORMAT        Maria del C. Nasr       01-Aug-1980
!
!--
!
! macros to extract offsets, field widths, etc., from field extraction macros
!

MACRO   $BYTEOFFSET (OFFSET, POSITION, WIDTH, SIGN) = OFFSET%;

MACRO   $BITPOSITION (OFFSET, POSITION, WIDTH, SIGN) = POSITION%;

MACRO   $FIELDWIDTH (OFFSET, POSITION, WIDTH, SIGN) = WIDTH%;

MACRO   $EXTENSION (OFFSET, POSITION, WIDTH, SIGN) = SIGN%;

MACRO   $FIELDMASK (OFFSET, POSITION, WIDTH, SIGN) =
        (1^(POSITION+WIDTH) - 1^POSITION)%;

!
! macro to generate equlst constructs
!
MACRO
```

```
$EQULST(P,G,I,S)[A]=
    %NAME(P,GET1ST_ A) =
        %IF NUL2ND_ A
        %THEN (I) + %COUNT*(S)  ! assumes i, s always generated by conversion program
        %ELSE GET2ND_ A
        %FI %,

GET1ST_(A,B)=
        A %,
GET2ND_(A,B)=
        B %,        ! known non-null
NUL2ND_(A,B)=
        %NULL(B) %;
```

```
!       $BEGIN RMSIDXMAC,V04-000
!
!       MACRO DEFINITIONS FOR RMS-32 INDEX FILE ORGANIZATION
!
!****************************************************************************
!*                                                                          *
!*  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.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************
!
!++
!
! FACILITY:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
!
! ABSTRACT:
!
!
! ENVIRONMENT:
!
!               VAX/VMS OPERATING SYSTEM
!
!--
!
! AUTHOR:  D. H. Gillespie      CREATION DATE:  17-MAR-1978
!          and W. Koenig
!
!
! MODIFIED BY:
!
!       V03-002 MCN0003         Maria del C. Nasr       15-Mar-1982
!               Use new general linkage for RM$BUG3.  Take out definition
!               for R_REC_SIZE, and R_VBN.
!
!       V03-001 MCN0002         Maria del C. Nasr       25-Mar-1982
!               Add macro definition to calculate key buffer address.
!
!       V02-003 CDS0001         C Saether               9-Dec-1981
!               Comment out references to CSH$M_READAHEAD flag in the
!               $CSHFLAGS macro.
!
```

```
 0    !      V02-002 MCN0001          Maria del C. Nasr        22-Apr-1981
 0    !              Add macro definition for determining record identifier size
 0    !
 0    !**
 0
 0
```

```
!
! Define macro for psect attributes
!
MACRO   PSECT_ATTR = EXECUTE,PIC,GLOBAL %;
!
! Define macro to extract size
!
MACRO   $BYTESIZE(OFFSET,POSITION,WIDTH,SIGN) = WIDTH / 8 %;

!
! Structure declarations used for system defined structrues to save typing
!
STRUCTURE
        BBLOCK [O, P, S, E; N] =
                [N]
                (BBLOCK+O)<P,S,E>,

        BBLOCKVECTOR [I, O, P, S, E; N, BS] =
                [N*BS]
                (BBLOCKVECTOR+(O+I*BS))<P,S,E>;

!
! ***** The following two macros violate the BLISS language definition
! ***** in that they make use of the value of SP while building the argument
! ***** list.  It is the opinion of the BLISS maintainers that this usage is safe
! ***** from planned future optimizations.
!
! Macro to call the change mode to kernel system service.
!
! Macro call format is "KERNEL_CALL( ROUTINE, ARG1, ARG2, ...).
!
MACRO
        KERNEL_CALL (R) =
                BEGIN
                EXTERNAL ROUTINE SYS$CMKRNL     : ADDRESSING_MODE (ABSOLUTE);
                BUILTIN SP;
                SYS$CMKRNL(4, .SP, %LENGTH-1
                        %IF %LENGTH GTR 1 %THEN, %REMAINING %FI)
                END%;

!
! macro to generate a string descriptor
!
MACRO
        DESCRIPTOR (STRING) =
                UPLIT (%CHARCOUNT (STRING), UPLIT BYTE (STRING))%;

!
! macro to return the number of actual parameters supplied to a routine call
!
MACRO
        ACTUALCOUNT =
                BEGIN
                BUILTIN AP;
                .(.AP)<0,8>
                END
                %;
```

```
        !
        ! macro to generate call to bug check routine
        !
        MACRO BUG_CHECK =
                ( LINKAGE L_JSB;
                  EXTERNAL ROUTINE
                      RM$BUG3 :  RL$JSB;
                  RM$BUG3 ()  ) %;

        !
        ! Macro used to determine record identifier size (byte or word) depending on
        ! prologue version of the file.
        !
        MACRO IRC$_ID(RECADR) =
                (IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
                THEN
                    .RECADR[IRC$B_ID]
                ELSE
                    .RECADR[IRC$W_ID]) % ;


        ! Macro used to determine address of key buffer wanted.  Parameter is
        ! the keybuffer number.
        !

        MACRO KEYBUF_ADDR(KBUFNO) =
                .IRAB[IRB$L_KEYBUF] + .IFAB[IFB$W_KBUFSZ] * ((KBUFNO) - 1) % ;


```

```
!KEEP THESE DEFINITIONS IN ALPHABETICAL ORDER, PLEASE!

!
! internal register definitions
!
!       common register definitions
!
MACRO
        COMMON_FABREG =
                R_FAB, R_IFAB, R_IFAB_FILE, R_IMPURE %,

        COMMON_IOREG =
                R_BDB, R_BKT_ADDR %,

        COMMON_RABREG =
                R_RAB, R_IRAB, R_IFAB, R_IMPURE%,

        COMMON_FAB_STR =
                R_FAB_STR,
                R_IFAB_STR,
                R_IFAB_FILE_STR,
                R_IMPURE_STR %,

        COMMON_IO_STR =
                R_BDB_STR,
                R_BKT_ADDR_STR %,

        COMMON_RAB_STR =
                R_RAB_STR,
                R_IRAB_STR,
                R_IFAB_STR,
                R_IMPURE_STR%;


!       miscellaneous register definitions
!       the macros associated with registers will follow these conventions:
!               macro r_name =
!                       name = register_number %; (no trailing punctuation)
!               to be used basically in a linkage statement, and
!               macro r_name_str =
!                       r_name : ref bblock %;  (or whatever structure)
!               to be used basically in external or global register declarations
MACRO
        R_BDB =
                BDB      = 4 %,
        R_BKT_ADDR =
                BKT_ADDR = 5 %,
        R_FAB   =
                FAB      = 8 %,
        R_IDX_DFN =
                IDX_DFN = 7 %,
        R_IFAB_FILE =
                IFAB_FILE = 9 %,
        R_IMPURE =
                IMPURE   = 11 %,
        R_IRAB =
```

```
                IRAB     = 9 %,
        R_RAB =
                RAB      = 8 %,
        R_REC_ADDR =
                REC_ADDR = 6 %,
        R_IFAB =
                IFAB     = 10 %,


        R_BDB_STR =
                R_BDB    : REF BBLOCK %,
        R_BKT_ADDR_STR =
                R_BKT_ADDR : REF BBLOCK %,
        R_FAB_STR =
                R_FAB    : REF BBLOCK %,
        R_ID_STR =
                R_ID     : BYTE %,
        R_IDX_DFN_STR =
                R_IDX_DFN : REF BBLOCK %,
        R_IFAB_STR =
                R_IFAB   : REF BBLOCK %,
        R_IMPURE_STR =
                R_IMPURE : REF BBLOCK %,
        R_IRAB_STR =
                R_IRAB   : REF BBLOCK %,
        R_RAB_STR =
                R_RAB    : REF BBLOCK %,
        R_REC_ADDR_STR =
                R_REC_ADDR : REF BBLOCK %,
        R_IFAB_FILE_STR =
                R_IFAB_FILE : REF BBLOCK %,
        R_VBN_STR =
                R_VBN           %;
```

```
!
!       macro to make the status codes small
!
MACRO WORDMASK(CODE) = (
        (CODE AND %X'FFFF'))
        %;
MACRO RMSERR (CODE) = (
        %NAME('RMS$_',CODE) AND %X'FFFF')
        %,

      RMSSUC (CODE) = (
        %IF %LENGTH EQL 0 %THEN
                1
        %ELSE
                %NAME('RMS$_',CODE) AND %X'FFFF'
        %FI)
        %;
!
! macro to make constants that are calculated have the <0,16> attribute
!
!
!       macros to make the code a little nicer
!
MACRO
!
! this macro allows you to make a call to another routine
!  (and do whatever you want in a block before the call),
! and if an error resulted, do whatever you want and
!   then return with the status.
!
    RETURN_ON_ERROR (CALL) [] =
        (LOCAL STATUS;
         IF NOT (STATUS = (CALL)) THEN
                                        (%REMAINING;
                                         RETURN .STATUS)) %,

!
! this macro is the same as the one above, except that it
!  it returns w/ status whether or not there was an error
!  and the caller can supply an 'else' block
!   note:  the 'call' part and 'else' part must be separated by a comma
!    not a semicolon and the 'else' part must be terminated by a semicolon
!
    RETURN_ELSE (CALL) [] =
        (LOCAL STATUS;
         IF NOT (STATUS = (CALL)) THEN RETURN .STATUS
         ELSE
                %REMAINING
                RETURN .STATUS)
    %;
```

```
! this is an internal cache macro to put the value of the flags into cshtmp
MACRO IRP(A)[] =
        %ASSIGN (CSHTMP,CSHTMP OR %NAME('CSH$M_',A));
        IRP(%REMAINING);
        %;
!
! this is an internal cache macro to verify the cache flags and set them up
MACRO $CSHFLAGS(FLAGS) =
        COMPILETIME CSHTMP = 0;
        %IF NOT %NULL(FLAGS) %THEN
                IRP (%REMOVE(FLAGS));
        %FI;
!       %IF (CSHTMP AND CSH$M_READAHEAD) NEQ 0 %THEN
!               %ASSIGN (CSHTMP,CSHTMP OR CSH$M_NOWAIT);
!               %IF (CSHTMP AND
!                   (CSH$M_LOCK OR CSH$M_NOREAD OR CSH$M_NOBUFFER))
!                       NEQ 0 %THEN
!                               %ERRORMACRO ('INVALID CACHE FLAG COMBINATION');
!               %FI;
!       %FI;
        %IF (CSHTMP AND CSH$M_NOBUFFER)  NEQ 0 %THEN
                %ASSIGN (CSHTMP,CSHTMP OR CSH$M_NOREAD);
        %FI;
        %;
!
! this is a macro to call cache or cachec
!
MACRO CACHE (VBN,SIZE,FLAGS,EP) =
    BEGIN
        %IF %NULL(FLAGS) %THEN
                COMPILETIME CSHTMP = 0
        %ELSE
        $CSHFLAGS(FLAGS)
        %FI;
        %IF %NULL(EP) %THEN
                RM$CACHE(VBN,SIZE,CSHTMP)
        %ELSE
                %NAME('RM$CACHE',EP)(VBN,SIZE,CSHTMP)
        %FI
    END
        %;
!
! this is a macro to call getbkt or getbktc
!
MACRO GETBKT (VBN,SIZE,FLAGS,EP) =
    BEGIN
        %IF %NULL(FLAGS) %THEN
                COMPILETIME CSHTMP = 0
        %ELSE
        $CSHFLAGS(FLAGS)
        %FI;
        %IF %NULL(EP) %THEN
                RM$GETBKT(VBN,SIZE,CSHTMP)
        %ELSE
                %NAME('RM$GETBKT',EP)(VBN,SIZE,CSHTMP)
        %FI
    END
```

```
;     0418   0           %;
;     0419   0
;     0420   0
;     0421   0
```

```
!
! these are macros to do probing of user structures
!
MACRO
    IFNORD (SIZE,ADDR,MODE) [] =
    (
    IF NOT PROBER(
        %IF %NULL(MODE) %THEN 0 %ELSE MODE %FI,
        SIZE,
        ADDR)
    THEN
        %REMAINING)
    %,

    IFNOWRT (SIZE,ADDR,MODE) [] =
    (
    IF NOT PROBEW(
        %IF %NULL(MODE) %THEN 0 %ELSE MODE %FI,
        SIZE,
        ADDR)
    THEN
        %REMAINING)
    %,

    IFRD (SIZE,ADDR,MODE) [] =
    (
    IF PROBER(
        %IF %NULL(MODE) %THEN 0 %ELSE MODE %FI,
        SIZE,
        ADDR)
    THEN
        %REMAINING)
    %,
! macros to do long probes
    READ_LONG(SIZE,ADDR,MODE) = NOT RM$NOREAD_LONG(SIZE,ADDR,MODE) %,

    WRT_LONG(SIZE,ADDR,MODE) = NOT RM$NOWRT_LONG(SIZE,ADDR,MODE) %;
```

```
!
! macro to release a bucket and clear the location where its bdb addr is stored
!
MACRO RELEASE(B) =
        BEGIN
        BDB = .B;
        B = 0;
        RM$RLSBKT(0);
        END%;
!
! macro to make sure that an assumption made about the position of symbols
! in a structure is still valid
! the arguments to this macro must be preceded by %quote
! e.g., assume (%quote ifb$b_bid, %quote ifb$b_bln);
!
 MACRO ASSUME (A,B) =
        %IF $BYTEOFFSET(A) + $BYTESIZE(A) NEQ $BYTEOFFSET(B)
        %THEN %WARN('WARNING CONSTANT HAS CHANGED')
        %FI %;
!
! this version of assume is good for constants
! e.g. assume (irc$c_fixovhdsz + 2, irc$c_varovhdsz);
!
 MACRO ASSUME_C (A,B) =
        %IF $BYTEOFFSET(A)  NEQ $BYTEOFFSET(B)
        %THEN %WARN('WARNING CONSTANT HAS CHANGED')
        %FI %;
!       [ 2 0 1 , 1 0 ] R M S I D X L N K . R 3 2
!
!       Define subroutine linkage
!
!
!****************************************************************************
!*                                                                          *
!*  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.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************
!
!++
!
```

```
0       ! FACILITY:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
0       !
0       ! ABSTRACT:
0       !               This module defines all the routine linkage
0       !
0       !
0       ! ENVIRONMENT:
0       !
0       !               VAX/VMS OPERATING SYSTEM
0       !
0       !--
0       !
0       ! AUTHOR:  D. H. Gillespie      CREATION DATE:  17-MAR-1978
0       !                and W. Koenig
0       !
0       ! MODIFIED BY:
0       !
0       !       V03-024 RAS0154         Ron Schaefer            2-May-1983
0       !               Add NOPRESERVE (R2) to L_EXTENDO linkage.
0       !
0       !       V03-023 MCN0020         Maria del C. Nasr       07-Apr-1983
0       !               Eliminage linkages of RM$NULLKEY,and RM$COMPRESS_KEY.
0       !               They will be using general linkages.  Modify L_ALLOC3,
0       !               and L_EXTENDO to use parameters instead of global registers.
0       !
0       !       V03-022 MCN0019         Maria del C. Nasr       05-Apr-1983
0       !               Preserve all registers except R0 and R1 in linkage
0       !               FABREG.  RM$XSUMO requires a separate linkage because
0       !               it cannot preserve R4.
0       !
0       !       V03-021 TMK0001         Todd M. Katz            26-Mar-1983
0       !               Add the linkage RABREG_4.
0       !
0       !       V03-020 MCN0018         Maria del C. Nasr       24-Mar-1983
0       !               Define new general linkages.  Also, since the linkages
0       !               have changed so much, eliminate all history comments.
0       !
0       !*****************************************************************
```

```
        !
        !       This module defines all the routine linkage for RMS-32 index file
        !       organization.
        !
        !       KEEP THESE DEFINITIONS IN ALPHABETICAL ORDER PLEASE
        !
        !       The following conventions will be used for linkage macros:
        !
        !               MACRO L_NAME =
        !                       RL$NAME =
        !                       JSB (REGISTERS) :
        !                       GLOBAL (REGISTER DEFINITIONS) %;
        !
        !               The register definitions are macros of the forms
        !                       COMMON_FABREG, COMMON_RABREG, COMMON_IOREG, etc.
        !                       or R_REGNAME as described in RMSIDXMAC.R32
        !
MACRO

        L_ALDBUF =
                RL$ALDBUF =
                JSB (REGISTER = 5) :
                GLOBAL (R_IMPURE, R_IFAB)
                NOPRESERVE (2,3,4)
                NOTUSED (8,9) %,

        L_ALLOC3 =
                RL$ALLOC3 =
                JSB (REGISTER = 7; REGISTER = 1, REGISTER = 2) :
                GLOBAL (R_IFAB) %,

        L_BDBALLOC =
                RL$BDBALLOC =
                JSB (REGISTER = 4, REGISTER = 5) :
                GLOBAL (COMMON_RABREG)
                NOPRESERVE (2,3,4,5,6) %,

        L_CACHE =
                RL$CACHE =
                JSB (REGISTER = 1, REGISTER = 2, REGISTER = 3) :
                GLOBAL (COMMON_IOREG)
                NOPRESERVE (1,2,3)
                NOTUSED (8,9,10,11) %,

        L_CHECK_SEGMENT=
                RL$CHECK_SEGMENT =
                JSB (REGISTER = 0, REGISTER = 4, REGISTER = 2) :
                GLOBAL (R_IDX_DFN)
                NOPRESERVE (2,4,5)
                PRESERVE (1) %,

        L_CHKSUM =
                RL$CHKSUM =
                JSB (REGISTER = 5) :
                NOPRESERVE (0,1,2) %,

        L_COMPARE_KEY =
```

```
                RL$COMPARE_KEY =
                JSB (REGISTER = 1, REGISTER = 3, REGISTER = 0) :
                GLOBAL (R_IDX_DFN)
                NOPRESERVE (3) %,

        L_ERROR_LINK1 =
                RL$ERROR_LINK1 =
                JSB () :
                GLOBAL (COMMON_RABREG)
                PRESERVE (0) %,

        L_ERROR_LINK2 =
                RL$ERROR_LINK2 =
                JSB () :
                GLOBAL (COMMON_RABREG, R_IDX_DFN)
                PRESERVE (0) %,

        L_EXTEND0 =
                RL$EXTEND0 =
                JSB (REGISTER = 5, REGISTER = 6; REGISTER = 1, REGISTER = 6) :
                GLOBAL (COMMON_FABREG)
                NOPRESERVE (2,3,4,5) %,

        L_FABREG =
                RL$FABREG =
                JSB () :
                GLOBAL (COMMON_FABREG)
                NOPRESERVE (0,1) %,

        L_FABREG_7 =
                RL$FABREG_7 =
                JSB () :
                GLOBAL (COMMON_FABREG, R_IDX_DFN) %,

        L_GETSPC =
                RL$GETSPC =
                JSB (REGISTER = 1, REGISTER = 2; REGISTER = 1) :
                GLOBAL (R_IMPURE)
                NOPRESERVE (2,3,4)
                NOTUSED (8,9,10) %,

        L_JSB =
                RL$JSB =
                JSB () %,

        L_JSB01 =
                RL$JSB01 =
                JSB (REGISTER = 0, REGISTER = 1) :
                GLOBAL (R_BKT_ADDR, R_REC_ADDR, R_IDX_DFN, R_IRAB, R_IFAB)
                NOPRESERVE (0,1) %,

        L_LINK_7_10_11 =
                RL$LINK_7_10_11 =
                JSB () :
                GLOBAL (R_IDX_DFN, R_IFAB, R_IMPURE)
                NOPRESERVE (0,1) %,
```

```
L_PRESERVE1 =
        RL$PRESERVE1 =
        JSB () :
        GLOBAL (COMMON_RABREG, R_BDB, R_REC_ADDR, R_IDX_DFN)
        PRESERVE (1) %,

L_QUERY_AND_LOCK =
        RL$QUERY_AND_LOCK =
        JSB (REGISTER = 1, REGISTER = 2) :
        GLOBAL (COMMON_RABREG)
        NOPRESERVE (3) %,

L_RABREG =
        RL$RABREG =
        JSB () :
        GLOBAL (COMMON_RABREG)
        NOPRESERVE (0,T) %,

L_RABREG_4 =
        RL$RABREG_4 =
        JSB () :
        GLOBAL (COMMON_RABREG, R_BDB)
        NOPRESERVE (0,T) %,

L_RABREG_4567 =
        RL$RABREG_4567 =
        JSB () :
        GLOBAL (COMMON_RABREG, COMMON_IOREG, R_REC_ADDR, R_IDX_DFN)
        NOPRESERVE (0,T) %,

L_RABREG_457 =
        RL$RABREG_457 =
        JSB () :
        GLOBAL (COMMON_RABREG, COMMON_IOREG, R_IDX_DFN)
        NOPRESERVE (0,T) %,

L_RABREG_467 =
        RL$RABREG_467 =
        JSB () :
        GLOBAL (COMMON_RABREG, R_BDB, R_REC_ADDR, R_IDX_DFN)
        NOPRESERVE (0,T) %,

L_RABREG_567 =
        RL$RABREG_567 =
        JSB () :
        GLOBAL (COMMON_RABREG, R_BKT_ADDR, R_REC_ADDR, R_IDX_DFN)
        NOPRESERVE (0,T) %,

L_RABREG_67 =
        RL$RABREG_67 =
        JSB () :
        GLOBAL (COMMON_RABREG, R_REC_ADDR, R_IDX_DFN)
        NOPRESERVE (0,T) %,

L_RABREG_7 =
        RL$RABREG_7 =
        JSB () :
```

```
                GLOBAL (COMMON_RABREG, R_IDX_DFN)
                NOPRESERVE (0,T) %,

        L_REC_OVHD =
                RL$REC_OVHD =
                JSB (REGISTER = 1; REGISTER = 1) :
                GLOBAL (R_REC_ADDR, R_IDX_DFN, R_IFAB) %,

        L_RELEASE =
                RL$RELEASE =
                JSB (REGISTER = 3) :
                GLOBAL (R_BDB, R_IRAB, R_IFAB, R_IMPURE)
                NOPRESERVE (1,2)
                NOTUSED (8) %,

        L_RELEASE_FAB =
                RL$RELEASE_FAB =
                JSB (REGISTER = 3) :
                GLOBAL (R_BDB, R_IFAB, R_IFAB_FILE, R_IMPURE)
                NOPRESERVE (1,2)
                NOTUSED(8) %,

        L_RETSPC =
                RL$RETSPC =
                JSB (REGISTER= 2, REGISTER = 3, REGISTER = 4) :
                GLOBAL (R_IMPURE)
                NOPRESERVE (2,3,5)
                NOTUSED (8,9,10) %,

        L_SIDR_FIRST =
                RL$SIDR_FIRST =
                JSB (STANDARD; REGISTER = 1, REGISTER = 2) :
                GLOBAL (R_REC_ADDR, R_IDX_DFN, COMMON_RABREG) %,

        L_XSUMO =
                RL$XSUMO =
                JSB () :
                GLOBAL (COMMON_FABREG)
                NOPRESERVE (0,T,4) %;


!
! VERSION:      '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.                                                            *
!*                                                                          *
```

```
    0   !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
    0   !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
    0   !*  CORPORATION.                                                            *
    0   !*                                                                          *
    0   !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
    0   !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
    0   !*                                                                          *
    0   !*                                                                          *
    0   !****************************************************************************
    0   !
    0   !++
    0   ! FACILITY:
    0   !
    0   !       MESSAGES ARE FOR THE FAL (DECNET FILE ACCESS LISTENER) FACILITY, BUT
    0   !       ARE GENERATED ON BEHALF OF FAL BY RMS.
    0   !
    0   ! ABSTRACT:
    0   !
    0   !       THIS MODULE DEFINES FAL STATUS CODE SYMBOLS AND CORRESPONDING MESSAGE
    0   !       TEXT. NOTE THAT THIS MESSAGE FILE IS MAINTAINED BY THE RMS FACILITY,
    0   !       NOT THE FAL FACILITY, BECAUSE RMS DOES THE TRANSLATION OF DAP STATUS
    0   !       INTO FAL STATUS CODES WHICH ARE REPORTED AS SECONDARY STATUS INFORMATION
    0   !       TO THE RMS$_NETFAIL AND RMS$_SUPPORT COMPLETION CODES ON BEHALF OF FAL.
    0   !
    0   ! ENVIRONMENT: VAX/VMS
    0   !
    0   ! AUTHOR: JAMES A. KRYCKA,      CREATION DATE:  15-JAN-1982
    0   !
    0   ! MODIFIED BY:
    0   !
    0   !       V03-001 JAK0147          J A KRYCKA       09-JUL-1984
    0   !               ADD ERROR CODES THAT CORRESPOND TO DAP MICCODES 416 TO 470
    0   !               (OCTAL) FOR MACCODES 4 THROUGH 7 DEFINED IN THE DAP V7.0
    0   !               SPECIFICATION.
    0   !
    0   !--
    0   !++
    0   ! DEFINE FAL STATUS CODES FOR USE BY RMS AS SECONDARY STATUS CODES.
    0   !
    0   ! LAYOUT OF MESSAGE SPACE:
    0   !
    0   !       MSG NUMBERS      MSG NUMBERS      MSG CODES        DESCRIPTION
    0   !       (DECIMAL)        (HEXADECIMAL)    (HEXADECIMAL)
    0   !
    0   !          0 - 1023      0000 - 03FF      8000 - 9FFF      UNUSED
    0   !       1024 - 2047      0400 - 07FF      A000 - BFFF      FOR RMS$_NETFAIL
    0   !       2048 - 4095      0800 - 0FFF      C000 - FFFF      FOR RMS$_SUPPORT
    0   !--
    0
    0   !...$RMSFALMSG
    0
    0   LITERAL
P   0   $EQULST (FAL$_,GBL,0,1
P   0       ,(FACIL[ITY,                  503)
P   0   !+
P   0   ! DEFINE FAL STATUS CODES THAT ARE ASSOCIATED WITH THE RMS$_NETFAIL COMPLETION
P   0   ! CODE AND RETURNED IN THE STV FIELD OF THE FAB OR RAB.
```

```
!
! USE THE FOLLOWING FORMULA TO MAP A DAP STATUS CODE (STSCODE FIELD WITH
! MACCODE = 4 THRU 7) INTO A CORRESPONDING FAL MESSAGE CODE (LOWER 16 BITS):
!
!       MESSAGE-CODE = (2**15 + ((MICCODE + 1024) * 8) + SEVERITY-LEVEL)
!
! NOTE:         THE FIRST THREE CHARACTERS OF THE MESSAGE IDENTIFICATION MNEMONIC
!       FOR EACH MESSAGE CODE IN THIS SET INDICATES ITS ORIGIN AS FOLLOWS:
!
!       RMS --- THE CODE IS DEFINED FOR RMS-11 OR RMS-20, BUT NOT FOR RMS-32.
!               (RMS-32 COMPLETION CODES ARE DEFINED IN RMSDEF.MDL.)
!       FCS --- THE CODE IS DEFINED FOR THE FCS-11 ENVIRONMENT.
!       TEN --- THE CODE IS DEFINED FOR THE TOPS-10 ENVIRONMENT.
!       DAP --- ORIGIN IN THE DAP SPECIFICATION IS UNKNOWN.
!-
        ,(OFFSET_A,             1024)
! MICCODE = 0000 (OCTAL)
        ,(DAPFAIL,              (%X'01F7A004'))
        ,(RMSABO,               (%X'01F7A00C'))
! MICCODE = 0011 (OCTAL)
        ,(RMSAST,               (%X'01F7A04C'))
        ,(RMSBPA,               (%X'01F7A054'))
        ,(RMSBPS,               (%X'01F7A05C'))
! MICCODE = 0026 (OCTAL)
        ,(RMSCLS,               (%X'01F7A0B4'))
! MICCODE = 0056 (OCTAL)
        ,(RMSFID,               (%X'01F7A174'))
! MICCODE = 0071 (OCTAL)
        ,(RMSINI,               (%X'01F7A1CC'))
! MICCODE = 0102 (OCTAL)
        ,(RMSLBL,               (%X'01F7A214'))
        ,(RMSLBY,               (%X'01F7A21C'))
        ,(RMSLCH,               (%X'01F7A224'))
! MICCODE = 0106 (OCTAL)
        ,(RMSLOC,               (%X'01F7A234'))
        ,(RMSMAP,               (%X'01F7A23C'))
! MICCODE = 0115 (OCTAL)
        ,(RMSNID,               (%X'01F7A26C'))
! MICCODE = 0117 (OCTAL)
        ,(RMSOPN,               (%X'01F7A27C'))
! MICCIDE = 0124 (OCTAL)
        ,(RMSPRM,               (%X'01F7A2A4'))
! MICCODE = 0153 (OCTAL)
        ,(RMSSTK,               (%X'01F7A35C'))
! MICCODE = 0162 (OCTAL)
        ,(RMSVOL,               (%X'01F7A394'))
! MICCODE = 0170 (OCTAL)
        ,(RMSCAA,               (%X'01F7A3C4'))
! MICCODE = 0231 (OCTAL)
        ,(RMSBLK,               (%X'01F7A4CC'))
        ,(RMSBSZ,               (%X'01F7A4D4'))
        ,(RMSCDR,               (%X'01F7A4DC'))
        ,(RMSCGJ,               (%X'01F7A4E4'))
        ,(RMSCOF,               (%X'01F7A4EC'))
        ,(RMSJFN,               (%X'01F7A4F4'))
        ,(RMSPEF,               (%X'01F7A4FC'))
        ,(RMSTRU,               (%X'01F7A504'))
```

```
; P 0               ,(RMSUDF,          (%X'01F7A50C'))
; P 0               ,(RMSXCL,          (%X'01F7A514'))
; P 0               ,(DAPDIRFUL,       (%X'01F7A51C'))
; P 0               ,(FCSHWR,          (%X'01F7A524'))
; P 0               ,(FCSFHE,          (%X'01F7A52C'))
; P 0               ,(DAPWRTEOF,       (%X'01F7A534'))
; P 0               ,(FCSONP,          (%X'01F7A53C'))
; P 0               ,(FCSDNA,          (%X'01F7A544'))
; P 0               ,(FCSDAA,          (%X'01F7A54C'))
; P 0               ,(FCSDUN,          (%X'01F7A554'))
; P 0               ,(FCSRSU,          (%X'01F7A55C'))
; P 0               ,(FCSOVR,          (%X'01F7A564'))
; P 0               ,(FCSBCC,          (%X'01F7A56C'))
; P 0               ,(FCSNOD,          (%X'01F7A574'))
; P 0               ,(FCSIFU,          (%X'01F7A57C'))
; P 0               ,(FCSHFU,          (%X'01F7A584'))
; P 0               ,(FCSWAC,          (%X'01F7A58C'))
; P 0               ,(FCSCKS,          (%X'01F7A594'))
; P 0               ,(FCSWAT,          (%X'01F7A59C'))
; P 0               ,(FCSALN,          (%X'01F7A5A4'))
; P 0               ,(FCSBTF,          (%X'01F7A5AC'))
; P 0               ,(FCSILL,          (%X'01F7A5B4'))
; P 0               ,(FCS2DV,          (%X'01F7A5BC'))
; P 0               ,(FCSFEX,          (%X'01F7A5C4'))
; P 0               ,(FCSRNM,          (%X'01F7A5CC'))
; P 0               ,(FCSFOP,          (%X'01F7A5D4'))
; P 0               ,(FCSVER,          (%X'01F7A5DC'))
; P 0               ,(FCSEOV,          (%X'01F7A5E4'))
; P 0               ,(FCSDAO,          (%X'01F7A5EC'))
; P 0               ,(FCSBBE,          (%X'01F7A5F4'))
; P 0               ,(FCSEOT,          (%X'01F7A5FC'))
; P 0               ,(FCSNBF,          (%X'01F7A604'))
; P 0               ,(FCSNBK,          (%X'01F7A60C'))
; P 0               ,(FCSNST,          (%X'01F7A614'))
; P 0               ,(FCSULK,          (%X'01F7A61C'))
; P 0               ,(FCSNLN,          (%X'01F7A624'))
; P 0               ,(FCSSRE,          (%X'01F7A62C'))
; P 0       ! MICCODE = 0311 (OCTAL)
; P 0               ,(DAPQUOEXC,       (%X'01F7A64C'))
; P 0       ! MICCODE = 0346 (OCTAL)
; P 0               ,(DAPDIRCAF,       (%X'01F7A734'))
; P 0               ,(DAPDIRCRA,       (%X'01F7A73C'))
; P 0               ,(DAPDIRPRO,       (%X'01F7A744'))
; P 0               ,(DAPDIRPRA,       (%X'01F7A74C'))
; P 0               ,(DAPDIRNFA,       (%X'01F7A754'))
; P 0               ,(DAPDIRCON,       (%X'01F7A75C'))
; P 0       ! MICCODE = 0416 (OCTAL)
; P 0               ,(TENFILMOD,       (%X'01F7A874'))
; P 0               ,(TENDEVNA,        (%X'01F7A87C'))
; P 0               ,(TENDEVNF,        (%X'01F7A884'))
; P 0               ,(TENPARALL,       (%X'01F7A88C'))
; P 0               ,(TENBNFREE,       (%X'01F7A894'))
; P 0               ,(TENCSDF,         (%X'01F7A89C'))
; P 0               ,(TENCDDF,         (%X'01F7A8A4'))
; P 0               ,(TENSFDNF,        (%X'01F7A8AC'))
; P 0               ,(TENSLE,          (%X'01F7A8B4'))
; P 0               ,(TENSFDCNT,       (%X'01F7A8BC'))
```

```
		,(TENNCESL,		(%X'01F7A8C4'))
		,(TENCUPFIL,		(%X'01F7A8CC'))
		,(TENNETCAP,		(%X'01F7A8D4'))
		,(TENTSKNA,		(%X'01F7A8DC'))
		,(TENNODNF,		(%X'01F7A8E4'))
		,(TENSFDREN,		(%X'01F7A8EC'))
		,(TENCDFNDR,		(%X'01F7A8F4'))
		,(TENJCREAD,		(%X'01F7A8FC'))
		,(TENRENSFD,		(%X'01F7A904'))
		,(TENDEVDWN,		(%X'01F7A90C'))
		,(TENDEVRES,		(%X'01F7A914'))
		,(TENDEVMDA,		(%X'01F7A91C'))
		,(TENDEVALL,		(%X'01F7A924'))
		,(TENILLDM,		(%X'01F7A92C'))
		,(TENLPTPAG,		(%X'01F7A934'))
		,(TENLPTVFU,		(%X'01F7A93C'))
		,(TENLPTCHR,		(%X'01F7A944'))
		,(TENLPTRAM,		(%X'01F7A94C'))
		,(TENFILSPC,		(%X'01F7A954'))
	! MICCODE = 0453 (OCTAL)
		,(TENNSNOD,		(%X'01F7A95C'))
		,(TENANODI,		(%X'01F7A964'))
		,(TENSNODI,		(%X'01F7A96C'))
		,(TENNSDEV,		(%X'01F7A974'))
		,(TENADEVI,		(%X'01F7A97C'))
		,(TENSDEVI,		(%X'01F7A984'))
		,(TENNSDIR,		(%X'01F7A98C'))
		,(TENADIRI,		(%X'01F7A994'))
		,(TENSDIRI,		(%X'01F7A99C'))
		,(TENNSFIL,		(%X'01F7A9A4'))
		,(TENAFILI,		(%X'01F7A9AC'))
		,(TENSFILI,		(%X'01F7A9B4'))
		,(TENAFILR,		(%X'01F7A9BC'))
		,(TENSFILR,		(%X'01F7A9C4'))
	!+
	! DEFINE FAL STATUS CODES THAT ARE ASSOCIATED WITH THE RMS$_SUPPORT COMPLETION
	! CODE AND RETURNED IN THE STV FIELD OF THE FAB OR RAB.
	!
	! USE THE FOLLOWING FORMULA TO MAP A DAP STATUS CODE (STSCODE FIELD WITH
	! MACCODE = 2) INTO A CORRESPONDING FAL MESSAGE CODE (LOWER 16 BITS):
	!
	!	MESSAGE-CODE = (2**15 + ((MICCODE + 2048) * 8) + SEVERITY-LEVEL)
	!
	! NOTE:		THE INTENT IN THIS SECTION IS TO DEFINE FAL STATUS CODES ONLY FOR DAP
	!	FIELDS THAT CORRESPOND DIRECTLY TO RMS CONTROL BLOCK FIELDS USED FOR
	!	INPUT. FOR OTHER DAP FIELDS THAT MAY BE REJECTED BY THE REMOTE FAL AS
	!	BEING UNSUPPORTED, THE RMS$_SUP COMPLETION CODE IS RETURNED IN THE STS
	!	FIELD OF THE FAB OR RAB WITH AN ASSOCIATED DAP CODE IN THE STV FIELD.
	!-
		,(OFFSET_B,		2048)
	! MICCODE = 0222 (OCTAL)
		,(ORG,			(%X'01F7C494'))
		,(RFM,			(%X'01F7C49C'))
		,(RAT,			(%X'01F7C4A4'))
		,(BLS,			(%X'01F7C4AC'))
		,(MRS,			(%X'01F7C4B4'))
		,(ALQ1,			(%X'01F7C4BC'))
```

```
; P 0                ,(BKS,              (%X'01F7C4C4'))
; P 0                ,(FSZ,              (%X'01F7C4CC'))
; P 0                ,(MRN,              (%X'01F7C4D4'))
; P 0        ! MICCODE = 0234 (OCTAL)
; P 0                ,(DEQ1,             (%X'01F7C4E4'))
; P 0                ,(FOP1,             (%X'01F7C4EC'))
; P 0        ! MICCODE = 0241 (OCTAL)
; P 0                ,(LRL,              (%X'01F7C50C'))
; P 0        ! MICCODE = 0320 (OCTAL)
; P 0                ,(ACCFUNC,          (%X'01F7C684'))
; P 0        ! MICCODE = 0323 (OCTAL)
; P 0                ,(FAC,              (%X'01F7C69C'))
; P 0                ,(SHR,              (%X'01F7C6A4'))
; P 0        ! MICCODE = 0420 (OCTAL)
; P 0                ,(CTLFUNC,          (%X'01F7C884'))
; P 0        ! MICCODE = 0422 (OCTAL)
; P 0                ,(RAC,              (%X'01F7C894'))
; P 0                ,(KEY,              (%X'01F7C89C'))
; P 0                ,(KRF,              (%X'01F7C8A4'))
; P 0                ,(ROP,              (%X'01F7C8AC'))
; P 0        ! MICCODE = 0520 (OCTAL)
; P 0                ,(CONFUNC,          (%X'01F7CA84'))
; P 0        ! MICCODE = 0720 (OCTAL)
; P 0                ,(CMPFUNC,          (%X'01F7CE84'))
; P 0                ,(FOP2,             (%X'01F7CE8C'))
; P 0        ! MICCODE = 1221 (OCTAL)
; P 0                ,(FLG,              (%X'01F7D48C'))
; P 0                ,(DFL,              (%X'01F7D494'))
; P 0                ,(IFL,              (%X'01F7D49C'))
; P 0        ! MICCODE = 1225 (OCTAL)
; P 0                ,(POS,              (%X'01F7D4AC'))
; P 0                ,(SIZ,              (%X'01F7D4B4'))
; P 0                ,(REF,              (%X'01F7D4BC'))
; P 0                ,(KNM,              (%X'01F7D4C4'))
; P 0                ,(NUL,              (%X'01F7D4CC'))
; P 0                ,(IAN,              (%X'01F7D4D4'))
; P 0                ,(LAN,              (%X'01F7D4DC'))
; P 0                ,(DAN,              (%X'01F7D4E4'))
; P 0                ,(DTP,              (%X'01F7D4EC'))
; P 0        ! MICCODE = 1321 (OCTAL)
; P 0                ,(VOL,              (%X'01F7D68C'))
; P 0                ,(ALN,              (%X'01F7D694'))
; P 0                ,(AOP,              (%X'01F7D69C'))
; P 0                ,(LOC,              (%X'01F7D6A4'))
; P 0        ! MICCODE = 1326 (OCTAL)
; P 0                ,(ALQ2,             (%X'01F7D6B4'))
; P 0                ,(AID,              (%X'01F7D6BC'))
; P 0                ,(BKZ,              (%X'01F7D6C4'))
; P 0                ,(DEQ2,             (%X'01F7D6CC'))
; P 0        ! MICCODE = 1521 (OCTAL)
; P 0                ,(CDT,              (%X'01F7DA8C'))
; P 0                ,(RDT,              (%X'01F7DA94'))
; P 0                ,(EDT,              (%X'01F7DA9C'))
; P 0                ,(RVN,              (%X'01F7DAA4'))
; P 0        ! MICCODE = 1621 (OCTAL)
; P 0                ,(OWNER,            (%X'01F7DC8C'))
; P 0                ,(PROTSYS,          (%X'01F7DC94'))
```

```
              ,(PROTOWN,                   (%X'01F7DC9C'))
              ,(PROTGRP,                   (%X'01F7DCA4'))
              ,(PROTWLD,                   (%X'01F7DCAC'))
              );

        !
        !  *************************************************************************
        !  *                                                                       *
        !  *  Copyright (c) 1982, 1983                                             *
        !  *  by DIGITAL Equipment Corporation, Maynard, Mass.                     *
        !  *                                                                       *
        !  *  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.              *
        !  *                                                                       *
        !  *************************************************************************
        !  ***************************************************************************************************************
        !  Created 15-SEP-1984 22:54:35  by  VAX-11 SDL V2.0       Source: 15-SEP-1984 22:49:24 _$255$DUA28:[RMS.SRC]RMSINTS
        !  ***************************************************************************************************************


        !*** MODULE $IFBDEF ***
        !
        !   NOTE:  The fields thru JNLBDB inclusive are common between the ifb and irb
        !
        literal IFB$C_BID = 11;                 !  ifab id code
        literal IFB$M_PUT = 1;
        literal IFB$M_GET = 2;
        literal IFB$M_DEL = 4;
        literal IFB$M_UPD = 8;
        literal IFB$M_TRN = 16;
        literal IFB$M_BIO = 32;
        literal IFB$M_BRO = 64;
        literal IFB$M_EXE = 128;
        literal IFB$C_SEQ = 0;                  !  sequential
        literal IFB$C_REL = 1;                  !  relative
        literal IFB$C_IDX = 2;                  !  indexed
        literal IFB$C_DIR = 3;                  !  direct
        literal IFB$C_MAXORG = 2;               !  release 1.5 maximum
        literal IFB$K_FHAEND = 102;             !  end of file header attributes
        literal IFB$C_FHAEND = 102;             !  end of file header attributes
        literal IFB$C_KBUFNUM = 6;              !  constant - the number of key buffers allocated
        literal IFB$M_ONLY_RU = 1;
        literal IFB$M_RU = 2;
        literal IFB$M_BI = 4;
        literal IFB$M_AI = 8;
        literal IFB$M_AT = 16;
```

```
literal IFB$M_NEVER_RU = 32;
literal IFB$M_RU_RECVR = 1;
literal IFB$M_AI_RECVR = 2;
literal IFB$M_BI_RECVR = 4;
literal IFB$M_VALID_AT = 1;
literal IFB$M_JNL = 2;
literal IFB$M_RUP = 4;
literal IFB$M_RU_RLK = 8;
literal IFB$M_DONE_ASS_JNL = 16;
literal IFB$K_BLN_SEQ = 172;
literal IFB$C_BLN_SEQ = 172;
 ! --
 !
 !    organization-dependent fields
 !
 !    the following fields are used differently
 !    depending upon the file's organization
 !
 ! ++
 !
 !    relative org specific fields
 !
literal IFB$S_IFBDEF = 172;
 !  (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
 !
macro IFB$V_BUSY = 4,0,1,0 %;             !  stream busy
macro IFB$V_EOF = 4,1,1,0 %;              !  file positioned at eof
macro IFB$V_PPF_IMAGE = 4,2,1,0 %;        !  flag for indirect processing of process-
!  permanent files (restricts allowable operations)
macro IFB$V_ASYNC = 4,3,1,0 %;            !  async i/o flag (must be zero for ifab)
macro IFB$V_ASYNCWAIT = 4,4,1,0 %;        !  wait on async i/o (must be zero for ifab)
 ! --
 !
 !    ifab specific bits
 !
macro IFB$V_ACCESSED = 4,5,1,0 %;         !  file is accessed
macro IFB$V_ANSI_D = 4,6,1,0 %;           !  ansi d variable records
macro IFB$V_RWC = 4,7,1,0 %;              !  copy of fop bit from open
macro IFB$V_DMO = 4,8,1,0 %;              !  copy of fop bit from open
macro IFB$V_SPL = 4,9,1,0 %;              !  copy of fop bit from open
macro IFB$V_SCF = 4,10,1,0 %;             !  copy of fop bit from open
macro IFB$V_DLT = 4,11,1,0 %;             !  copy of fop bit from open
macro IFB$V_DFW = 4,12,1,0 %;             !  deferred write (copy of fop bit from $open)
macro IFB$V_SQO = 4,13,1,0 %;             !  sequential operations only
macro IFB$V_PPF_INPUT = 4,14,1,0 %;       !  this is command 'input' stream
macro IFB$V_NFS = 4,15,1,0 %;             !  non-file structured flag
macro IFB$V_WRTACC = 4,16,1,0 %;          !  logical or of fac bits:
!     put, upd, del, trn
macro IFB$V_MSE = 4,17,1,0 %;             !  multi-streams enabled
macro IFB$V_CREATE = 4,18,1,0 %;          !  set if doing create (may be "create if")
macro IFB$V_NORECLK = 4,19,1,0 %;         !  record locking not required
!  (i.e., no shared access or multi-stream)
macro IFB$V_RW_ATTR = 4,20,1,0 %;         !  set if file attributes must be re-written
```

```
0       macro IFB$V_TMP = 4,21,1,0 %;             ! temporary file (i.e., no directory entry)
0       macro IFB$V_TEF = 4,22,1,0 %;             ! truncate at eof due to large auto extend
0       macro IFB$V_STALL_LOCK = 4,23,1,0 %;      ! RMS is stalled for file lock
0       macro IFB$V_SEQFIL = 4,24,1,0 %;          ! this is really a sequential file being shared
0       macro IFB$V_SEARCH = 4,25,1,0 %;          ! search ifab - left during wildcard operations
0       macro IFB$V_RMS_STALL = 4,26,1,0 %;       ! RMS is stalled on this file operation
0       macro IFB$V_RESTART = 4,27,1,0 %;         ! Reopen or recreate operation in progress
0       macro IFB$V_FILEFOUND = 4,28,1,0 %;       ! A file was found on a search operation
0       macro IFB$V_DAP_OPEN = 4,29,1,0 %;        ! open/create function was performed via dap
0       macro IFB$V_DAP = 4,30,1,0 %;             ! data access protocol transmission
0       macro IFB$V_NSP = 4,31,1,0 %;             ! network services protocol transmission
0       macro IFB$L_PRIM_DEV = 0,0,32,0 %;        ! device characteristics bits
0       !  (for primary device - bit encoding same as for fab)
0       macro IFB$L_BKPBITS = 4,0,32,0 %;         ! bookkeeping bits
0       !
0       macro IFB$B_BID = 8,0,8,0 %;              ! block id
0       macro IFB$B_BLN = 9,0,8,0 %;              ! block length in longwords
0       macro IFB$B_MODE = 10,0,8,0 %;            ! caller's mode
0       macro IFB$B_EFN = 11,0,8,0 %;             ! event flag used for synchronous qio
0       macro IFB$L_IOS = 12,0,32,0 %;            ! internal i/o status block
0       macro IFB$L_BWB = 12,0,32,0 %;            ! bucket wait block for inter stream waiting
0       macro IFB$W_IOS2 = 14,0,16,0 %;           ! high word of io status block
0       macro IFB$L_IOS4 = 16,0,32,0 %;           ! 2nd longword of io status block
0       macro IFB$L_ASBADDR = 20,0,32,0 %;        ! address of asynchronous context block
0       macro IFB$L_ARGLST = 24,0,32,0 %;         ! user call parameters addr
0       macro IFB$L_IRAB_LNK = 28,0,32,0 %;       ! pointer to irab(s)
0       macro IFB$W_CHNL = 32,0,16,0 %;           ! i/o channel number
0       macro IFB$B_FAC = 34,0,8,0 %;             ! file access
0       macro IFB$V_PUT = 34,0,1,0 %;             ! (same as in fab's fac field)
0       macro IFB$V_GET = 34,1,1,0 %;
0       macro IFB$V_DEL = 34,2,1,0 %;
0       macro IFB$V_UPD = 34,3,1,0 %;
0       macro IFB$V_TRN = 34,4,1,0 %;
0       macro IFB$V_BIO = 34,5,1,0 %;
0       macro IFB$V_BRO = 34,6,1,0 %;
0       macro IFB$V_EXE = 34,7,1,0 %;
0       !  note: if both bio and bro set, implies block i/o
0       !           access only allowed for this connect, resets
0       !           to bro on disconnect (seq. file org. only).
0       !
0       macro IFB$B_ORGCASE = 35,0,8,0 %;         ! copy of org for case dispatching
0       macro IFB$L_LAST_FAB = 36,0,32,0 %;       ! address of fab for last operation
0       macro IFB$W_IFI = 40,0,16,0 %;            ! Internal file Identifier, the one we gave to the user
0       macro IFB$W_ECHO_ISI = 42,0,16,0 %;       ! ISI of stream to echo records from SYS$INPUT
0       macro IFB$L_ATJNLBUF = 44,0,32,0 %;       ! address of IFAB audit trail buffer
0       macro IFB$L_JNLBDB = 48,0,32,0 %;         ! address of Journaling BDB for FAB operations
0       ! -----*****
0       macro IFB$L_EXTJNLBUF = 52,0,32,0 %;      ! pointer to buffer to contain extend journal record
0       macro IFB$L_FWA_PTR = 56,0,32,0 %;        ! pointer to file work area control block
0       macro IFB$L_NWA_PTR = 60,0,32,0 %;        ! pointer to network work area control block
0       macro IFB$L_BDB_FLNK = 64,0,32,0 %;       ! pointer to bdb(s)
0       macro IFB$L_BDB_BLNK = 68,0,32,0 %;       ! bdb backward link
0       macro IFB$L_DEVBUFSIZ = 72,0,32,0 %;      ! device default (or bls if mt) buff size
0       macro IFB$W_RTDEQ = 76,0,16,0 %;          ! run-time default extend quantity
0       macro IFB$B_SHR = 78,0,8,0 %;             ! File sharing bits from users FAB
0       macro IFB$B_AGENT_MODE = 79,0,8,0 %;      ! User's FAB$V_FILE_MODE field, maximized with mode of caller
0       !
```

```
;  0    ! ++++++++++
;  0    !
;  0    !    the following fields must remain as is since
;  0    !    they correspond to the rms attributes stored
;  0    !    in the file header
;  0    !
;  0    macro IFB$B_RFMORG = 80,0,8,0 %;            !  organization and record format
;  0    macro IFB$V_RFM = 80,0,4,0 %;
;  0    literal IFB$S_RFM = 4;                      !  record format (n.b. constant values defined in rfm field of fab)
;  0    macro IFB$V_ORG = 80,4,4,0 %;
;  0    literal IFB$S_ORG = 4;                      !  file organization
;  0    macro IFB$B_RAT = 81,0,8,0 %;               !  record attributes (n.b. bit offsets defined in rat field of fab)
;  0    macro IFB$W_LRL = 82,0,16,0 %;              !  longest record's length (or fixed record length)
;  0    macro IFB$L_HBK_DISK = 84,0,32,0 %;         !  hi vbn allocated (note: disk format!)
;  0    macro IFB$L_EBK_DISK = 88,0,32,0 %;         !  eof vbn (note: disk format!)
;  0    macro IFB$W_FFB = 92,0,16,0 %;              !  first free byte in eof block
;  0    macro IFB$B_BKS = 94,0,8,0 %;               !  bucket size (! vbns)
;  0    macro IFB$B_FSZ = 95,0,8,0 %;               !  record header size for vfc
;  0    macro IFB$W_MRS = 96,0,16,0 %;              !  max record size allowable
;  0    macro IFB$W_DEQ = 98,0,16,0 %;              !  default extend quantity
;  0    macro IFB$W_GBC = 100,0,16,0 %;             !  global buffer count
;  0    ! -----*****
;  0    macro IFB$B_DRT_REHIT = 104,0,8,0 %;        !  hit count for local dirty buffers.
;  0    macro IFB$B_GBL_REHIT = 105,0,8,0 %;        !  rehit count for gbl buffers.
;  0    macro IFB$L_RNS_LEN = 108,0,32,0 %;         !  resultant name string length (used as a temp field by $search)
;  0    macro IFB$L_LOCK_BDB = 108,0,32,0 %;        !  lock bdb address (used by $extend for rel. file)
;  0    macro IFB$L_HBK = 112,0,32,0 %;             !  hi vbn allocated.
;  0    macro IFB$L_EBK = 116,0,32,0 %;             !  eof vbn.
;  0    macro IFB$L_SFSB_PTR = 120,0,32,0 %;        !  pointer to shared file synchronization block
;  0    macro IFB$L_GBSB_PTR = 124,0,32,0 %;        !  pointer to global buffer synchronization block.
;  0    macro IFB$L_PAR_LOCK_ID = 128,0,32,0 %;     !  Parent lock ID for bucket locks (get from SFSB.)
;  0    macro IFB$W_AVLCL = 132,0,16,0 %;           !  local buffers available.
;  0    macro IFB$W_AVGBPB = 134,0,16,0 %;          !  gbl ptr blocks available.
;  0    macro IFB$L_GBH_PTR = 136,0,32,0 %;         !  pointer to global header.
;  0    macro IFB$L_AS_DEV = 140,0,32,0 %;          !  assigned device characteristics
;  0    !                  AS_DEV and ASDEVBSIZ *)
;  0    macro IFB$L_ASDEVBSIZ = 148,0,32,0 %;       !  assigned device buffer size
;  0    macro IFB$L_BLBFLNK = 152,0,32,0 %;         !  Forward link to BLB chain.
;  0    macro IFB$L_BLBBLNK = 156,0,32,0 %;         !  Back link to BLB chain.
;  0    macro IFB$B_JNLFLG = 160,0,8,0 %;           !  journaling attribute flags
;  0    macro IFB$V_ONLY_RU = 160,0,1,0 %;          !  Recovery Unit journaling, no access outside RU
;  0    macro IFB$V_RU = 160,1,1,0 %;               !  Recovery Unit journaling
;  0    macro IFB$V_BI = 160,2,1,0 %;               !  Before Image journaling
;  0    macro IFB$V_AI = 160,3,1,0 %;               !  After Image jouraling
;  0    macro IFB$V_AT = 160,4,1,0 %;               !  Audit Trail journaling
;  0    macro IFB$V_NEVER_RU = 160,5,1,0 %;         !  never do RU journaling
;  0    macro IFB$B_RECVRFLGS = 161,0,8,0 %;        !  Recovery flags
;  0    macro IFB$V_RU_RECVR = 161,0,1,0 %;         !  Recovery Unit Rollback in progress
;  0    macro IFB$V_AI_RECVR = 161,1,1,0 %;         !  AI Roll Forward Recovery in progress
;  0    macro IFB$V_BI_RECVR = 161,2,1,0 %;         !  BI Roll Backward Recovery in progress
;  0    macro IFB$B_JNLFLG2 = 162,0,8,0 %;          !  Secondary journaling flags (generally operation specific)
;  0    macro IFB$V_VALID_AT = 162,0,1,0 %;         !  AT entry in IFB buffer is valid and should be written
;  0    macro IFB$V_JNL = 162,1,1,0 %;              !  Journaling Initialized for this file
;  0    macro IFB$V_RUP = 162,2,1,0 %;              !  Recovery Unit in progress
;  0    macro IFB$V_RU_RLK = 162,3,1,0 %;           !  Fake record locking during recovery unit
;  0    macro IFB$V_DONE_ASS_JNL = 162,4,1,0 %;     !  Journal channels already assigned
;  0    macro IFB$L_RJB = 164,0,32,0 %;             !  RMS Journaling Block address
```

```
macro IFB$W_BUFFER_OFFSET = 168,0,16,0 %; !  ANSI buffer offset
literal IFB$K_BLN_REL = 180;
literal IFB$C_BLN_REL = 180;
! --
! ++
!
!    indexed org specific fields
!
literal IFB$S_IFBDEF1 = 180;
macro IFB$L_MRN = 172,0,32,0 %;          !  (rel) max record number
macro IFB$L_DVBN = 176,0,32,0 %;         !  (rel) first data bucket vbn
literal IFB$K_BLN_IDX = 184;
literal IFB$C_BLN_IDX = 184;
literal IFB$K_BLN = 184;                 !  ifab length
literal IFB$C_BLN = 184;                 !  ifab length
! --
literal IFB$S_IFBDEF2 = 184;
macro IFB$L_IDX_PTR = 172,0,32,0 %;      !  (idx) pointer to primary key index descriptor
macro IFB$B_AVBN = 176,0,8,0 %;          !  (idx) vbn of 1st area descriptor
macro IFB$B_AMAX = 177,0,8,0 %;          !  (idx) total number of area descriptors
macro IFB$B_NUM_KEYS = 178,0,8,0 %;      !  (idx) ! of keys in file
macro IFB$B_UBUFSZ = 179,0,8,0 %;        !  (idx) update buffer size for keys
macro IFB$W_KBUFSZ = 180,0,16,0 %;       !  (idx) key buffer size
macro IFB$B_EXTRABUF = 182,0,8,0 %;      !  (idx) number of extra buffers for 'cache'ing
macro IFB$B_PLG_VER = 183,0,8,0 %;       !  (idx) prologue version number

!*** MODULE $IRBDEF ***
!
!          IRB field definitions
!
!          Internal rab (irb)
!
!          There is 1 irab per connected record access stream
!
!
!   NOTE: The fields thru JNLBDB inclusive are common between the irb and ifb
!
literal IRB$C_BID = 10;                  !  irab code
literal IRB$M_POSINSERT = 1;
literal IRB$M_SRCHGT = 2;
literal IRB$M_POSDELETE = 4;
literal IRB$M_NEW_IDX = 8;
literal IRB$M_SRCHGE = 16;
literal IRB$M_NORLS_RNF = 32;
literal IRB$M_FIRST_TIM = 64;
literal IRB$M_PRM = 128;
literal IRB$M_DUP_KEY = 256;
literal IRB$M_DEL_SEEN = 512;
literal IRB$M_LAST_GT = 1024;
literal IRB$M_BKT_NO_LO = 1;
literal IRB$M_NEW_BKTS = 6;
literal IRB$M_REC_W_LO = 8;
literal IRB$M_CONT_BKT = 16;
literal IRB$M_CONT_R = 32;
literal IRB$M_EMPTY_BKT = 64;
literal IRB$M_DUPS_SEEN = 128;
literal IRB$M_BKT_NO = 3;
```

```
literal IRB$M_BIG_SPLIT = 4;
literal IRB$M_SPL_IDX = 1;
literal IRB$M_EMPT_SEEN = 2;
literal IRB$M_VALID_AT = 1;
literal IRB$K_BLN_REL = 100;
literal IRB$C_BLN_REL = 100;
! ++
!
!   sequential org specific fields
!
literal IRB$K_BLN_SEQ = 108;
literal IRB$C_BLN_SEQ = 108;
literal IRB$S_IRBDEF = 108;
!  to apply from start of irab
! ++
!
!   the following bits are defined in common
!   with the ifab
!
macro IRB$V_BUSY = 4,0,1,0 %;              !   file busy
macro IRB$V_EOF = 4,1,1,0 %;               !   stream positioned at eof
macro IRB$V_PPF_IMAGE = 4,2,1,0 %;         !   flag for indirect processing of process-
!  permanent file
macro IRB$V_ASYNC = 4,3,1,0 %;             !   asynchronous i/o request
macro IRB$V_ASYNCWAIT = 4,4,1,0 %;         !   $wait issued for asynchronous i/o request
! --
!
!    irab specific bits
!
macro IRB$V_FIND_LAST = 4,5,1,0 %;         !   last operation was a find
macro IRB$V_PUTS_LAST = 4,6,1,0 %;         !   last operation was a put sequential
macro IRB$V_BIO_LAST = 4,7,1,0 %;          !   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$rset
!          refers to the current operation and bro_sw gives
!          type of last operation.
macro IRB$V_BRO_SW = 4,8,1,0 %;            !   switched from record operation to block i/o operation
macro IRB$V_FIND = 4,9,1,0 %;              !   operation is a find
macro IRB$V_RAHWBH = 4,10,1,0 %;           !   read ahead or write behind processing
macro IRB$V_SKIP_NEXT = 4,11,1,0 %;        !   skip to next record flag for index fo
macro IRB$V_DUP = 4,12,1,0 %;              !   duplicate records seen
macro IRB$V_UNLOCK_RP = 4,13,1,0 %;        !   release lock on current (rp) record
macro IRB$V_PPF_EOF = 4,14,1,0 %;          !   give one-shot rms$_eof error on sys$input
macro IRB$V_PPF_SKIP = 4,15,1,0 %;         !   skip sys$input record ($deck), redoing $get
!  or $find on next record
macro IRB$V_PPF_FNDSV = 4,16,1,0 %;        !   save value for find bit when ppf_skip set
macro IRB$V_IDX_ERR = 4,17,1,0 %;          !   index update error occurred
macro IRB$V_RRV_ERR = 4,18,1,0 %;          !   rrv update error occurred
macro IRB$V_UPDATE = 4,19,1,0 %;           !   operation is an update (indexed)
macro IRB$V_UPDATE_IF = 4,20,1,0 %;        !   operation was a $PUT -> $UPDATE
macro IRB$V_ABOVELCKD = 4,21,1,0 %;        !   level above was locked by search_tree
macro IRB$V_GBLBUFF = 4,22,1,0 %;          !   global buffers are in use.
macro IRB$V_CON_EOF = 4,23,1,0 %;          !   file positioned at EOF by $CONNECT (isam)
macro IRB$V_NO_Q_WAIT = 4,24,1,0 %;        !   do not wait for enqueues on query locks
macro IRB$V_PPF_ECHO = 4,25,1,0 %;         !   echo SYS$INPUT records to SYS$OUTPUT
macro IRB$V_RMS_STALL = 4,26,1,0 %;        !   RMS is stalled on this record operation
macro IRB$V_RESTART = 4,27,1,0 %;          !   Reconnect operation in progress
```

```
;  0   macro IRB$V_DAP_CONN = 4,28,1,0 %;        ! connect function was performed via dap
;  0   macro IRB$V_RU_DELETE = 4,29,1,0 %;       ! recovery unit deletion in progress
;  0   macro IRB$V_RU_UNDEL = 4,30,1,0 %;        ! recovery unit un-deletion in progress
;  0   macro IRB$V_RU_UPDATE = 4,31,1,0 %;       ! place new record is special RU UPDATE format
;  0   !
;  0   !   the following are alternate definitions for alternate
;  0   !   (non-conflicting) use of the above bits
;  0   !
;  0   macro IRB$V_WRITE = 4,9,1,0 %;            ! operation is a write
;  0   macro IRB$L_IFAB_LNK = 0,0,32,0 %;        ! pointer to ifab
;  0   macro IRB$L_BKPBITS = 4,0,32,0 %;         ! bookkeeping status bits
;  0   !
;  0   macro IRB$B_BID = 8,0,8,0 %;              ! block id
;  0   macro IRB$B_BLN = 9,0,8,0 %;              ! block length in longwords
;  0   macro IRB$B_MODE = 10,0,8,0 %;            ! caller's mode
;  0   macro IRB$B_EFN = 11,0,8,0 %;             ! event flag for synchronous io
;  0   macro IRB$L_IOS = 12,0,32,0 %;            ! internal i/o status block
;  0   macro IRB$L_BWB = 12,0,32,0 %;            ! bucket wait block for inter stream locking
;  0   macro IRB$W_IOS2 = 14,0,16,0 %;           ! high word of io status block
;  0   macro IRB$L_IOS4 = 16,0,32,0 %;           ! io status block (2nd longword)
;  0   macro IRB$L_ASBADDR = 20,0,32,0 %;        ! address of permanent asynchronous context block
;  0   macro IRB$L_ARGLST = 24,0,32,0 %;         ! user arg list address
;  0   !  if async, points to copy at head
;  0   !  of async context block
;  0   macro IRB$L_IRAB_LNK = 28,0,32,0 %;       ! pointer to next irab
;  0   macro IRB$L_CURBDB = 32,0,32,0 %;         ! current bdb address
;  0   macro IRB$L_LAST_RAB = 36,0,32,0 %;       ! address of rab for last operation
;  0   macro IRB$W_ISI = 40,0,16,0 %;            ! Internal stream Identifier, the one we gave to the user
;  0   macro IRB$L_ATJNLBUF = 44,0,32,0 %;       ! address of IRAB audit trail journaling buffer
;  0   macro IRB$L_JNLBDB = 48,0,32,0 %;         ! address of journaling BDB for RAB operations
;  0   ! -----*****
;  0   macro IRB$L_IDENT = 52,0,32,0 %;          ! process unique identifier for the IRB
;  0   macro IRB$L_RLB_LNK = 56,0,32,0 %;        ! pointer to RLBs
;  0   macro IRB$L_NXTBDB = 60,0,32,0 %;         ! next bdb address
;  0   macro IRB$L_NRP = 64,0,32,0 %;            ! next record pointer (relative record number)
;  0   macro IRB$L_NRP_VBN = 64,0,32,0 %;        ! next record pointer (relative)
;  0   macro IRB$B_CACHEFLGS = 64,0,8,0 %;       ! cacheflags for calls to getbkt,cache, etc. (indexed)
;  0   macro IRB$B_STOPLEVEL = 65,0,8,0 %;       ! level to stop at on tree search (indexed)
;  0   macro IRB$W_SRCHFLAGS = 66,0,16,0 %;      ! search flags (indexed)
;  0   macro IRB$V_POSINSERT = 66,0,1,0 %;       ! position for insert
;  0   macro IRB$V_SRCHGT = 66,1,1,0 %;          ! approximate search gt
;  0   macro IRB$V_POSDELETE = 66,2,1,0 %;       ! position for delete
;  0   macro IRB$V_NEW_IDX = 66,3,1,0 %;         ! need to read in new idx dsc from file
;  0   macro IRB$V_SRCHGE = 66,4,1,0 %;          ! approximate search ge
;  0   macro IRB$V_NORLS_RNF = 66,5,1,0 %;       ! don't release bkt on rnf error, if set
;  0   macro IRB$V_FIRST_TIM = 66,6,1,0 %;       ! flag to indicate 1st time for seq. processing
;  0   macro IRB$V_PRM = 66,7,1,0 %;             ! flag to indicate that the permanence bit in the bdb
;  0   macro IRB$V_DUP_KEY = 66,8,1,0 %;         ! a duplicate key seen on scan of any data bucket
;  0   macro IRB$V_DEL_SEEN = 66,9,1,0 %;        ! a deleted record has been encountered between current
;  0   macro IRB$V_LAST_GT = 66,10,1,0 %;        ! result of last search of compressed key bucket was GT
;  0   !  and a next record during a $GET/$FIND
;  0   !    should be set
;  0   macro IRB$L_NRP_OFF = 68,0,32,0 %;        ! next record pointer offset (relative)
;  0   macro IRB$L_CURVBN = 68,0,32,0 %;         ! vbn of current record (relative)
;  0   macro IRB$W_NRP_OFF = 68,0,16,0 %;        ! ''
;  0   macro IRB$B_SPL_BITS = 68,0,8,0 %;        ! bits for splitting (indexed)
;  0   macro IRB$V_BKT_NO_LO = 68,0,1,0 %;       ! low bit of bucket number processing
```

```
0   macro IRB$V_NEW_BKTS = 68,1,2,0 %;
0   literal IRB$S_NEW_BKTS = 2;                 ! number of new buckets (0-3)
0   macro IRB$V_REC_W_LO = 68,3,1,0 %;          ! if splitting at pos_insert than rec goes w/ lo
0   macro IRB$V_CONT_BKT = 68,4,1,0 %;          ! middle bucket is a continuation bkt
0   macro IRB$V_CONT_R = 68,5,1,0 %;            ! right bucket is a continuation bkt
0   macro IRB$V_EMPTY_BKT = 68,6,1,0 %;         ! bucket contains no data records
0   macro IRB$V_DUPS_SEEN = 68,7,1,0 %;         ! dups seen on scan of bucket, any key
0   macro IRB$V_BKT_NO = 68,0,2,0 %;
0   literal IRB$S_BKT_NO = 2;
0   macro IRB$V_BIG_SPLIT = 68,2,1,0 %;
0   macro IRB$V_SPL_IDX = 68,0,1,0 %;           ! split up new index record and swing pointer
0   macro IRB$V_EMPT_SEEN = 68,1,1,0 %;         ! empty bucket passed over on posinsert
0   macro IRB$L_RP = 72,0,32,0 %;               ! record pointer (relative record !)
0   macro IRB$L_RP_VBN = 72,0,32,0 %;           ! record pointer (relative)
0   macro IRB$W_POS_INS = 72,0,16,0 %;          ! offset for position for insert for put (indexed)
0   macro IRB$W_SPLIT = 74,0,16,0 %;            ! first split point (indexed)
0   macro IRB$L_RP_OFF = 76,0,32,0 %;           ! record pointer offset
0   macro IRB$L_LST_REC = 76,0,32,0 %;          ! last record address (indexed)
0   macro IRB$L_PTR_VBN = 76,0,32,0 %;          ! pointer vbn used by find_by_rrv (indexed)
0   macro IRB$W_RP_OFF = 76,0,16,0 %;           ! record pointer offset
0   macro IRB$W_SPLIT_1 = 76,0,16,0 %;          ! second split point -- 3-bkt split (indexed)
0   macro IRB$W_SPLIT_2 = 78,0,16,0 %;          ! third split point -- 4-bkt split (indexed)
0   macro IRB$L_OWNER_ID = 80,0,32,0 %;         ! owner id used for record locks
0   macro IRB$W_OWN_ID = 80,0,16,0 %;           ! index part of process id (pid)
0   macro IRB$W_OWN_ISI = 82,0,16,0 %;          ! isi value for this irab
0   macro IRB$B_PPF_ISI = 82,0,8,0 %;           ! isi value for this process-permanent irab
0   macro IRB$B_BCNT = 84,0,8,0 %;              ! i/o buffer count
0   macro IRB$B_MBC = 85,0,8,0 %;               ! multi-block count
0   macro IRB$W_RSZ = 86,0,16,0 %;              ! record size from user
0   macro IRB$L_RBF = 88,0,32,0 %;              ! user record buffer address
0   macro IRB$B_MBF = 92,0,8,0 %;               ! Multi-buffer count from user's RAB
0   macro IRB$B_JNLFLG3 = 93,0,8,0 %;           ! IRB journaling flags
0   macro IRB$V_VALID_AT = 93,0,1,0 %;          ! IRB MJB contains valid AT entry to write
0   ! ++
0   !
0   !    start of organization dependent fields
0   !
0   ! ++
0   !
0   !  used by sequential and relative files
0   !
0   macro IRB$W_CSIZ = 98,0,16,0 %;             ! current record size (seq)
0   ! ++
0   !
0   !    relative org specific fields
0   !
0   macro IRB$L_TEMPO = 100,0,32,0 %;
0   macro IRB$W_ROVHDSZ = 100,0,16,0 %;         ! overhead size for record
0   macro IRB$B_PRE_CCTL = 100,0,8,0 %;         ! 'pre' carriage control
0   macro IRB$B_POST_CCTL = 101,0,8,0 %;        ! 'post' carriage control
0   macro IRB$W_RTOTLSZ = 102,0,16,0 %;         ! total size for record
0   macro IRB$L_TEMP1 = 104,0,32,0 %;
0   macro IRB$B_NVBNS = 104,0,8,0 %;            ! number of vbns transferred (nxtblk1)
0   !
0   !  indexed org specific fields
0   !
0   literal IRB$K_BLN_IDX = 196;
```

```
literal IRB$C_BLN_IDX = 196;
literal IRB$S_IRBDEF1 = 196;
macro IRB$L_KEYBUF = 96,0,32,0 %;           ! address of internal key buffer & update buffer
macro IRB$L_UPDBUF = 100,0,32,0 %;          ! address of internal update buffer
macro IRB$L_RECBUF = 104,0,32,0 %;          ! address of internal record buffer
macro IRB$L_OLDBUF = 108,0,32,0 %;          ! address of internal old record buffer (updates only)
macro IRB$L_RFA_VBN = 112,0,32,0 %;         ! save record vbn for nrp data
macro IRB$L_UPD_BDB = 112,0,32,0 %;         ! save current bdb during insert operation
macro IRB$L_LAST_VBN = 112,0,32,0 %;        ! last vbn at data level for update
macro IRB$W_RFA_ID = 116,0,16,0 %;          ! save record id for search data
macro IRB$W_LAST_ID = 116,0,16,0 %;         ! id for udr during update (plg 3)
macro IRB$W_SAVE_POS = 118,0,16,0 %;        ! save duplicate position for nrp data
macro IRB$L_NEXT_VBN = 120,0,32,0 %;        ! save next user data record VBN for nrp data
macro IRB$L_PUTUP_VBN = 120,0,32,0 %;       ! RFA VBN of $PUT/$UPDATE record
macro IRB$L_FIRST_VBN = 124,0,32,0 %;       ! save SIDR first element VBN for search NRP data
macro IRB$W_NEXT_ID = 128,0,16,0 %;         ! save next user data record ID for nrp data
macro IRB$W_PUTUP_ID = 128,0,16,0 %;        ! ID of $PUT/$UPDATE record
macro IRB$W_FIRST_ID = 130,0,16,0 %;        ! save SIDR first element ID for search NRP data
macro IRB$L_LOCK_BDB = 132,0,32,0 %;        ! lock bdb addr of level below on splits
macro IRB$L_VBN_LEFT = 136,0,32,0 %;        ! left vbn of split
macro IRB$L_MIDX_TMP1 = 136,0,32,0 %;       ! temporary one for make index
macro IRB$L_VBN_RIGHT = 140,0,32,0 %;       ! right vbn of split
macro IRB$L_MIDX_TMP2 = 140,0,32,0 %;       ! temporary two for make index
macro IRB$L_VBN_MID = 144,0,32,0 %;         ! middle vbn of split
macro IRB$L_MIDX_TMP3 = 144,0,32,0 %;       ! temporary three for make index
macro IRB$L_NEXT_DOWN = 144,0,32,0 %;       ! used by search_tree
macro IRB$L_REC_COUNT = 148,0,32,0 %;       ! number of current record in this bucket (plg 3)
macro IRB$L_LST_NCMP = 152,0,32,0 %;        ! address of last key with zero front compression (plg 3)
macro IRB$L_SPL_COUNT = 156,0,32,0 %;       ! number of the first record to be moved into new bucket
!   when splitting indexes and SIDRs
macro IRB$W_NID_RIGHT = 160,0,16,0 %;       ! Next record ID of the right bucket
macro IRB$W_NID_MID = 162,0,16,0 %;         ! Next record ID of the middle bucket
macro IRB$W_RFA_NID = 164,0,16,0 %;         ! Next record ID of the RFA bucket
macro IRB$B_KEYSZ = 166,0,8,0 %;            ! size of key in keybuffer !2
macro IRB$L_CUR_VBN = 168,0,32,0 %;         ! VBN of current record (primary/SIDR)
macro IRB$L_POS_VBN = 172,0,32,0 %;         ! VBN of primary data record for NRP positioning
macro IRB$L_UDR_VBN = 176,0,32,0 %;         ! VBN of current primary data record
macro IRB$L_SIDR_VBN = 180,0,32,0 %;        ! SIDR array first element VBN of current record (SIDR)
macro IRB$W_CUR_ID = 184,0,16,0 %;          ! ID of current record (primary)
macro IRB$W_POS_ID = 186,0,16,0 %;          ! ID of primary data record for NRP positioning
macro IRB$W_UDR_ID = 188,0,16,0 %;          ! ID of current primary data record
macro IRB$W_SIDR_ID = 190,0,16,0 %;         ! SIDR array first element ID of current record (SIDR)
macro IRB$W_CUR_COUNT = 192,0,16,0 %;       ! SIDR array count of current record (SIDR)
macro IRB$B_RP_KREF = 194,0,8,0 %;          ! Key of reference by which next record is retrieved
macro IRB$B_CUR_KREF = 195,0,8,0 %;         ! Key of reference of current record (primary/SIDR)

!*** 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$c_bln_xxx must be longword aligned
!
literal ASB$C_BID = 13;                     !  asb id = 13
literal ASB$K_BLN_FIX = 48;                 !  block length of fixed asb
literal ASB$C_BLN_FIX = 48;                 !  block length of fixed asb
!    regs 4 and 5 are saved on stack
literal ASB$K_BLN_SEQ = 188;                !  block length for seq org irab operations
literal ASB$C_BLN_SEQ = 188;                !  block length for seq org irab operations
literal ASB$K_BLN_REL = 192;                !  block length for rel org irab operations
literal ASB$C_BLN_REL = 192;                !  block length for rel org irab operations
literal ASB$K_BLN_FAB = 352;                !  block length for fab-related operations
literal ASB$C_BLN_FAB = 352;                !  block length for fab-related operations
literal ASB$K_BLN_IDX = 512;
literal ASB$C_BLN_IDX = 512;
literal ASB$S_ASBDEF = 512;
macro ASB$W_STKLEN = 0,0,16,0 %;            !  save stack length (must be first word in block)
!    STKLEN = BLN_org - BLN_FIX
macro ASB$W_STKSIZ = 2,0,16,0 %;            !  size of saved stack in bytes
macro ASB$B_BID = 8,0,8,0 %;                !  block id
macro ASB$B_BLN = 9,0,8,0 %;                !  block length in longwords
macro ASB$L_ARGLST = 12,0,0,0 %;
literal ASB$S_ARGLST = 16;                  !  saved argument list on async irab operations
macro ASB$B_ARGCNT = 12,0,8,0 %;            !  argument count
!   value will be 0, 1, 2, or 3
macro ASB$L_FABRAB = 16,0,32,0 %;           !  fab or rab address
macro ASB$L_ERR = 20,0,32,0 %;              !  err routine addr
macro ASB$L_SUC = 24,0,32,0 %;              !  suc routine addr
macro ASB$L_REGS = 28,0,0,0 %;
literal ASB$S_REGS = 20;                    !  save register area for regs 6, 7, 8, 10 and 11
macro ASB$L_STK = 48,0,0,0 %;
literal ASB$S_STK = 140;                    !  saved stack area

!*** MODULE $BDBDEF ***
!
!        BDB field definitions
!
!       buffer descriptor block (bdb)
!
!       there is one bdb per i/o buffer
!       ( the i/o buffers exist in separate pages, page aligned)
!
literal BDB$C_BID = 12;                     !  bdb id code
literal BDB$M_VAL = 1;
literal BDB$M_DRT = 2;
literal BDB$M_IOP = 4;
literal BDB$M_PRM = 8;
literal BDB$M_NOLOCATE = 16;
literal BDB$M_WFO = 32;
literal BDB$M_AST_DCL = 64;
literal BDB$S_BDBDEF = 80;
macro BDB$L_FLINK = 0,0,32,0 %;             !  forward link
macro BDB$L_BLINK = 4,0,32,0 %;             !  backward link
macro BDB$B_BID = 8,0,8,0 %;                !  block id
macro BDB$B_BLN = 9,0,8,0 %;                !  block length in longwords
macro BDB$B_FLGS = 10,0,8,0 %;              !  bdb flags
macro BDB$V_VAL = 10,0,1,0 %;               !  buffer contents valid
```

```
macro BDB$V_DRT = 10,1,1,0 %;              ! buffer content dirty
macro BDB$V_IOP = 10,2,1,0 %;              ! buffer has i/o in progress
macro BDB$V_PRM = 10,3,1,0 %;              ! buffer has permanence factor
macro BDB$V_NOLOCATE = 10,4,1,0 %;         ! buffer shared - no locate mode
macro BDB$V_WFO = 10,5,1,0 %;              ! other streams awaiting
macro BDB$V_AST_DCL = 10,6,1,0 %;          ! ast has been declared for
!  waiting stream
!  the releasing of this bdb
!  (set/cleared by rm$cache)
macro BDB$B_CACHE_VAL = 11,0,8,0 %;        ! relative value of buffer in cache
macro BDB$B_VERTYP = 11,0,8,0 %;           ! version type (1 = wild)
macro BDB$W_USERS = 12,0,16,0 %;           ! number of streams referencing this buffer
macro BDB$W_BUFF_ID = 14,0,16,0 %;         ! buffer identification number
macro BDB$L_BLB_PTR = 16,0,32,0 %;         ! pointer to BLB chain for this BDB
macro BDB$W_NUMB = 20,0,16,0 %;            ! ! of bytes of buffer in use
macro BDB$W_DIRSEQ = 20,0,16,0 %;          ! UCB$W_DIRSEQ at directory read time
macro BDB$W_SIZE = 22,0,16,0 %;            ! ! bytes in buffer
macro BDB$L_ADDR = 24,0,32,0 %;            ! address of buffer
macro BDB$L_VBN = 28,0,32,0 %;             ! 1st vbn in buffer
macro BDB$L_VBNSEQNO = 32,0,32,0 %;        ! vbn seq number of validity check vs. bcb copy
macro BDB$L_LAST = 32,0,32,0 %;            ! address of last directory record
macro BDB$L_WAIT = 36,0,32,0 %;            ! wait thread (irab addr)
!   (for inter-stream intra-
!    process locking only)
macro BDB$L_VERCOUNT = 36,0,32,0 %;        ! negative count of version entries scanned
macro BDB$L_ALLOC_ADDR = 40,0,32,0 %;      ! buffer allocation addr
macro BDB$W_ALLOC_SIZE = 44,0,16,0 %;      ! buffer allocation size
macro BDB$L_BI_BDB = 48,0,32,0 %;          ! address of isam/block i/o bi journaling BDB
macro BDB$L_AI_BDB = 52,0,32,0 %;          ! address of isam/block i/o ai journaling BDB
macro BDB$T_JNLSEQ = 56,0,0,0 %;
literal BDB$S_JNLSEQ = 16;                 ! Journaling Sequence Number Block
macro BDB$L_WR1 = 72,0,32,0 %;             ! work area
macro BDB$B_REL_VBN = 72,0,8,0 %;          ! current vbn rel to start of buffer
macro BDB$B_VAL_VBNS = 73,0,8,0 %;         ! ! of valid vbns in buffer
macro BDB$B_PRE_CCTL = 74,0,8,0 %;         ! unit record carriage control byte ('pre')
macro BDB$B_POST_CCTL = 75,0,8,0 %;        ! unit record carriage control byte ('post')
macro BDB$L_CURBUFADR = 76,0,32,0 %;       ! current buffer addr
literal BDB$K_BLN = 80;                    ! length of bdb block
literal BDB$C_BLN = 80;                    ! length of bdb block
literal BDB$S_BDBDEF1 = 80;
macro BDB$L_IOSB = 72,0,0,0 %;
literal BDB$S_IOSB = 8;                    ! i/o status block for buffer
macro BDB$L_VERSION = 72,0,32,0 %;         ! addr of current/next directory version entry
macro BDB$L_RECORD = 76,0,32,0 %;          ! address of current/next directory record

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

```
literal GBPB$C_BID = 21;                        !  gbpb id code
literal GBPB$K_BLN = 40;                        !  Length of GBPB block
literal GBPB$C_BLN = 40;                        !  Length of GBPB block
literal GBPB$S_GBPBDEF = 40;
macro GBPB$L_FLINK = 0,0,32,0 %;                !  forward link
macro GBPB$L_BLINK = 4,0,32,0 %;                !  backward link
macro GBPB$B_BID = 8,0,8,0 %;                   !  block id
macro GBPB$B_BLN = 9,0,8,0 %;                   !  block length in longwords
macro GBPB$B_FLGS = 10,0,8,0 %;                 !  gbpb flags (use BDB flgs definitions)
macro GBPB$B_CACHE_VL = 11,0,8,0 %;             !  relative cache value of this buffer
macro GBPB$W_USERS = 12,0,16,0 %;               !  number of streams referencing this buffer
macro GBPB$W_BUFF_ID = 14,0,16,0 %;             !  buffer identification number
macro GBPB$L_BLB_PTR = 16,0,32,0 %;             !  pointer to BLB chain for this GBPB
macro GBPB$W_NUMB = 20,0,16,0 %;                !  ! of bytes of buffer in use
macro GBPB$W_SIZE = 22,0,16,0 %;                !  ! bytes in buffer
macro GBPB$L_ADDR = 24,0,32,0 %;                !  address of buffer
macro GBPB$L_VBN = 28,0,32,0 %;                 !  1st vbn in buffer
macro GBPB$L_VBNSEQNO = 32,0,32,0 %;            !  sequence number field.
macro GBPB$L_GBD_PTR = 36,0,32,0 %;             !  Pointer to the GBD for this buffer.

!*** MODULE $RLBDEF ***
!
!               RLB field definitions
!
!               record lock block (rlb)
!
!               The rlb describes one locked record for a particular
!               process-record stream (rab/irab). if the owner field
!               is 0 then the rlb is available for use. otherwise, it
!               describes a locked record. note: when owner is 0 the
!               record rfa fields are zeroed (0).
!
!
!                       +---------------------------------------+
!               rlb:    !                  link                  !
!                       +-------------------+-------------------+
!                       !                   !        rfa4  id          !
!                       +---------+---------+---------+---------+
!                       !  flags  !reserved!   bln   !   bid   !
!                       +---------+---------+---------+---------+
!                       !                 rfa0                  !
!                       +---------------------------------------+
!                        !                  owner                          !
!                        +-------------------+-------------------+
!               lksb:    ! Still to be def-  !  VMS status code  !
!                       ! ined status bits  !                   !
!                       +-------------------+-------------------+
!                       !  Lock Id. (Returned for new locks,    !
!                       !         input for conversions)        !
!                       +---------------------------------------+
!
literal RLB$M_TIMER_INPROG = 1;
literal RLB$C_BID = 14;                         !  rlb code
literal RLB$M_WAIT = 1;
literal RLB$M_CR = 2;
literal RLB$M_PW = 4;
literal RLB$M_PR = 8;
```

```
0    literal RLB$M_CONV = 16;
0    literal RLB$M_LV2 = 32;
0    literal RLB$M_FAKE = 64;
0    literal RLB$M_TMO = 128;
0    literal RLB$K_BLN = 28;                      !  length of rlb
0    literal RLB$C_BLN = 28;                      !  length of rlb
0    literal RLB$S_RLBDEF = 28;
0    macro RLB$L_LNK = 0,0,32,0 %;                !  link to next rlb
0    macro RLB$L_MISC = 4,0,32,0 %;               !  longword definition to optimize clearing field
0    macro RLB$W_FLAGS2 = 4,0,16,0 %;             !  more flag bits
0    macro RLB$V_TIMER_INPROG = 4,0,1,0 %;        !  Timer queued.
0    macro RLB$W_RFA4 = 6,0,16,0 %;               !  3'rd word of records rfa
0    !   offset for seq f.o. (bits 0:14)
0    !   always 0 for rel f.o. (bits 0:14)
0    macro RLB$W_ID = 6,0,16,0 %;                 !  id for idx f.o.
0    macro RLB$B_BID = 8,0,8,0 %;                 !  block id
0    macro RLB$B_BLN = 9,0,8,0 %;                 !  block length in longwords
0    macro RLB$B_TMO = 10,0,8,0 %;                !  propagation of ROP TMO field
0    macro RLB$B_FLAGS = 11,0,8,0 %;              !  various locking flags
0    macro RLB$V_WAIT = 11,0,1,0 %;               !  propagation of ROP WAT bit
0    macro RLB$V_CR = 11,1,1,0 %;                 !  defines lock manager mode "concurrent read"
0    macro RLB$V_PW = 11,2,1,0 %;                 !  allow reader access to locked record flag
0    macro RLB$V_PR = 11,3,1,0 %;                 !  used to query lock database
0    macro RLB$V_CONV = 11,4,1,0 %;               !  defines lock manager option "convert"
0    macro RLB$V_LV2 = 11,5,1,0 %;                !  sets lock as "level 2" RU consistancy
0    macro RLB$V_FAKE = 11,6,1,0 %;               !  this RLB contains no lock.
0    macro RLB$V_TMO = 11,7,1,0 %;                !  propagation of ROP TMO bit
0    !   indicate "lock for write, allow readers"
0    !   used to query lock database for records
0    macro RLB$L_RFA0 = 12,0,32,0 %;              !  1'st and 2'nd words of record's rfa
0    !   seq f.o. vbn
0    !   rel f.o. relative record number
0    !   idx f.o. start vbn
0    macro RLB$L_OWNER = 16,0,32,0 %;             !  identification of owning stream
0    macro RLB$L_LKSB = 20,0,32,0 %;              !  first longword of lock status block
0    macro RLB$W_STATUS = 20,0,16,0 %;            !  VMS status code
0    macro RLB$W_S_BITS = 22,0,16,0 %;            !  various status bits
0    macro RLB$L_LOCK_ID = 24,0,32,0 %;           !  second longword of lksb is lock_id
0
0    !*** MODULE $FLBDEF ***
0    !
0    !     file lock block definitions
0    !
0    literal FLB$C_BID = 23;
0    literal FLB$K_BLN = 20;
0    literal FLB$C_BLN = 20;
0    literal FLB$S_FLBDEF = 20;
0    macro FLB$L_FLB_LNK = 0,0,32,0 %;            !  pointer to next FLB
0    macro FLB$L_RLB_LNK = 4,0,32,0 %;            !  pointer to RLBs
0    macro FLB$B_BID = 8,0,8,0 %;                 !  block id
0    macro FLB$B_BLN = 9,0,8,0 %;                 !  block length
0    macro FLB$L_IFB_PTR = 12,0,32,0 %;           !  IFAB address
0    macro FLB$L_LOCK_ID = 16,0,32,0 %;           !  lock id
0
0    !*** MODULE $DRCDEF ***
0    !
0    !     directory cache node definitions
```

```
!
literal DRC$K_BLN = 62;                 ! length of directory cache node
literal DRC$C_BLN = 62;                 ! length of directory cache node
literal DRC$S_DRCDEF = 62;
macro DRC$L_NXTFLNK = 0,0,32,0 %;       ! link to next entry, this level
macro DRC$L_NXTBLNK = 4,0,32,0 %;       ! link to previous entry, this level
macro DRC$L_LVLFLNK = 8,0,32,0 %;       ! link to first entry, next lower level
macro DRC$L_LVLBLNK = 12,0,32,0 %;      ! link to last entry, next lower level
!  note: the links are maintained in lru order
macro DRC$T_NAME = 16,0,0,0 %;
literal DRC$S_NAME = 40;                ! directory name or device and unit
!  note: stored as counted string counting count itself
macro DRC$W_DID = 56,0,0,0 %;
literal DRC$S_DID = 6;                  ! file id for directory
macro DRC$W_DIRSEQ = 58,0,16,0 %;       ! directory sequence ! for device node

!*** MODULE $RLSDEF ***
!
!                release option flag definitions
!
literal RLS$M_RETURN = 1;
literal RLS$M_WRT_THRU = 2;
literal RLS$M_KEEP_LOCK = 4;
literal RLS$M_DEQ = 8;
literal RLS$S_RLSDEF = 1;
macro RLS$V_RETURN = 0,0,1,0 %;         ! return buffer and bdb to free space lists
macro RLS$V_WRT_THRU = 0,1,1,0 %;       ! write buffer if dirty
macro RLS$V_KEEP_LOCK = 0,2,1,0 %;      ! keep bdb locked
macro RLS$V_DEQ = 0,3,1,0 %;            ! always release lock

!*** MODULE $CSHDEF ***
!
!                cache option flag definitions
!
literal CSH$M_LOCK = 1;
literal CSH$M_NOWAIT = 2;
literal CSH$M_NOREAD = 4;
literal CSH$M_NOBUFFER = 8;
literal CSH$S_CSHDEF = 1;
macro CSH$V_LOCK = 0,0,1,0 %;           ! obtain exclusive access to block
macro CSH$V_NOWAIT = 0,1,1,0 %;         ! do not wait for block on access interlock
macro CSH$V_NOREAD = 0,2,1,0 %;         ! do not read in block
macro CSH$V_NOBUFFER = 0,3,1,0 %;       ! obtain access to block but don't allocate
!  a buffer for it and don't read it
!  collision

!*** MODULE $PIODEF ***
!
!
!   rms overall status bit definitions
!
literal PIO$S_PIODEF = 1;
macro PIO$V_INHAST = 0,0,1,0 %;         ! set if asts implicitly inhibited
!   if reset by disabled ast, ast must be re-
!  enabled
macro PIO$V_EOD = 0,1,1,0 %;            ! set if searching for 'eod' string on 'input'
macro PIO$V_SYNC1 = 0,2,1,0 %;          ! sync stalled operation using efn 27
```

```
macro PIO$V_SYNC2 = 0,3,1,0 %;          ! sync stalled operation using efn 28

!*** MODULE $FTLDEF ***
!
!       definitions for rms debug failure codes
!
!
!   the following codes are for temporary bug check tests, and are
!   internal to rms.  all of the codes are negative, implying that they
!   do not return to the caller, probably killing the process (if not
!   the entire system).
!
literal FTL$_SETPRTFAIL = -1;           ! set protection system service failed (rm0bufmgr)
literal FTL$_STKTOOBIG = -2;            ! stack too big for asb (rm0stall)
literal FTL$_BADIFAB = -3;              ! invalid ifab or irab (rm0fset,rm0conn,rm0rset,rm0prflnm)
literal FTL$_GTCHNFAIL = -4;            ! get channel system service failure (rm0prflnm)
literal FTL$_BADORGCASE = -5;           ! invalid orgcase value for dispatch (all rms$
!  level routines execept open and create)
literal FTL$_BADBDB = -6;               ! block not a bdb (rm0bufmgr)
literal FTL$_ASBALLFAIL = -7;           ! couldn't allocate an asb (rm0stall)
literal FTL$_BADASTPRM = -8;            ! ast parameter not a valid ifab/irab addr (rm0stall)
literal FTL$_CANTDOAST = -9;            ! couldn't redeclare ast (insf. mem.) (rm0stall)
literal FTL$_NOSTRUCT = -10;            ! rab or fab not same on ast (rm0stall)
literal FTL$_NOASB = -11;               ! asb not allocated or stream not busy on ast (rm0stall)
literal FTL$_NONXTBDB = -12;            ! no next bdb available (rm1seqxfr)
literal FTL$_BADBUFSIZ = -13;           ! disk buffer size not = 512 (rm1conn)
literal FTL$_ENQDEQFAIL = -14;          ! enq or deq service failed (rm0reclck)
literal FTL$_NOCURBDB = -15;            ! no current bdb before calling rm$release (rm0reclck)
literal FTL$_NOPARENT = -16;            ! no parent lock available for global buffer section lock (rm0share)
literal FTL$_DEALLERR = -17;            ! ifab deallocation attempted with other block(s)
!  still allocated (rms0close)
literal FTL$_IORNDN = -18;              ! i/o rundown inconsistency (either ifab or irab
!  table entries not zeroed) (rms0rndwn)
literal FTL$_XFERSIZE = -19;            ! size of requested transfer not equal to
!  or less than the current number of bytes
!  in use for the bdb (rm0cache)
literal FTL$_NOTLOCKED = -20;           ! bdb not locked and a keep lock request
!  was made on a release request.
literal FTL$_NODIDORFID = -21;          ! neither a fid nor a did was set upon exit from
!  rm$setdid (rms0erase)
literal FTL$_RELEASFAIL = -22;          ! release of non-dirty bdb failed (rm0xtnd23,rms0extend)
literal FTL$_NOLOCKBDB = -23;           ! no lock bdb found (rm0xtnd23)
literal FTL$_NONETWORK = -24;           ! network routine entered but no network support in rms
literal FTL$_LOCKFAILED = -25;          ! failed to lock prolog (rm2create)
literal FTL$_BADLEVEL = -26;            ! to search by id, structure level must be 0
literal FTL$_ASTDECERR = -27;           ! ast declaration for file sharing failed
literal FTL$_BADGBLCNT = -28;           ! Zero global buffer count found when not expected (rm1conn)
literal FTL$_ACCNTOVFLO = -29;          ! access count overflow (rm0share)
literal FTL$_BDBAVAIL = -30;            ! BDB was available and shouldn't have been.
literal FTL$_GBLNOLK = -31;             ! Record locking was not set with global buffers.
literal FTL$_LCKFND = -32;              ! A lock was found and we don't know what to do.
literal FTL$_NOBLB = -33;               ! No BLB was found and there should have been one.
literal FTL$_NOGBPB = -34;              ! No GBPB was found and should have been.
literal FTL$_NOLCLBUF = -35;            ! Should have found a local buffer.
literal FTL$_NORDNOTSET = -36;          ! NOREAD not set when NOBUFFER was.
literal FTL$_NOTGBPB = -37;             ! Found an illegit BDB.
literal FTL$_NOSFSB = -38;              ! No SFSB when allocating BLB.
```

```
literal FTL$_LOCKHELD = -39;           ! Attempted to return a BLB with lock_id neq 0
literal FTL$_RLSDRT = -40;             ! Dirty buffer found in releasall.
literal FTL$_BADBLB = -41;             ! Bad BLB found in blocking AST routine.
literal FTL$_BADOWNER = -42;           ! Owner field in BLB is bad in blocking AST routine.
literal FTL$_GETLKIFAIL = -43;         ! $GETLKIW failed in last chance (rms0lstch).
literal FTL$_BADEBKHBK = -44;          ! tried to store an invalid EBK/HBK (rm0share).

!*** 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.
!
literal BUG$_BADDFLTDIR = 1;           ! DEFAULT DIRECTORY STRING INVALID (RM0XPFN)

!*** MODULE $IDXDEF ***
!
!         IDX field definitions
!
!         index descriptor definition
!
!         An index descriptor block exists for each key of reference in use.
!         they are not necessarily contiguous in memory.
!
literal IDX$C_BID = 15;                !  id for index descriptor block
literal IDX$M_DUPKEYS = 1;
literal IDX$M_CHGKEYS = 2;
literal IDX$M_NULKEYS = 4;
literal IDX$M_IDX_COMPR = 8;
literal IDX$M_INITIDX = 16;
literal IDX$M_KEY_COMPR = 64;
literal IDX$M_REC_COMPR = 128;
literal IDX$C_STRING = 0;              ! string data type
literal IDX$C_SGNWORD = 1;             ! signed binary word
literal IDX$C_UNSGNWORD = 2;           ! unsigned binary word
literal IDX$C_SGNLONG = 3;             ! signed binary long word
literal IDX$C_UNSGNLONG = 4;           ! unsigned binary long word
literal IDX$C_PACKED = 5;              ! packed decimal
literal IDX$C_SGNQUAD = 6;             ! signed binary quadword
literal IDX$C_UNSGNQUAD = 7;           ! unsigned binary quadword
literal IDX$C_V2_BKT = 0;              ! Prologue two bucket
literal IDX$C_CMPIDX = 1;              ! Prologue 3, index keys are compressed
literal IDX$C_NCMPIDX = 2;             ! Prologue 3, index keys are not compressed
literal IDX$C_CMPCMP = 3;              ! Prologue 3, primary key is compressed, data
!    is compressed
!  Prologue 3, SIDR key is compressed
literal IDX$C_CMPNCMP = 4;             ! Prologue 3, primary key is compressed,
!    data is not compressed
literal IDX$C_NCMPCMP = 5;             ! Prologue 3, primary key is not compressed
!    data is compressed
literal IDX$C_NCMPNCMP = 6;            ! Prologue 3, primary key is not compressed
!    data is not compressed
!  Prologue 3, SIDR key is compressed
literal IDX$K_FIXED_BLN = 44;
literal IDX$C_FIXED_BLN = 44;
!
```

```
!   the following is the length of the fixed part of the index descriptor
!
!
!   the following is repeated for each key segment
!
literal IDX$S_IDXDEF = 48;
macro IDX$L_IDXFL = 0,0,32,0 %;             ! forward link to next index descriptor
macro IDX$B_BID = 8,0,8,0 %;                ! block id
macro IDX$B_BLN = 9,0,8,0 %;                ! length of block
macro IDX$L_VBN = 10,0,32,0 %;              ! VBN where the descriptor came from
macro IDX$W_OFFSET = 14,0,16,0 %;           ! Offset into the block (VBN) of the descriptor
macro IDX$B_DESC_NO = 16,0,8,0 %;           ! Descriptor number (index into update buffer)
macro IDX$B_IANUM = 18,0,8,0 %;             ! area number for index buckets
macro IDX$B_LANUM = 19,0,8,0 %;             ! area number for lower index buckets
macro IDX$B_DANUM = 20,0,8,0 %;             ! area number for data buckets
macro IDX$B_ROOTLEV = 21,0,8,0 %;           ! level of root
macro IDX$B_IDXBKTSZ = 22,0,8,0 %;          ! size of index bucket in vbn's
macro IDX$B_DATBKTSZ = 23,0,8,0 %;          ! size of data bucket in vbn's
macro IDX$L_ROOTVBN = 24,0,32,0 %;          ! start vbn of root bucket
macro IDX$B_FLAGS = 28,0,8,0 %;             ! index/key flags
macro IDX$V_DUPKEYS = 28,0,1,0 %;           ! duplicate keys allowed
macro IDX$V_CHGKEYS = 28,1,1,0 %;           ! keys can change values
macro IDX$V_NULKEYS = 28,2,1,0 %;           ! null key value allowed
macro IDX$V_IDX_COMPR = 28,3,1,0 %;         ! index is compressed
macro IDX$V_INITIDX = 28,4,1,0 %;           ! index is not initialized
macro IDX$V_KEY_COMPR = 28,6,1,0 %;         ! key has been compressed
macro IDX$V_REC_COMPR = 28,7,1,0 %;         ! data record is in compressed form
macro IDX$B_DATATYPE = 29,0,8,0 %;          ! data type of key field
macro IDX$B_SEGMENTS = 30,0,8,0 %;          ! number of key field segments
macro IDX$B_NULLCHAR = 31,0,8,0 %;          ! null character
macro IDX$B_KEYSZ = 32,0,8,0 %;             ! total key size
macro IDX$B_KEYREF = 33,0,8,0 %;            ! key of reference(0-primary)
macro IDX$W_MINRECSZ = 34,0,16,0 %;         ! minimum record size
macro IDX$W_IDXFILL = 36,0,16,0 %;          ! index fill
macro IDX$W_DATFILL = 38,0,16,0 %;          ! data fill
macro IDX$B_IDXBKTYP = 40,0,8,0 %;          ! PLG3 - type of index bucket and SIDR bucket
macro IDX$B_DATBKTYP = 41,0,8,0 %;          ! PLG3 - type of primary data bucket
macro IDX$W_POSITION = 44,0,16,0 %;         ! key segment position
macro IDX$B_SIZE = 46,0,8,0 %;              ! key segment size (plg 3)
macro IDX$B_TYPE = 47,0,8,0 %;              ! key segment datatype (plg 3)

!*** MODULE $UPDDEF ***
!
!   update buffer flags
!
literal UPD$M_INS_NEW = 1;
literal UPD$M_OLD_DEL = 2;
literal UPD$S_UPDDEF = 1;
macro UPD$B_FLAGS = 0,0,8,0 %;
macro UPD$V_INS_NEW = 0,0,1,0 %;            ! alternate key to be inserted from record buffer
macro UPD$V_OLD_DEL = 0,1,1,0 %;            ! delete this key value using old record

!*** 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  ***
!
literal GBH$C_BID = 17;                    !  Block ID code for GBH
literal GBH$M_CACHE_IN = 1;
literal GBH$M_CACHE_OUT = 2;
literal GBH$M_RLS_IN = 4;
literal GBH$M_RLS_OUT = 8;
literal GBH$M_QIO_START = 16;
literal GBH$M_QIO_DONE = 32;
literal GBH$M_STALL = 64;
literal GBH$M_THREADGO = 128;
literal GBH$M_BLB_ENQ = 256;
literal GBH$M_BLB_GRANT = 512;
literal GBH$M_BLB_DEQ = 1024;
literal GBH$M_BLB_BLOCK = 2048;
literal GBH$M_F1 = 4096;
literal GBH$M_F2 = 8192;
literal GBH$M_F3 = 16384;
literal GBH$M_F4 = 32768;
literal GBH$K_BLN = 88;                    !  Length of global buffer header structure
literal GBH$C_BLN = 88;                    !  Length of global buffer header structure
literal GBH$S_GBHDEF = 88;
macro GBH$L_GBD_FLNK = 0,0,32,0 %;         !  Self relative queue header for GBD's
macro GBH$L_GBD_BLNK = 4,0,32,0 %;
macro GBH$B_BID = 8,0,8,0 %;               !  Block ID
macro GBH$B_BLN = 9,0,8,0 %;               !  Length of GBH in longwords
macro GBH$W_TRC_FLGS = 10,0,16,0 %;        !  Trace flags (set to trace given function)
macro GBH$V_CACHE_IN = 10,0,1,0 %;         !  Cache inputs
macro GBH$V_CACHE_OUT = 10,1,1,0 %;        !  Cache outputs
macro GBH$V_RLS_IN = 10,2,1,0 %;           !  Release inputs
macro GBH$V_RLS_OUT = 10,3,1,0 %;          !  Release outputs
macro GBH$V_QIO_START = 10,4,1,0 %;        !  Qio inputs
macro GBH$V_QIO_DONE = 10,5,1,0 %;         !  Qio outputs
macro GBH$V_STALL = 10,6,1,0 %;            !  Stall inputs
macro GBH$V_THREADGO = 10,7,1,0 %;         !  Stall outputs
macro GBH$V_BLB_ENQ = 10,8,1,0 %;          !  Bucket lock ENQ inputs
macro GBH$V_BLB_GRANT = 10,9,1,0 %;        !  Bucket lock grant status
macro GBH$V_BLB_DEQ = 10,10,1,0 %;         !  Bucket lock DEQ request
macro GBH$V_BLB_BLOCK = 10,11,1,0 %;       !  Blocking AST received
macro GBH$V_F1 = 10,12,1,0 %;
macro GBH$V_F2 = 10,13,1,0 %;
macro GBH$V_F3 = 10,14,1,0 %;
macro GBH$V_F4 = 10,15,1,0 %;
macro GBH$L_HI_VBN = 12,0,32,0 %;          !  Highest possible VBN value (FFFFFFFF).
macro GBH$L_GS_SIZE = 16,0,32,0 %;         !  Size of total section in bytes.
macro GBH$L_LOCK_ID = 20,0,32,0 %;         !  Lock ID of system file lock.
macro GBH$L_GS_LOCK_ID = 24,0,32,0 %;      !  Lock ID of system global section lock.
macro GBH$L_USECNT = 28,0,32,0 %;          !  Accessor count for section.
macro GBH$L_TRC_FLNK = 32,0,32,0 %;        !  Trace blocks forward link
macro GBH$L_TRC_BLNK = 36,0,32,0 %;        !  Trace blocks back link
macro GBH$L_GBD_START = 40,0,32,0 %;       !  Offset to first GBD.
macro GBH$L_GBD_END = 44,0,32,0 %;         !  Offset to last GBD.
macro GBH$L_GBD_NEXT = 48,0,32,0 %;        !  Offset to next cache victim GBD.
macro GBH$L_SCAN_NUM = 52,0,32,0 %;        !  Number of GBD's to scan for victim.
```

```
!
!   Global buffer statistics section
!
macro GBH$L_HIT = 56,0,32,0 %;            !  Buffer found in global cache
macro GBH$L_MISS = 60,0,32,0 %;           !  Buffer not found in global cache
macro GBH$L_READ = 64,0,32,0 %;           !  Buffer read from disk into cache
macro GBH$L_WRITE = 68,0,32,0 %;          !  Buffer written from cache to disk
macro GBH$L_DFW_WRITE = 72,0,32,0 %;      !  Deferred writeback from cache to disk
macro GBH$L_CROSS_HIT = 76,0,32,0 %;      !  Cross process hit count.
macro GBH$L_OUTBUFQUO = 80,0,32,0 %;      !  Count of times GBLBUFQUO limit was hit.
macro GBH$L_FILL_1 = 84,0,32,0 %;         !  Force quadword alignment

!*** MODULE $TRCDEF ***
!
!          TRC field definitions
!
!          Trace block structure (TRC)
!
!          Tracing saves at specific points in the RMS code for debugging and
!          algorithm analysis purposes.
!
!          ***  WARNING - THIS STRUCTURE MUST BE QUADWORD ALIGNED  ***
!
literal TRC$C_BID = 18;                   !  Trace block code
literal TRC$K_BLN = 64;                   !  NOTE: should be quadwords multiple to
literal TRC$C_BLN = 64;                   !  NOTE: should be quadwords multiple to
literal TRC$S_TRCDEF = 64;
macro TRC$L_FLNK = 0,0,32,0 %;            !  Trace block forward link
macro TRC$L_BLNK = 4,0,32,0 %;            !  Trace block back link
macro TRC$B_BID = 8,0,8,0 %;              !  Block ID
macro TRC$B_BLN = 9,0,8,0 %;              !  Length of block in longwords
macro TRC$W_FUNCTION = 10,0,16,0 %;       !  Function code (see GBH definitions)
macro TRC$L_STRUCTURE = 12,0,32,0 %;      !  Ifab/irab address.
macro TRC$W_PID = 16,0,16,0 %;            !  Process ID
macro TRC$W_SEQNUM = 18,0,16,0 %;         !  Sequence number.
macro TRC$L_VBN = 20,0,32,0 %;            !  VBN requested.
macro TRC$L_RETURN1 = 24,0,32,0 %;        !  Address of caller.
macro TRC$L_RETURN2 = 28,0,32,0 %;        !  Caller's caller.
macro TRC$L_ARGS = 32,0,0,0 %;
literal TRC$S_ARGS = 32;                  !  Function specific arguments
macro TRC$L_ARG_FLG = 32,0,32,0 %;        !  Argument flags (R3).
macro TRC$L_BDB_ADDR = 36,0,32,0 %;       !  BDB address.
macro TRC$W_BDB_USERS = 40,0,16,0 %;      !  Use count from BDB.
macro TRC$W_BDB_BUFF = 42,0,16,0 %;       !  BDB buffer ID.
macro TRC$B_BDB_CACHE = 44,0,8,0 %;       !  BDB cache value.
macro TRC$B_BDB_FLAGS = 45,0,8,0 %;       !  Status flags from BDB.
macro TRC$L_BDB_SEQ = 46,0,32,0 %;        !  Sequence number from BDB.
macro TRC$B_BLB_MODE = 50,0,8,0 %;        !  Mode held in BLB.
macro TRC$B_BLB_FLAGS = 51,0,8,0 %;       !  Flags from BLB.
macro TRC$L_BLB_ADDR = 52,0,32,0 %;       !  Address of BLB.
macro TRC$L_BLB_LOCK = 56,0,32,0 %;       !  Lock ID from BLB.
macro TRC$L_BLB_SEQ = 60,0,32,0 %;        !  Sequence number from BLB.
!  maintain quad alignment on header

!*** MODULE $GBDDEF ***
!
!          GBD structure definitions
```

```
0       !
0       !       Global Buffer Descriptor (GBD)
0       !
0       !       There is a single GBD for every buffer in a global buffer
0       !       section (used only with shared files).  The GBD's themselves
0       !       are in the section also and linked from a queue header in
0       !       the Global Buffer Header (GBH).
0       !
0       !       ***  WARNING - THIS STRUCTURE MUST BE QUADWORD ALIGNED  ***
0       !
0       literal GBD$C_BID = 19;                 !  Block ID code for GBD
0       literal GBD$M_VALID = 1;
0       literal GBD$K_BLN = 40;                 !  Length of Global Buffer Descriptor structure.
0       literal GBD$C_BLN = 40;                 !  Length of Global Buffer Descriptor structure.
0       literal GBD$S_GBDDEF = 40;
0       macro GBD$L_FLINK = 0,0,32,0 %;         !  Forward link - Note: This is a self relative queue
0       macro GBD$L_BLINK = 4,0,32,0 %;         !  Back link
0       macro GBD$B_BID = 8,0,8,0 %;            !  Block ID
0       macro GBD$B_BLN = 9,0,8,0 %;            !  Block length of GBD
0       macro GBD$B_FLAGS = 10,0,8,0 %;         !  Buffer status flags
0       macro GBD$V_VALID = 10,0,1,0 %;         !  Buffer is valid.
0       macro GBD$B_CACHE_VAL = 11,0,8,0 %;     !  Cache value of this bucket
0       macro GBD$L_VBN = 12,0,32,0 %;          !  VBN of bucket the buffer describes
0       macro GBD$L_VBNSEQNUM = 16,0,32,0 %;    !  VBN sequence number validity check
0       macro GBD$L_LOCK_ID = 20,0,32,0 %;      !  Lock ID of system lock.
0       macro GBD$W_NUMB = 24,0,16,0 %;         !  Number of bytes in use
0       macro GBD$W_SIZE = 26,0,16,0 %;         !  Size of buffer in bytes
0       macro GBD$L_REL_ADDR = 28,0,32,0 %;     !  Address of buffer relative to GBH
0       macro GBD$W_USECNT = 32,0,16,0 %;       !  Accessor count for bucket
0       macro GBD$B_REHIT_RD = 34,0,8,0 %;      !  Rehit by same process count.
0       macro GBD$B_REHIT_LK = 35,0,8,0 %;      !  Rehit by same locker process.
0
0       !*** MODULE $BLBDEF ***
0       !
0       !       BLB field definitions
0       !
0       !       Bucket Lock Block (BLB)
0       !
0       !       The BLB contains the argument list for the SYS$ENQ system service
0       !       as well a pointer to the BDB it relates to and other status.
0       !
0       literal BLB$C_BID = 16;                 !  BLB code
0       literal BLB$M_LOCK = 1;
0       literal BLB$M_NOWAIT = 2;
0       literal BLB$M_NOREAD = 4;
0       literal BLB$M_NOBUFFER = 8;
0       literal BLB$M_IOLOCK = 16;
0       literal BLB$M_DFW = 32;
0       literal BLB$M_WRITEBACK = 64;
0       literal BLB$K_BLN = 56;                 !  Length of BLB
0       literal BLB$C_BLN = 56;                 !  Length of BLB
0       literal BLB$S_BLBDEF = 56;
0       macro BLB$L_FLNK = 0,0,32,0 %;          !  Link to next BLB
0       macro BLB$L_BLNK = 4,0,32,0 %;          !  Back link
0       macro BLB$B_BID = 8,0,8,0 %;            !  Block ID
0       macro BLB$B_BLN = 9,0,8,0 %;            !  Block length
0       macro BLB$B_BLBFLGS = 10,0,8,0 %;       !  Control flags for BLB
```

```
macro BLB$V_LOCK = 10,0,1,0 %;           ! Corresponds to CSH$V_LOCK
macro BLB$V_NOWAIT = 10,1,1,0 %;         ! Same as CSH$V_NOWAIT
macro BLB$V_NOREAD = 10,2,1,0 %;         ! Same as CSH$V_NOREAD
macro BLB$V_NOBUFFER = 10,3,1,0 %;       ! Same as CSH$V_NOBUFFER
macro BLB$V_IOLOCK = 10,4,1,0 %;         ! Lock mode for read/write
macro BLB$V_DFW = 10,5,1,0 %;            ! This is lock for deferred write buffer
macro BLB$V_WRITEBACK = 10,6,1,0 %;      ! The associated buffer must be written back
macro BLB$B_MODEHELD = 11,0,8,0 %;       ! Mode of current lock held.
macro BLB$L_BDB_ADDR = 12,0,32,0 %;      ! BDB for which this lock is held
macro BLB$L_OWNER = 16,0,32,0 %;         ! Address of stream owning this lock
macro BLB$L_VBN = 20,0,32,0 %;           ! VBN of bucket lock (resource name)
macro BLB$L_RESDSC = 24,0,0,0 %;
literal BLB$S_RESDSC = 8;                ! Resource name descriptor
macro BLB$W_LKSTS = 32,0,16,0 %;         ! Lock status word
macro BLB$L_LOCK_ID = 36,0,32,0 %;       ! Lock ID
macro BLB$L_VALBLK = 40,0,0,0 %;
literal BLB$S_VALBLK = 16;               ! Lock value block
macro BLB$L_VALSEQNO = 40,0,32,0 %;      ! Sequence number part of value block

!*** MODULE $RJBDEF ***
!
!        RJB Definitions
!
!        RMS Journaling Block (RJB)
!
!        This block contains the necessary control information to keep
!        track of the state of journaling on this file
!
literal RJB$C_BID = 22;
literal RJB$K_BLN = 12;                ! Length of RJB
literal RJB$C_BLN = 12;                ! Length of RJB
literal RJB$M_RU = 1;
literal RJB$M_BI = 2;
literal RJB$M_AI = 4;
literal RJB$M_AT = 8;
literal RJB$M_OPEN = 16;
literal RJB$S_RJBDEF = 12;
macro RJB$Q_CHAN = 0,0,0,0 %;
literal RJB$S_CHAN = 8;                ! Channel Block
macro RJB$W_RUCHAN = 0,0,16,0 %;       !  channel for recovery unit journal
macro RJB$W_BICHAN = 2,0,16,0 %;       !  channel for before image journal
macro RJB$W_AICHAN = 4,0,16,0 %;       !  channel for after image journal
macro RJB$W_ATCHAN = 6,0,16,0 %;       !  channel for audit trail journal
macro RJB$B_BID = 8,0,8,0 %;           ! Block Id
macro RJB$B_BLN = 9,0,8,0 %;           ! Block Length
macro RJB$W_FLAGS = 10,0,16,0 %;       ! Flags word
macro RJB$V_RU = 10,0,1,0 %;           ! Set to indicate RU channel open
macro RJB$V_BI = 10,1,1,0 %;           ! Set to indicate BI channel open
macro RJB$V_AI = 10,2,1,0 %;           ! Set to indicate AI channel open
macro RJB$V_AT = 10,3,1,0 %;           ! Set to indicate AT channel open
macro RJB$V_OPEN = 10,4,1,0 %;         ! Indicates $OPEN mapping entry written

!*** 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.
!
literal MJB$C_BID = 24;
literal MJB$M_INIT = 1;
literal MJB$M_FORCE = 2;
literal MJB$M_FILE = 4;
literal MJB$M_SYNCH_SHARE = 8;
literal MJB$K_BLN = 32;
literal MJB$C_BLN = 32;
literal MJB$S_MJBDEF = 32;
macro MJB$B_BID = 8,0,8,0 %;              !  block id
macro MJB$B_BLN = 9,0,8,0 %;              !  block length in longwords
macro MJB$W_FLAGS = 10,0,16,0 %;          !  flags
macro MJB$V_INIT = 10,0,1,0 %;            !  set if RJR overhead is initialized
macro MJB$V_FORCE = 10,1,1,0 %;           !  set if RJR is to be written thru to journal
macro MJB$V_FILE = 10,2,1,0 %;            !  set if file operation to journal
macro MJB$V_SYNCH_SHARE = 10,3,1,0 %;     !  set if file lock can't be released during
!  STALL
!  and not buffered by CJF (input to WRITE_MJB)
macro MJB$B_JNL = 12,0,8,0 %;             !  set to CJF$_"jnl type" as input to WRITE_MJB
macro MJB$Q_DESC = 16,0,0,0 %;
literal MJB$S_DESC = 8;                   !  RJR descriptor used in $WRITEJNL service
macro MJB$W_SIZE = 16,0,16,0 %;           !  size of RJR to write
macro MJB$L_POINTER = 20,0,32,0 %;        !  pointer to RJR
macro MJB$Q_IOSB = 24,0,0,0 %;
literal MJB$S_IOSB = 8;                   !  IOSB to use in $WRITEJNL
macro MJB$T_RJR = 32,0,0,0 %;             !  the journal record begins here
!
! ****************************************************************************
! *                                                                          *
! *  Copyright (c) 1982, 1983                                                *
! *  by DIGITAL Equipment Corporation, Maynard, Mass.                        *
! *                                                                          *
! *  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.                 *
! *                                                                          *
! ****************************************************************************
! *********************************************************************************************************************
!  Created 15-SEP-1984 22:54:20  by  VAX-11 SDL V2.0       Source: 15-SEP-1984 22:49:17 _$255$DUA28:[RMS.SRC]RMSFWAD
! *********************************************************************************************************************


!*** MODULE $FWADEF ***
! ++
```

```
!
!  Flags
!
!--
literal FWA$M_DUPOK = 1;
literal FWA$M_NOCOPY = 2;
literal FWA$M_SL_PASS = 4;
literal FWA$M_RLF_PASS = 8;
literal FWA$M_FNA_PASS = 16;
literal FWA$M_NAM_DVI = 32;
literal FWA$M_EXP_NODE = 64;
literal FWA$M_VERSION = 2048;
literal FWA$M_TYPE = 4096;
literal FWA$M_NAME = 8192;
literal FWA$M_DIR = 16384;
literal FWA$M_DEVICE = 32768;
literal FWA$M_EXP_VER = 65536;
literal FWA$M_EXP_TYPE = 131072;
literal FWA$M_EXP_NAME = 262144;
literal FWA$M_WC_VER = 524288;
literal FWA$M_WC_TYPE = 1048576;
literal FWA$M_WC_NAME = 2097152;
literal FWA$M_EXP_DIR = 4194304;
literal FWA$M_EXP_DEV = 8388608;
literal FWA$M_WILDCARD = 16777216;
literal FWA$M_NODE = 33554432;
literal FWA$M_QUOTED = 67108864;
literal FWA$M_GRPMBR = 134217728;
literal FWA$M_WILD_DIR = 268435456;
literal FWA$M_DIR_LVLS = -536870912;
literal FWA$C_ALL = 248;
!
!  constant for all flags that vary per parsing pass
!
literal FWA$C_ALLPASS = 25;
!++
!
!  Misc. Fields
!
!--
literal FWA$C_BID = 40;                   ! bid of fwa
literal FWA$C_MAXNODNAM = 6;              ! max node name size
literal FWA$C_MAXLNDNAM = 15;             ! max logical node name size
literal FWA$C_MAXNODLST = 127;            ! max node spec list size (concatenated node specs)
!
!          device name descriptor
!
literal FWA$C_MAXDEVICE = 255;            ! max device name size
literal FWA$C_MAXCDIR = 8;                ! max number of concealed directories
literal FWA$C_MAXSUBDIR = 7;              ! max number of sub directories
literal FWA$C_MAXDIRLEN = 255;            ! max size of total directory spec
!  should be:  top + subdir   39 * 8
!              dots between        7
!              delimiters          2
!              -------------------
!              total             321
!
```

```
0      !       The filename, filetype and fileversion descriptors MUST be contiguous
0      !
0      !         file name descriptor
0      !
0      literal FWA$C_MAXNAME = 39;              !  max file name size
0      literal FWA$C_MAXQUOTED = 255;           !  max quoted string size
0      !
0      !         file type descriptor
0      !
0      literal FWA$C_MAXTYPE = 39;              !  max file type size
0      !
0      !         file version number descriptor
0      !
0      literal FWA$C_MAXVER = 6;                !  maximum version
0      literal FWA$C_MAXRNS = 86;               !  max resultant name string size
0      literal FWA$C_STATBLK = 10;              !  define length of statistics block
0      literal FWA$K_BLN_FWA = 500;             !  length of fwa
0      literal FWA$C_BLN_FWA = 500;             !  length of fwa
0      literal FWA$C_MAXSUBNOD = 7;             !  max number of secondary (sub) node specs
0      ! ++
0      !
0      !   buffers for parsed filename elements
0      !
0      ! --
0      literal FWA$C_FIBLEN = 76;               !  fib buffer size
0      literal FWA$C_DIRBUFSIZ = 39;            !  size of each directory buffer
0      !
0      !   rooted directory name buffers
0      !
0      !   NOTE:  These buffers must be contiguous
0      !
0      literal FWA$K_BLN_BUF = 2364;            !  length of fwa and buffers
0      literal FWA$C_BLN_BUF = 2364;            !  length of fwa and buffers
0      literal FWA$K_BLN = 2364;                !  length of fwa and buffers
0      literal FWA$C_BLN = 2364;                !  length of fwa and buffers
0      literal FWA$S_FWADEF = 2364;
0      macro FWA$Q_FLAGS = 0,0,0,0 %;
0      literal FWA$S_FLAGS = 8;                 !  various parse status flags
0      macro FWA$B_PASSFLGS = 0,0,8,0 %;        !  flags for pass only
0      macro FWA$B_FLDFLGS = 1,0,8,0 %;         !  flags for fields seen
0      macro FWA$B_WILDFLGS = 2,0,8,0 %;        !  flags for wild cards
0      macro FWA$B_PARSEFLGS = 3,0,8,0 %;       !  flags for parse results
0      macro FWA$B_DIRFLGS = 4,0,8,0 %;         !  flags primarily for directory spec
0      macro FWA$B_DIRWCFLGS = 5,0,8,0 %;       !  directory wild flags
0      macro FWA$B_LNFLGS = 6,0,8,0 %;          !  logical name flag byte
0      macro FWA$B_SLFLGS = 7,0,8,0 %;          !  search list + rooted directory flags
0      !
0      !  flags for pass
0      !
0      macro FWA$V_DUPOK = 0,0,1,0 %;           !  discard duplicate element
0      macro FWA$V_NOCOPY = 0,1,1,0 %;          !  do not copy this field
0      macro FWA$V_SL_PASS = 0,2,1,0 %;         !  search list pass
0      macro FWA$V_RLF_PASS = 0,3,1,0 %;        !  set if applying related file defaults
0      macro FWA$V_FNA_PASS = 0,4,1,0 %;        !  set if primary name string parse pass
0      macro FWA$V_NAM_DVI = 0,5,1,0 %;         !  set if open by name block
0      macro FWA$V_EXP_NODE = 0,6,1,0 %;        !  explicit node has been seen, null or normal
0      !
```

```
!  flags for fields seen
!
macro FWA$V_VERSION = 0,11,1,0 %;          !  set if version seen
macro FWA$V_TYPE = 0,12,1,0 %;             !  set if type seen
macro FWA$V_NAME = 0,13,1,0 %;             !  set if name seen
macro FWA$V_DIR = 0,14,1,0 %;              !  set if directory spec seen
macro FWA$V_DEVICE = 0,15,1,0 %;           !  set if device seen
!
!  flags for wild cards
!
macro FWA$V_EXP_VER = 0,16,1,0 %;          !  set if explicit version
macro FWA$V_EXP_TYPE = 0,17,1,0 %;         !  set if explicit type
macro FWA$V_EXP_NAME = 0,18,1,0 %;         !  set if explicit name
macro FWA$V_WC_VER = 0,19,1,0 %;           !  set if wildcard (*) version
macro FWA$V_WC_TYPE = 0,20,1,0 %;          !  " type
macro FWA$V_WC_NAME = 0,21,1,0 %;          !  " name
macro FWA$V_EXP_DIR = 0,22,1,0 %;          !  set if explicit directory
macro FWA$V_EXP_DEV = 0,23,1,0 %;          !  set if explicit device
!
!  flags for parse results
!
macro FWA$V_WILDCARD = 0,24,1,0 %;         !  set if any wildcard seen
macro FWA$V_NODE = 0,25,1,0 %;             !  set if node name seen
macro FWA$V_QUOTED = 0,26,1,0 %;           !  set is quoted string seen
macro FWA$V_GRPMBR = 0,27,1,0 %;           !  set if directory in [grp,mbr] format
macro FWA$V_WILD_DIR = 0,28,1,0 %;         !  inclusive or of directory wild cards
macro FWA$V_DIR_LVLS = 0,29,3,0 %;
literal FWA$S_DIR_LVLS = 3;                !  ! of directory sublevels (0 = ufd only)
!   (valid only if node set and no fldflgs)
!
!  flags primarily for directory spec
!
macro FWA$V_DIR1 = 4,0,1,0 %;              !  ufd level directory or group seen
macro FWA$V_DIR2 = 4,1,1,0 %;              !  sfd level 1 directory or member seen
!
!  directory wild flags
!
macro FWA$V_WILD_UFD = 4,8,1,0 %;          !  the dir1 spec was a wild card
macro FWA$V_WILD_SFD1 = 4,9,1,0 %;         !  the dir2 spec was a wild card
macro FWA$V_WILD_GRP = 4,8,1,0 %;          !  the grp spec contained a wild card
macro FWA$V_WILD_MBR = 4,9,1,0 %;          !  the mbr spec contained a wild card
!
!  logical name flag and miscellaneous byte
!
macro FWA$V_LOGNAME = 4,16,1,0 %;          !  a logical name has been seen this pass
!   (note: this byte is saved as context
!   when processing [.dir-list] format)
macro FWA$V_OBJTYPE = 4,17,1,0 %;          !  set if quoted string is of the
!   "objecttype=..." form
!   (valid only if quoted set)
macro FWA$V_NETSTR = 4,18,1,0 %;           !  set if quoted string is of the
!   "objecttype=taskname/..." form
!   (valid only if quoted and objtype set)
macro FWA$V_DEV_UNDER = 4,19,1,0 %;        !  device name was prefixed with an underscore
macro FWA$V_FILEFOUND = 4,20,1,0 %;        !  true if at least one file found by search
macro FWA$V_REMRESULT = 4,21,1,0 %;        !  use resultant string returned by fal
macro FWA$V_SYNTAX_CHK = 4,22,1,0 %;       !  syntax-only checking is requested (NAM$V_SYNCHK set)
```

```
        !
        !   search list and rooted directory flag byte
        !
        macro FWA$V_SLPRESENT = 4,24,1,0 %;        !  search list present
        macro FWA$V_CONCEAL_DEV = 4,25,1,0 %;      !  concealed device present
        macro FWA$V_ROOT_DIR = 4,26,1,0 %;         !  root directory present
        macro FWA$V_DFLT_MFD = 4,27,1,0 %;         !  default MFD string inserted, due to [-]
        macro FWA$V_EXP_ROOT = 4,28,1,0 %;         !  explicit root directory
        !
        !   Value for all filename elements except node
        !
        macro FWA$B_BID = 8,0,8,0 %;               !  bid
        macro FWA$B_BLN = 9,0,8,0 %;               !  bln
        macro FWA$B_DIRTERM = 10,0,8,0 %;          !  directory spec terminator (']' or '>')
        macro FWA$B_ROOTERM = 11,0,8,0 %;          !  root directory spec terminator (']' or '>')
        macro FWA$L_ESCSTRING = 12,0,32,0 %;       !  escape equivalence string
        macro FWA$B_ESCFLG = 12,0,8,0 %;           !  set to the char <esc> if an escape string
        !  seen, zero otherwise
        macro FWA$B_ESCTYP = 13,0,8,0 %;           !  escape 'type' byte
        macro FWA$W_ESCIFI = 14,0,16,0 %;          !  escape ifi value
        macro FWA$Q_FIB = 16,0,0,0 %;
        literal FWA$S_FIB = 8;                     !  fib descriptor
        macro FWA$L_DEVBUFSIZ = 24,0,32,0 %;       !  device buffer size
        macro FWA$L_DEV_CLASS = 28,0,32,0 %;       !  device class
        macro FWA$L_RECSIZ = 32,0,32,0 %;          !  blocked record size
        macro FWA$L_UNIT = 36,0,32,0 %;            !  device unit number
        macro FWA$L_UIC = 40,0,32,0 %;             !  file owner uic
        macro FWA$W_PRO = 44,0,16,0 %;             !  file protection word
        macro FWA$B_DIRLEN = 46,0,8,0 %;           !  overall directory spec length
        macro FWA$B_SUBNODCNT = 47,0,8,0 %;        !  number of secondary (sub) node specs found
        macro FWA$L_DIRBDB = 48,0,32,0 %;          !  address of directory file bdb
        macro FWA$L_LOOKUP = 52,0,32,0 %;          !  address of new directory cache node
        macro FWA$L_DEVNODADR = 56,0,32,0 %;       !  address of device directory cache node
        macro FWA$Q_DIR = 60,0,0,0 %;
        literal FWA$S_DIR = 8;                     !  directory name scratch buffer
        macro FWA$L_UCHAR = 68,0,32,0 %;           !  user characteristics longword
        macro FWA$W_UCHAR = 68,0,16,0 %;
        macro FWA$L_FWA_PTR = 72,0,32,0 %;         !  pointer to second fwa if any ($RENAME)
        macro FWA$L_SWB_PTR = 76,0,32,0 %;         !  pointer to swb
        macro FWA$L_BUF_PTR = 80,0,32,0 %;         !  address of temporary buffer
        macro FWA$L_IMPURE_AREA = 84,0,32,0 %;     !  saved R11 (rm$xpfn only)
        macro FWA$L_ATR_WORK = 88,0,32,0 %;        !  pointer to work area for ACP attributes
        !    (zero    one not currently allocated)
        !  ++
        !
        !   Logical name and search list fields
        !
        !  --
        !
        !   Item list block for logical name services
        !
        macro FWA$T_ITMLST = 92,0,0,0 %;
        literal FWA$S_ITMLST = 64;                 !  logical name item list
        macro FWA$T_ITM_INDEX = 92,0,0,0 %;
        literal FWA$S_ITM_INDEX = 12;              !  index
        macro FWA$T_ITM_ATTR = 104,0,0,0 %;
        literal FWA$S_ITM_ATTR = 12;               !  attributes
```

```
macro FWA$T_ITM_STRING = 116,0,0,0 %;
literal FWA$S_ITM_STRING = 12;               !  string
macro FWA$T_ITM_MAX_INDEX = 128,0,0,0 %;
literal FWA$S_ITM_MAX_INDEX = 12;            !  max index
macro FWA$L_ITM_END = 140,0,32,0 %;          !  terminating longword
!
!  Logical name translation fields
!
macro FWA$B_BUFFLG = 156,0,8,0 %;            !  flag for which translation buffer is in use
!  (0 = buf2 in use, -1 = buf1 in use)
macro FWA$B_XLTMODE = 157,0,8,0 %;           !  mode of translation on input to $TRNLNM
!  mode of equivalence string on output from $TRNLNM
macro FWA$W_XLTSIZ = 158,0,16,0 %;           !  length of equivalence string
macro FWA$L_XLTBUFF1 = 160,0,32,0 %;         !  primary translation buffer descriptor
macro FWA$L_XLTBUFF2 = 164,0,32,0 %;         !  secondary translation buffer descriptor
!
!  SLBH and SLB pointers
!
macro FWA$L_SLBH_PTR = 168,0,32,0 %;         !  current SLB list
macro FWA$L_SLB_PTR = 172,0,32,0 %;          !  current SLB list
macro FWA$L_SLBH_FLINK = 176,0,32,0 %;       !  SLBH que fwd link
macro FWA$L_SLBH_BLINK = 180,0,32,0 %;       !  SLBH que back link
!
!  Fake SLB -   NOTE:       This MUST be the size of SLB$C_BLN
!                          The field FWA$B_LEVEL must be at the same offset
!                          as SLB$Q_LEVEL would be.  (It sounds like a real
!                          hack but it works very nicely)
!
macro FWA$T_SLB = 184,0,0,0 %;
literal FWA$S_SLB = 24;                      !  space for SLB$C_BLN
macro FWA$B_LEVEL = 195,0,8,0 %;             !  recursion level
!
!  Logical name descriptor
!
macro FWA$Q_LOGNAM = 208,0,0,0 %;
literal FWA$S_LOGNAM = 8;                    !  logical name descriptor
!++
!
!   descriptors for parsed filename elements
!
!
!         The descriptors are defined as:
!
!         ------------------------------------------------
!         |       flags          |             length    |
!         |-----------------------------------------------|
!         |                  address                      |
!         ------------------------------------------------
!
!
!
!         The flags are defined by FSCB$V_flag in $FSCBDEF
!
!--
macro FWA$Q_NODE = 216,0,0,0 %;
literal FWA$S_NODE = 8;                      !  node name (actually node spec list) descriptor
!  (the associated buffer is fwa$t_nodebuf)
macro FWA$Q_DEVICE = 224,0,0,0 %;
```

```
0   literal FWA$S_DEVICE = 8;                   ! device name descriptor
0   macro FWA$Q_CONCEAL_DEV = 232,0,0,0 %;
0   literal FWA$S_CONCEAL_DEV = 8;              ! concealed device descriptor
0   !
0   !       directory name descriptors   NOTE:  The two sets of directory
0   !                                           descriptors must be contigous
0   !                                           or RM$SETDID will break
0   !
0   macro FWA$Q_CDIR1 = 240,0,0,0 %;
0   literal FWA$S_CDIR1 = 8;                    ! concealed top directory descriptors
0   macro FWA$Q_CDIR2 = 248,0,0,0 %;
0   literal FWA$S_CDIR2 = 8;                    ! concealed subdirectory 1
0   macro FWA$Q_CDIR3 = 256,0,0,0 %;
0   literal FWA$S_CDIR3 = 8;                    ! " " 2
0   macro FWA$Q_CDIR4 = 264,0,0,0 %;
0   literal FWA$S_CDIR4 = 8;                    ! " " 3
0   macro FWA$Q_CDIR5 = 272,0,0,0 %;
0   literal FWA$S_CDIR5 = 8;                    ! " " 4
0   macro FWA$Q_CDIR6 = 280,0,0,0 %;
0   literal FWA$S_CDIR6 = 8;                    ! " " 5
0   macro FWA$Q_CDIR7 = 288,0,0,0 %;
0   literal FWA$S_CDIR7 = 8;                    ! " " 6
0   macro FWA$Q_CDIR8 = 296,0,0,0 %;
0   literal FWA$S_CDIR8 = 8;                    ! " " 7
0   macro FWA$Q_DIR1 = 304,0,0,0 %;
0   literal FWA$S_DIR1 = 8;                     ! top level directory descriptors
0   macro FWA$Q_DIR2 = 312,0,0,0 %;
0   literal FWA$S_DIR2 = 8;                     ! subdirectory 1
0   macro FWA$Q_DIR3 = 320,0,0,0 %;
0   literal FWA$S_DIR3 = 8;                     ! " 2
0   macro FWA$Q_DIR4 = 328,0,0,0 %;
0   literal FWA$S_DIR4 = 8;                     ! " 3
0   macro FWA$Q_DIR5 = 336,0,0,0 %;
0   literal FWA$S_DIR5 = 8;                     ! " 4
0   macro FWA$Q_DIR6 = 344,0,0,0 %;
0   literal FWA$S_DIR6 = 8;                     ! " 5
0   macro FWA$Q_DIR7 = 352,0,0,0 %;
0   literal FWA$S_DIR7 = 8;                     ! " 6
0   macro FWA$Q_DIR8 = 360,0,0,0 %;
0   literal FWA$S_DIR8 = 8;                     ! " 7
0   macro FWA$Q_NAME = 368,0,0,0 %;
0   literal FWA$S_NAME = 8;                     ! file name descriptor
0   macro FWA$Q_QUOTED = 368,0,0,0 %;
0   literal FWA$S_QUOTED = 8;                   ! quoted string descriptor
0   macro FWA$Q_TYPE = 376,0,0,0 %;
0   literal FWA$S_TYPE = 8;                     ! file type descriptor
0   macro FWA$Q_VERSION = 384,0,0,0 %;
0   literal FWA$S_VERSION = 8;                  ! file version descriptor
0   macro FWA$Q_RNS = 392,0,0,0 %;
0   literal FWA$S_RNS = 8;                      ! resultant name string descriptor
0   macro FWA$Q_SHRFIL = 400,0,0,0 %;
0   literal FWA$S_SHRFIL = 8;                   ! shared file device descriptor (readable form)
0   macro FWA$Q_SHRFIL_LCK = 408,0,0,0 %;
0   literal FWA$S_SHRFIL_LCK = 8;               ! shared file device descriptor (unreadable form - used for lock name)
0   macro FWA$Q_AS_SHRFIL = 416,0,0,0 %;
0   literal FWA$S_AS_SHRFIL = 8;                ! secondary device descriptor (readable form)
0   macro FWA$T_STATBLK = 424,0,0,0 %;
```

```
0       literal FWA$S_STATBLK = 10;                    !
0       macro FWA$L_SBN = 424,0,32,0 %;                !  starting lbn if contiguous
0       macro FWA$L_HBK = 428,0,32,0 %;                !  high vbn
0       !
0       !   node descriptors
0       !
0       macro FWA$Q_NODE1 = 436,0,0,0 %;
0       literal FWA$S_NODE1 = 8;                       !  primary node spec descriptor
0       !  (the associated buffer is fwa$t_nodebuf)
0       macro FWA$Q_NODE2 = 444,0,0,0 %;
0       literal FWA$S_NODE2 = 8;                       !  secondary (sub) node spec descriptors (1-7)
0       macro FWA$Q_NODE3 = 452,0,0,0 %;
0       literal FWA$S_NODE3 = 8;                       !  note: bytes 2-3 of each of these descriptors
0       macro FWA$Q_NODE4 = 460,0,0,0 %;
0       literal FWA$S_NODE4 = 8;                       !  contains the flags word that is output
0       macro FWA$Q_NODE5 = 468,0,0,0 %;
0       literal FWA$S_NODE5 = 8;                       !  from nxtfld subroutine in rm0xpfn
0       macro FWA$Q_NODE6 = 476,0,0,0 %;
0       literal FWA$S_NODE6 = 8;                       !  note: fwa$q_node1 thru 'fwa$q_node8'
0       macro FWA$Q_NODE7 = 484,0,0,0 %;
0       literal FWA$S_NODE7 = 8;                       !  describe the same string as does
0       macro FWA$Q_NODE8 = 492,0,0,0 %;
0       literal FWA$S_NODE8 = 8;                       !  fwa$q_node
0       macro FWA$T_FIBBUF = 500,0,0,0 %;
0       literal FWA$S_FIBBUF = 76;                     !  fib buffer
0       macro FWA$T_RNM_FID = 576,0,0,0 %;
0       literal FWA$S_RNM_FID = 6;                     !  saved fid for rename directory check
0       !
0       !   directory name buffers
0       !
0       !   NOTE:  These buffers must be contiguous
0       !
0       macro FWA$T_DIR1BUF = 582,0,0,0 %;
0       literal FWA$S_DIR1BUF = 39;                    !  ufd level (or group)
0       macro FWA$T_DIR2BUF = 621,0,0,0 %;
0       literal FWA$S_DIR2BUF = 39;                    !  1st sfd level (or member)
0       macro FWA$T_DIR3BUF = 660,0,0,0 %;
0       literal FWA$S_DIR3BUF = 39;                    !  subdirectory 2
0       macro FWA$T_DIR4BUF = 699,0,0,0 %;
0       literal FWA$S_DIR4BUF = 39;                    !  subdirectory 3
0       macro FWA$T_DIR5BUF = 738,0,0,0 %;
0       literal FWA$S_DIR5BUF = 39;                    !  subdirectory 4
0       macro FWA$T_DIR6BUF = 777,0,0,0 %;
0       literal FWA$S_DIR6BUF = 39;                    !  subdirectory 5
0       macro FWA$T_DIR7BUF = 816,0,0,0 %;
0       literal FWA$S_DIR7BUF = 39;                    !  subdirectory 6
0       macro FWA$T_DIR8BUF = 855,0,0,0 %;
0       literal FWA$S_DIR8BUF = 39;                    !  subdirectory 7
0       macro FWA$T_CDIR1BUF = 894,0,0,0 %;
0       literal FWA$S_CDIR1BUF = 39;                   !  ufd level (or group)
0       macro FWA$T_CDIR2BUF = 933,0,0,0 %;
0       literal FWA$S_CDIR2BUF = 39;                   !  1st sfd level (or member)
0       macro FWA$T_CDIR3BUF = 972,0,0,0 %;
0       literal FWA$S_CDIR3BUF = 39;                   !  subdirectory 2
0       macro FWA$T_CDIR4BUF = 1011,0,0,0 %;
0       literal FWA$S_CDIR4BUF = 39;                   !  subdirectory 3
0       macro FWA$T_CDIR5BUF = 1050,0,0,0 %;
```

```
0   literal FWA$S_CDIR5BUF = 39;                 !  subdirectory 4
0   macro FWA$T_CDIR6BUF = 1089,0,0,0 %;
0   literal FWA$S_CDIR6BUF = 39;                 !  subdirectory 5
0   macro FWA$T_CDIR7BUF = 1128,0,0,0 %;
0   literal FWA$S_CDIR7BUF = 39;                 !  subdirectory 6
0   macro FWA$T_CDIR8BUF = 1167,0,0,0 %;
0   literal FWA$S_CDIR8BUF = 39;                 !  subdirectory 7
0   !
0   !    NOTES: 1. The following buffers must be contiguous as eventually the
0   !              type and version are appended to the name string
0   !
0   !           2. The name buffer and the type buffer must be 1 byte larger then
0   !              the max name and type size (resp) because xpfn writes the
0   !              name and type terminators in the buffer at the end of the string
0   !
0   macro FWA$T_NAMEBUF = 1206,0,0,0 %;
0   literal FWA$S_NAMEBUF = 256;                 !  file name/quoted string buffer
0   macro FWA$T_TYPEBUF = 1462,0,0,0 %;
0   literal FWA$S_TYPEBUF = 40;                  !  file type buffer
0   macro FWA$T_VERBUF = 1502,0,0,0 %;
0   literal FWA$S_VERBUF = 6;                    !  file version buffer
0   macro FWA$L_UCBSTS = 1508,0,32,0 %;          !  ucb$l_sts field for prim device
0   macro FWA$B_UNDER_DEV = 1512,0,8,0 %;        !  character "_" stored here
0   macro FWA$T_DEVICEBUF = 1513,0,0,0 %;
0   literal FWA$S_DEVICEBUF = 255;               !  device name buffer
0   macro FWA$T_CDEVICEBUF = 1768,0,0,0 %;
0   literal FWA$S_CDEVICEBUF = 256;              !  concealed device name buffer
0   macro FWA$B_UNDER_NOD = 2024,0,8,0 %;        !  character "_" stored here
0   macro FWA$T_NODEBUF = 2025,0,0,0 %;
0   literal FWA$S_NODEBUF = 127;                 !  node name buffer
0   macro FWA$T_WILD = 2152,0,0,0 %;
0   literal FWA$S_WILD = 48;                     !  scratch field used by RM0WILD
0   !     size =        count       1
0   !                   name       39
0   !                   .dir;*      6
0   !                   spare       2
0   !                   ------------
0   !                              48
0   macro FWA$T_SHRFILBUF = 2200,0,0,0 %;
0   literal FWA$S_SHRFILBUF = 16;                !  shared file device id buffer (readable form)
0   macro FWA$T_SHRFIL_LCKNAM = 2216,0,0,0 %;
0   literal FWA$S_SHRFIL_LCKNAM = 16;            !  shared file device id buffer (unreadable form - used for lock name)
0   macro FWA$T_AS_SHRFILBUF = 2232,0,0,0 %;
0   literal FWA$S_AS_SHRFILBUF = 16;             !  secondary device id buffer (readable form)
0   macro FWA$Q_BIJNL = 2248,0,0,0 %;
0   literal FWA$S_BIJNL = 8;                     !  descriptor of BI journal name
0   macro FWA$Q_AIJNL = 2256,0,0,0 %;
0   literal FWA$S_AIJNL = 8;                     !  descriptor of AI journal name
0   macro FWA$Q_ATJNL = 2264,0,0,0 %;
0   literal FWA$S_ATJNL = 8;                     !  descriptor of AT journal name
0   macro FWA$T_BIACE = 2272,0,0,0 %;
0   literal FWA$S_BIACE = 20;                    !  BI journal name ACE
0   macro FWA$T_BIJNLN = 2276,0,0,0 %;
0   literal FWA$S_BIJNLN = 16;
0   macro FWA$T_AIACE = 2292,0,0,0 %;
0   literal FWA$S_AIACE = 20;                    !  AI journal name ACE
0   macro FWA$T_AIJNLN = 2296,0,0,0 %;
```

```
0       literal FWA$S_AIJNLN = 16;
0       macro FWA$T_ATACE = 2312,0,0,0 %;
0       literal FWA$S_ATACE = 20;               !  AT journal name ACE
0       macro FWA$T_ATJNLN = 2316,0,0,0 %;
0       literal FWA$S_ATJNLN = 16;
0       macro FWA$T_IDACE = 2332,0,0,0 %;
0       literal FWA$S_IDACE = 32;               !  Journal ID ACE
0       macro FWA$T_JNLID = 2336,0,0,0 %;
0       literal FWA$S_JNLID = 28;               !  complete journal ID
0       macro FWA$T_VOLNAM = 2336,0,0,0 %;
0       literal FWA$S_VOLNAM = 12;              !  volume lable of media file resides on
0       macro FWA$T_FID = 2348,0,0,0 %;
0       literal FWA$S_FID = 6;                  !  file-id
0       macro FWA$Q_ID_DATE = 2356,0,0,0 %;
0       literal FWA$S_ID_DATE = 8;              !  id time stamp
0
0       !*** MODULE $SLBHDEF ***
0       !
0       !       SLBH        - Search List Header Block
0       !
0       literal SLBH$C_BID = 43;                !  ID
0       literal SLBH$K_BLN = 20;                !  length of SLBH
0       literal SLBH$C_BLN = 20;                !  length of SLBH
0       literal SLBH$S_SLBHDEF = 20;
0       macro SLBH$L_FLINK = 0,0,32,0 %;        !  forward link
0       macro SLBH$L_BLINK = 4,0,32,0 %;        !  backward link
0       macro SLBH$B_BID = 8,0,8,0 %;           !  block ID
0       macro SLBH$B_BLN = 9,0,8,0 %;           !  length
0       macro SLBH$B_PASSFLGS = 10,0,8,0 %;     !  flags for FWA$B_PASSFLGS
0       macro SLBH$B_STR_LEN = 11,0,8,0 %;      !  string length
0       macro SLBH$L_SLB_QUE = 12,0,32,0 %;     !  ptr to SLB queue
0       macro SLBH$L_NAM_FNB = 16,0,32,0 %;     !  saved FNB from RLF file
0       macro SLBH$T_STRING = 20,0,0,0 %;       !  start of string
0
0       !*** MODULE $SLBDEF ***
0       !
0       !       SLB         - Search List Block
0       !
0       literal SLB$C_BID = 41;                 !  ID
0       literal SLB$M_REALSLB = 1;
0       literal SLB$K_BLN = 24;                 !  length of SLB
0       literal SLB$C_BLN = 24;                 !  length of SLB
0       literal SLB$S_SLBDEF = 24;
0       macro SLB$L_FLINK = 0,0,32,0 %;         !  forward link
0       macro SLB$L_BLINK = 4,0,32,0 %;         !  backward link
0       macro SLB$B_BID = 8,0,8,0 %;            !  block ID
0       macro SLB$B_BLN = 9,0,8,0 %;            !  length
0       macro SLB$B_FLAGS = 10,0,8,0 %;         !  flags
0       macro SLB$V_REALSLB = 10,0,1,0 %;       !  "Real" SLB as opposed to the fake FWA one
0       macro SLB$B_LEVEL = 11,0,8,0 %;         !  recursion level
0       macro SLB$L_INDEX = 12,0,32,0 %;        !  translation index
0       macro SLB$L_MAX_INDEX = 16,0,32,0 %;    !  max translation index
0       macro SLB$L_ATTR = 20,0,32,0 %;         !  attributes flags
0
0       !*** MODULE $FSCBDEF ***
0       !
0       !       FSCB - FileScan control block
```

```
!
!               This block is passed to PARSE_STRING from XPFN and RMS$FILESCAN
!
!
!               The descriptors are defined as:
!
!               ------------------------------------------------
!               |        flags          |        length        |
!               |----------------------------------------------|
!               |                    address                   |
!               ------------------------------------------------
!
!
!
!
!               descriptor flags
!
!               These flags are used through out the RMS file name parsing routines.
!               The flags can be found in all of the field descriptors.
!
!               NOTE: The flag ELIPS must be the first bit in the second word.
!                     It is referenced this way in RMOWILD and other places
!
literal FSCB$M_ELIPS = 65536;
literal FSCB$M_WILD = 131072;
literal FSCB$M_ACS = 262144;
literal FSCB$M_QUOTED = 524288;
literal FSCB$M_NULL = 1048576;
literal FSCB$M_PWD = 2097152;
literal FSCB$M_GRPMBR = 4194304;
literal FSCB$M_MINUS = 8388608;
literal FSCB$M_CONCEAL = 16777216;
literal FSCB$M_MFD = 33554432;
literal FSCB$M_ROOTED = 67108864;
literal FSCB$S_FSCBDEF = 4;
macro FSCB$V_ELIPS = 0,16,1,0 %;           ! elipssis was detected in directory (dir)
macro FSCB$V_WILD = 0,17,1,0 %;            ! a wild card was detected (dir,name,type,ver)
macro FSCB$V_ACS = 0,18,1,0 %;             ! access control string in node name (node)
macro FSCB$V_QUOTED = 0,19,1,0 %;          ! quoted file spec (name)
macro FSCB$V_NULL = 0,20,1,0 %;            ! field was null (terminator only) (all)
macro FSCB$V_PWD = 0,21,1,0 %;             ! password masked out (set in xpfn) (node)
macro FSCB$V_GRPMBR = 0,22,1,0 %;          ! group,member format directory (dir)
macro FSCB$V_MINUS = 0,23,1,0 %;           ! minus directory field (dir)
macro FSCB$V_CONCEAL = 0,24,1,0 %;         ! name was concealed (dev)
macro FSCB$V_MFD = 0,25,1,0 %;             ! MFD directory (set in xpfn) (dir)
macro FSCB$V_ROOTED = 0,26,1,0 %;          ! directory was a root directory (dir)
!
!               FSCB
!
literal FSCB$M_NODE = 1;
literal FSCB$M_DEVICE = 2;
literal FSCB$M_ROOT = 4;
literal FSCB$M_DIRECTORY = 8;
literal FSCB$M_NAME = 16;
literal FSCB$M_TYPE = 32;
literal FSCB$M_VERSION = 64;
literal FSCB$C_MAXNODE = 8;                ! max number of node descriptors
literal FSCB$C_MAXROOT = 8;                ! max number of root descriptors
```

```
0      literal FSCB$K_BLN = 260;
0      literal FSCB$C_BLN = 260;
0      literal FSCB$C_MAXDIR = 8;             ! max number of directory descriptors
0      literal FSCB$S_FSCBDEF1 = 260;
0      macro FSCB$B_FLDFLAGS = 0,0,8,0 %;     ! field flags
0      macro FSCB$V_NODE = 0,0,1,0 %;
0      macro FSCB$V_DEVICE = 0,1,1,0 %;
0      macro FSCB$V_ROOT = 0,2,1,0 %;
0      macro FSCB$V_DIRECTORY = 0,3,1,0 %;
0      macro FSCB$V_NAME = 0,4,1,0 %;
0      macro FSCB$V_TYPE = 0,5,1,0 %;
0      macro FSCB$V_VERSION = 0,6,1,0 %;
0      macro FSCB$B_NODES = 1,0,8,0 %;        ! number of nodes in spec
0      macro FSCB$B_ROOTS = 2,0,8,0 %;        ! number of root directories in spec
0      macro FSCB$B_DIRS = 3,0,8,0 %;         ! number of directories in spec
0      macro FSCB$Q_FILESPEC = 4,0,0,0 %;
0      literal FSCB$S_FILESPEC = 8;           ! full file spec
0      macro FSCB$Q_NODE = 12,0,0,0 %;
0      literal FSCB$S_NODE = 8;               ! full node list spec
0      macro FSCB$Q_DEVICE = 20,0,0,0 %;
0      literal FSCB$S_DEVICE = 8;             ! device spec
0      macro FSCB$Q_ROOT = 28,0,0,0 %;
0      literal FSCB$S_ROOT = 8;               ! full root directory list spec
0      macro FSCB$Q_DIRECTORY = 36,0,0,0 %;
0      literal FSCB$S_DIRECTORY = 8;          ! full directory list spec
0      macro FSCB$Q_NAME = 44,0,0,0 %;
0      literal FSCB$S_NAME = 8;               ! file name
0      macro FSCB$Q_TYPE = 52,0,0,0 %;
0      literal FSCB$S_TYPE = 8;               ! file type
0      macro FSCB$Q_VERSION = 60,0,0,0 %;
0      literal FSCB$S_VERSION = 8;            ! file version
0      macro FSCB$Q_NODE1 = 68,0,0,0 %;
0      literal FSCB$S_NODE1 = 8;              ! the NODEn descriptors must be contiguous
0      macro FSCB$Q_NODE2 = 76,0,0,0 %;
0      literal FSCB$S_NODE2 = 8;
0      macro FSCB$Q_NODE3 = 84,0,0,0 %;
0      literal FSCB$S_NODE3 = 8;
0      macro FSCB$Q_NODE4 = 92,0,0,0 %;
0      literal FSCB$S_NODE4 = 8;
0      macro FSCB$Q_NODE5 = 100,0,0,0 %;
0      literal FSCB$S_NODE5 = 8;
0      macro FSCB$Q_NODE6 = 108,0,0,0 %;
0      literal FSCB$S_NODE6 = 8;
0      macro FSCB$Q_NODE7 = 116,0,0,0 %;
0      literal FSCB$S_NODE7 = 8;
0      macro FSCB$Q_NODE8 = 124,0,0,0 %;
0      literal FSCB$S_NODE8 = 8;
0      macro FSCB$Q_ROOT1 = 132,0,0,0 %;
0      literal FSCB$S_ROOT1 = 8;              ! the ROOTn descriptors must be contiguous
0      macro FSCB$Q_ROOT2 = 140,0,0,0 %;
0      literal FSCB$S_ROOT2 = 8;
0      macro FSCB$Q_ROOT3 = 148,0,0,0 %;
0      literal FSCB$S_ROOT3 = 8;
0      macro FSCB$Q_ROOT4 = 156,0,0,0 %;
0      literal FSCB$S_ROOT4 = 8;
0      macro FSCB$Q_ROOT5 = 164,0,0,0 %;
0      literal FSCB$S_ROOT5 = 8;
```

```
macro FSCB$Q_ROOT6 = 172,0,0,0 %;
literal FSCB$S_ROOT6 = 8;
macro FSCB$Q_ROOT7 = 180,0,0,0 %;
literal FSCB$S_ROOT7 = 8;
macro FSCB$Q_ROOT8 = 188,0,0,0 %;
literal FSCB$S_ROOT8 = 8;
macro FSCB$Q_DIRECTORY1 = 196,0,0,0 %;
literal FSCB$S_DIRECTORY1 = 8;          !  the DIRECTORYn descriptors must be contiguous
macro FSCB$Q_DIRECTORY2 = 204,0,0,0 %;
literal FSCB$S_DIRECTORY2 = 8;
macro FSCB$Q_DIRECTORY3 = 212,0,0,0 %;
literal FSCB$S_DIRECTORY3 = 8;
macro FSCB$Q_DIRECTORY4 = 220,0,0,0 %;
literal FSCB$S_DIRECTORY4 = 8;
macro FSCB$Q_DIRECTORY5 = 228,0,0,0 %;
literal FSCB$S_DIRECTORY5 = 8;
macro FSCB$Q_DIRECTORY6 = 236,0,0,0 %;
literal FSCB$S_DIRECTORY6 = 8;
macro FSCB$Q_DIRECTORY7 = 244,0,0,0 %;
literal FSCB$S_DIRECTORY7 = 8;
macro FSCB$Q_DIRECTORY8 = 252,0,0,0 %;
literal FSCB$S_DIRECTORY8 = 8;

!*** MODULE $SWBDEF ***
!
!  Directory string work buffer for wild card directory processing
!
literal SWB$M_ELLIPSIS = 1;
literal SWB$M_BOUNDED = 2;
literal SWB$M_WILD = 4;
literal SWB$M_DELIMITER = 8;
literal SWB$M_TRAVERSE = 16;
literal SWB$M_FIRST = 32;
literal SWB$M_ELLIPSIS_EXISTS = 64;
literal SWB$M_VALID_DID = 128;
literal SWB$C_BID = 42;                 !  ID
literal SWB$K_BLN = 328;
literal SWB$C_BLN = 328;
!   wild dir spec
literal SWB$S_SWBDEF = 328;
macro SWB$B_FLAGS = 0,0,8,0 %;          !  flags (must be first)
macro SWB$V_ELLIPSIS = 0,0,1,0 %;       !  ellipsis
macro SWB$V_BOUNDED = 0,1,1,0 %;        !  ellipsis bounded
macro SWB$V_WILD = 0,2,1,0 %;           !  wild name
macro SWB$V_DELIMITER = 0,3,1,0 %;      !  following delimiter
macro SWB$V_TRAVERSE = 0,4,1,0 %;       !  should skip subtree
macro SWB$V_FIRST = 0,5,1,0 %;          !  first time through
macro SWB$V_ELLIPSIS_EXISTS = 0,6,1,0 %; !  dir spec contains ...
macro SWB$V_VALID_DID = 0,7,1,0 %;      !  FIB DID is valid
macro SWB$B_PATLEN = 1,0,8,0 %;         !  length of current token
macro SWB$B_PPOS = 2,0,8,0 %;           !  position in pattern
macro SWB$B_TOKENS_LEFT = 3,0,8,0 %;    !  number of non ... tokens left
macro SWB$B_MINIMUM = 4,0,8,0 %;        !  minimum level for success
macro SWB$B_MAXIMUM = 5,0,8,0 %;        !  maximum level for success
macro SWB$B_FIRST_E = 6,0,8,0 %;        !  token ! of first ellipsis
macro SWB$B_DIRWCFLGS = 7,0,8,0 %;      !  FWA$B_DIRWCFLGS on entry
macro SWB$B_BID = 8,0,8,0 %;            !  block ID
```

```
macro SWB$B_BLN = 9,0,8,0 %;                   ! length
macro SWB$Q_PATTERN = 12,0,0,0 %;
literal SWB$S_PATTERN = 8;                     ! descriptor of pattern
macro SWB$L_SCRATCH_PAT = 20,0,32,0 %;         ! scratch copy of first longword
macro SWB$T_SCRATCH_BUF = 24,0,0,0 %;
literal SWB$S_SCRATCH_BUF = 48;                ! scratch temp buffer (same size as FWA$T_WILD)
macro SWB$T_PATTERN_BUF = 72,0,0,0 %;
literal SWB$S_PATTERN_BUF = 256;               ! should be: FWA$C_MAXDIRLEN-2,
!
! *****************************************************************************
! *                                                                           *
! *  Copyright (c) 1982, 1983                                                 *
! *  by DIGITAL Equipment Corporation, Maynard, Mass.                         *
! *                                                                           *
! *  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.                  *
! *                                                                           *
! *****************************************************************************
! *********************************************************************************************************************
! Created 15-SEP-1984 22:54:43  by  VAX-11 SDL V2.0       Source: 15-SEP-1984 22:49:34 _$255$DUA28:[RMS.SRC]RMSSHR.
! *********************************************************************************************************************


!*** MODULE $SFSBDEF ***
literal SFSB$C_BID = 16;                       ! sfsb code
literal SFSB$C_FIX_LEN = 10;                   ! 10 bytes of fixed size data
literal SFSB$K_BLN = 68;                       ! length of sfsb
literal SFSB$C_BLN = 68;                       ! length of sfsb
!
!       keep the next two fields in same order as they are in FAB
!
literal SFSB$S_SFSBDEF = 68;
macro SFSB$Q_FILENAME = 0,0,0,0 %;
literal SFSB$S_FILENAME = 8;                   ! descriptor of shared file resource name.
!   resource name is NODE, DEVICE, FILE_ID
!   points to RESNAM, below
macro SFSB$W_NAME_LEN = 0,0,16,0 %;            ! subfield to address descriptor length field
macro SFSB$L_ADDRESS = 4,0,32,0 %;             ! subfield to address descriptor address field
macro SFSB$B_BID = 8,0,8,0 %;                  ! block id
macro SFSB$B_BLN = 9,0,8,0 %;                  ! block length in longwords
macro SFSB$B_CURMODE = 10,0,8,0 %;             ! Mode of the current lock
macro SFSB$B_PREMODE = 11,0,8,0 %;             ! Mode of the previous lock
macro SFSB$T_RESNAM = 12,0,0,0 %;
literal SFSB$S_RESNAM = 32;                    ! 32 bytes for name of shared resource
macro SFSB$L_FAC_CODE = 12,0,32,0 %;           ! RMS facility code (RMS$)
macro SFSB$W_FID_NUM = 16,0,16,0 %;            ! file id word one
```

```
macro SFSB$W_FID_SEQ = 18,0,16,0 %;      !  file id word two
macro SFSB$W_FID_RVN = 20,0,16,0 %;      !  file id word three
macro SFSB$T_DEV_NAM = 22,0,0,0 %;
literal SFSB$S_DEV_NAM = 22;              !  22 bytes remain to hold device id (node$device_name)
macro SFSB$L_LKSB = 44,0,32,0 %;         !  lock status block
macro SFSB$W_STATUS = 44,0,16,0 %;       !  VMS status code
macro SFSB$W_S_BITS = 46,0,16,0 %;       !  various status bits
macro SFSB$L_LOCK_ID = 48,0,32,0 %;      !  second longword of LKSB is the lock id
macro SFSB$L_LVB = 52,0,0,0 %;
literal SFSB$S_LVB = 16;                  !  lock value block
macro SFSB$B_FAC = 52,0,8,0 %;           !  fac bits from FAB
macro SFSB$B_SHR = 53,0,8,0 %;           !  sharing bits (from FAB SHR field)
macro SFSB$L_HBK = 60,0,32,0 %;          !  high block
macro SFSB$L_EBK = 64,0,32,0 %;          !  end of file

!*** MODULE $GBSBDEF ***
!
!        GBSB field definitions - global buffer synchronization block
!
!        The GBSB contains the information necessary to determine if a
!        global section is already open for a file on a given node, and
!        is used for synchronizing access to the global section.
!
!                     +-------------------------------+
!  gbsb:              |           FILE_NAME           |
!                     |                               |
!                     |-------+---------+--------+--------|
!                     | FLAGS | CURMODE |   BLN  |   BID  |
!                     +-------+---------+--------+--------+
!                     |                                 |
!                     |          Resource Name          |   |
!                     |                                 |
!                     +---------------------------------+
!  lksb:              | Still to be def- | VMS status code |
!                     | ined status bits |                 |
!                     +-------------------------------+
!                     | Lock Id. (Returned for new locks, |
!                     |           input for conversions)  |
!                     +-------------------------------+
!  lkvb:              |      GBC      |      GBREF          |
!                     +-------------------------------+
!                     |    GBS - size of GS in bytes  |
!                     +-------------------------------+
!                     |             spare                 |
!                     +-------------------------------+
!                     |             spare                 |
!                     +-------------------------------+
!
!
literal GBSB$C_BID = 9;                   !  gbsb code
literal GBSB$M_NOTACCESSED = 1;
literal GBSB$K_BLN = 68;                  !  length of gbsb
literal GBSB$C_BLN = 68;                  !  length of gbsb
literal GBSB$S_GBSBDEF = 68;
macro GBSB$Q_FILENAME = 0,0,0,0 %;
literal GBSB$S_FILENAME = 8;              !  descriptor of shared file resource name.
!   resource name is NODE, DEVICE, FILE_ID
```

```
0	!   points to RESNAM, below
0	macro GBSB$W_NAME_LEN = 0,0,16,0 %;        !  subfield to address descriptor length field
0	macro GBSB$L_ADDRESS = 4,0,32,0 %;         !  subfield to address descriptor address field
0	macro GBSB$B_BID = 8,0,8,0 %;              !  block id
0	macro GBSB$B_BLN = 9,0,8,0 %;              !  block length in longwords
0	macro GBSB$B_CURMODE = 10,0,8,0 %;         !  Mode of the current lock
0	macro GBSB$B_FLAGS = 11,0,8,0 %;           !  spare
0	macro GBSB$V_NOTACCESSED = 11,0,1,0 %;     !  Process has already decremeted access count for GBS.
0	macro GBSB$T_RESNAM = 12,0,0,0 %;
0	literal GBSB$S_RESNAM = 32;                !  32 bytes for name of shared resource
0	macro GBSB$L_LKSB = 44,0,32,0 %;           !  lock status block
0	macro GBSB$W_STATUS = 44,0,16,0 %;         !  VMS status code
0	macro GBSB$W_S_BITS = 46,0,16,0 %;         !  various status bits
0	macro GBSB$L_LOCK_ID = 48,0,32,0 %;        !  second longword of LKSB is the lock id
0	macro GBSB$W_GBC = 52,0,16,0 %;            !  Number of global buffers in section.
0	macro GBSB$W_GBREF = 54,0,16,0 %;          !  Number of accessors to global section.
0	macro GBSB$L_GS_SIZE = 56,0,32,0 %;        !  Size of global section in bytes.
0	!
0	! ****************************************************************************
0	! *                                                                          *
0	! *  Copyright (c) 1982, 1983                                                *
0	! *  by DIGITAL Equipment Corporation, Maynard, Mass.                        *
0	! *                                                                          *
0	! *  This software is furnished under a license and may be used and  copied  *
0	! *  only  in  accordance  with  the  terms  of  such  license and with the  *
0	! *  inclusion of the above copyright notice.  This software or  any  other  *
0	! *  copies  thereof may not be provided or otherwise made available to any  *
0	! *  other person.  No title to and ownership of  the  software  is  hereby  *
0	! *  transferred.                                                            *
0	! *                                                                          *
0	! *  The information in this software is subject to change  without  notice  *
0	! *  and  should  not  be  construed  as  a commitment by DIGITAL Equipment  *
0	! *  Corporation.                                                            *
0	! *                                                                          *
0	! *  DIGITAL assumes no responsibility for the use or  reliability  of  its  *
0	! *  software on equipment which is not supplied by DIGITAL.                 *
0	! *                                                                          *
0	! ****************************************************************************
0	! *********************************************************************************************************
0	!  Created 15-SEP-1984 22:54:49  by  VAX-11 SDL V2.0       Source: 15-SEP-1984 22:53:49 _$255$DUA28:[RMS.SRC]RMSDEF.
0	! *********************************************************************************************************
0
0
0	!*** MODULE $RMSDEF ***
0	!
0	!  This SDL File Generated by VAX-11 Message V04-00 on 15-SEP-1984 22:53:50.83
0	!
0	!       .TITLE  RMSDEF -RMS COMPLETION CODES
0	! ****************************************************************************
0	! *                                                                          *
0	! *  COPYRIGHT (C) 1978, 1980, 1982, 1984 BY                                 *
0	! *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0	! *  ALL RIGHTS RESERVED.                                                    *
0	! *                                                                          *
0	! *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0	! *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0	! *  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.                  *
        ! *                                                                           *
        ! *                                                                           *
        ! *****************************************************************************
        !++
        ! FACILITY: RMS
        !
        ! ABSTRACT:
        !
        !      THIS MODULE DEFINES ALL RMS COMPLETION CODES.
        !
        ! ENVIRONMENT:
        !
        !      THE MESSAGE TRANSLATOR MUST BE USED TO CONVERT RMSDEF.MSG INTO
        !      RMSDEF.SDL.  THE SDL TRANSLATOR MUST THEN BE USED TO CONVERT
        !      RMSDEF.SDL INTO RMSDEF.MAR (AND RMSDEF.B32).
        !
        ! AUTHOR: LEO F. LAVERDURE,    CREATION DATE:  10-DEC-1976
        !
        ! MODIFIED BY:
        !
        !      V03-024 RAS0314         RON SCHAEFER                21-JUN-1984
        !              WORK OVER THE MESSAGES ONE MORE TIME:  FIX TYPO IN OK_RNF,
        !              DELETE ACPEOF ERROR, DELETE WSF ERROR.
        !
        !      V03-023 RAS0282         RON SCHAEFER                28-MAR-1984
        !              MINOR TEXT CHANGES AND COMMENTS ADDED.
        !              DELETE RMS$_COP ERROR.
        !
        !      V03-022 DAS0005         DAVID SOLOMON               19-MAR-1984
        !              REMOVE RMS$_ECHO (NO LONGER NEEDED AS A RESULT OF NEW
        !              IMPLEMENTATION FOR ECHO SYS$INPUT -> SYS$OUTPUT).
        !
        !      V03-021 JWT0150         JIM TEAGUE                  01-FEB-1984
        !              ADD RMS$_IFF FOR TRYING TO WRITE-ACCESS A FILE ON
        !              MAGTAPE THAT HAS NON-0 VALUE FOR ANSI BUFFER OFFSET.
        !
        !      V03-020 RAS0233         RON SCHAEFER                9-JAN-1984
        !              ADD RMS$_NOVALPRS ERROR FOR $SEARCH NOT PRECEDED BY
        !              VALID $PARSE.
        !
        !      V03-019 JWT0148         JIM TEAGUE                  20-DEC-1983
        !              ADD A JNL XAB ERROR FOR CONFLICTING RU ATTRIBUTES
        !              ON $CREATE: RMS$_XCR.
        !
        !      V03-018 RAS0171         RON SCHAEFER                28-JUL-1983
        !              ADD RMS$_BUSY; A STRUCTURE-LESS ERROR (RO-ONLY STATUS)
        !              REPLACING TEMP3.
```

```
0   !
0   !       V03-017 DAS0004         DAVID SOLOMON           28-JUN-1983
0   !               ADD RMS$_FILEPURGED; ADD RMS$_ACPEOF FOR ZALEWSKI.
0   !
0   !       V03-016 KPL0007         PETER LIEBERWIRTH       8-JUN-1983
0   !               FIX SPELLING IN JNL ERROR MESSAGES, ADD CJF ERROR WHICH
0   !               WILL SOON SUBSUME COP AND CQE.
0   !
0   !       V03-015 DAS0003         DAVID SOLOMON           18-FEB-1983
0   !               ADD XNF (REPLACING TEMP9), TMR (NEW), LWC (NEW).
0   !
0   !       V03-014 KBT0497         KEITH B. THOMPSON       18-FEB-1983
0   !               ADD INCOMPSHR ERROR (REUSE OF TEMP1 SLOT)
0   !
0   !       V03-013 RAS0120         RON SCHAEFER            2-FEB-1983
0   !               ADD ECHO PSEUDO-STATUS TO SUPPORT ECHO OF SYS$INPUT
0   !               TO SYS$OUTPUT.
0   !
0   !       V03-012 JWH0174         JEFFRY W. HORN          24-JAN-1983
0   !               ADD CQE AND COP ERRORS.
0   !
0   !       V03-011 KPL0006         PETER LIEBERWIRTH       7-JAN-1983
0   !               ADD OK_RULK FOR RECOVERY UNIT SUPPORT.
0   !
0   !               ADD REENT ERROR FOR $RENAME.
0   !
0   !       V03-009 JWH0153         JEFFREY W. HORN         8-DEC-1982
0   !               ADD NRU ERROR.
0   !
0   !       V03-008 JWH0152         JEFFREY W. HORN         8-DEC-1982
0   !               ADD JNS ERROR.
0   !
0   !       V03-007 MCN0002         MARIA DEL C. NASR       15-NOV-1982
0   !               REPLACE ORD ERROR CODE ELIMINATED BY MCN0001, SINCE
0   !               THE NETWORK CODE REFERENCES IT.
0   !
0   !       V03-006 MCN0001         MARIA DEL C. NASR       26-OCT-1982
0   !               PRINT KEY OF REFERENCE OR AREA IDENTIFICATION NUMBER
0   !               INSTEAD OF XAB ADDRESS FOR ERRORS RETURNED DURING
0   !               INDEXED FILE CREATION.  ALSO RENAME ALL UNUSED ERROR
0   !               CODES TO TEMP.
0   !
0   !       V03-005 JWH0102         JEFFREY W. HORN         24-SEP-1982
0   !               ADD RUP ERROR.
0   !
0   !       V03-004 RAS0095         RON SCHAEFER            7-SEP-1982
0   !               ADD OVRDSKQUOTA ERROR.
0   !
0   !       V03-002 JWH0002         JEFFREY W. HORN         26-JUL-1982
0   !               CONVERT TO .MSG FORMAT.   ADD RUM ERROR.
0   !
0   !       V03-001 JWH0001         JEFFREY W. HORN         20-JUL-1982
0   !               ADD JNF, JOP, AND NOJ ERRORS.
0   !
0   !       V02-042 KPL0005         PETER LIEBERWIRTH       4-FEB-1982
0   !               ADD ERROR MESSAGES RMS$_EXENQLM AND RMS$_DEADLOCK
0   !               CORRECTLY.
```

```
!
!       V02-041 CDS0001          C D SAETHER      24-JAN-1982
!               ADD GBC AND CRMP ERRORS.
!
!       V02-040 JAK0069          J A KRYCKA       15-JAN-1982
!               ADD SUPPORT AND NETFAIL ERROR CODES.
!
!       V02-039 JAK0068          J A KRYCKA       31-DEC-1981
!               MODIFY TEXT FOR SUP, NET, BUG_DAP, AND ENV ERROR MESSAGES.
!
!       V02-038 LJA0001          LAURIE ANDERSON 20-DEC-1981
!               MODIFIED SOME MESSAGES TO READ BETTER.
!
!       V02-037 JAK0063          J A KRYCKA       31-AUG-1981
!               ADD CRE_STM SUCCESS CODE.
!
!       V02-036 KPL0004          PETER LIEBERWIRTH       13-JAN-1981
!               ADD ENQ SEVERE ERROR MESSAGE, TO INDICATE $ENQ/$DEQ FAILURE.
!               ALSO ADD SUCCESS CODES OK_RRL, AND OK_WAT.
!
!       V02-035 REFORMAT         J A KRYCKA       30-JUL-1980
!
! --
literal RMS$_FACILITY = 1;
literal RMS$V_STVSTATUS = 14;              !  MOVE TO BIT 14 OF THE
!  STATUS CODE IT INDICATES
!  THAT STV CONTAINS A SECONDARY
!  STATUS CODE.
literal RMS$_SUC = 65537;
literal RMS$_NORMAL = 65537;
! +
!
!       SUCCESS CODES
!
! -
!  BIT 16 = BIT 15 = 1
literal RMS$_STALL = 98305;
!  (NOTE: USER NEVER RECEIVES THIS CODE)
literal RMS$_PENDING = 98313;
literal RMS$_OK_DUP = 98321;
literal RMS$_OK_IDX = 98329;
!    (RECORD HAS BEEN INSERTED, BUT INDEX STRUCTURE IS NO LONGER
!    OPTIMAL.)
literal RMS$_OK_RLK = 98337;
!  (BECAUSE LOCKER SET RLK IN ROP FIELD WHEN RECORD WAS LOCKED.)
literal RMS$_OK_RRL = 98345;
!  (BECAUSE READER SET RRL IN ROP FIELD WHEN ACCESSING RECORD.)
!  (THIS CODE USED TO BE OK_RRV)
! OK_RRV        ;MSG <record was inserted successfully in primary>
!    (BUT IT MAY NOT BE ACCESSIBLE VIA ONE-OR-MORE SECONDARY KEYS,
!    AND NOT BY RFA ADDRESSING. FILE RE-ORGANIZATION RECOMMENDED!)
literal RMS$_KFF = 98353;
literal RMS$_OK_ALK = 98361;
literal RMS$_OK_DEL = 98369;
literal RMS$_OK_RNF = 98377;
literal RMS$_OK_LIM = 98385;
literal RMS$_OK_NOP = 98393;
```

```
literal RMS$_OK_WAT = 98401;
!  (BECAUSE WAT BIT IN ROP WAS SET AND RECORD WAS ALREADY
!  LOCKED.)
literal RMS$_CRE_STM = 98409;
literal RMS$_OK_RULK = 98417;
! +
!
!       SUCCESS CODES PASSED THRU FROM DRIVERS AND ACP -        BIT 15 = 0
!
! -
literal RMS$_CONTROLC = 67153;          !  TERMINAL I/O ABORTED DUE TO CTRL/C
literal RMS$_CONTROLO = 67081;
!  TERMINAL I/O ABORTED DUE TO CTRL/O
literal RMS$_CONTROLY = 67089;
!  TERMINAL I/O ABORTED DUE TO CTRL/Y
literal RMS$_CREATED = 67097;
!  FILE WAS CREATED, NOT OPENED
literal RMS$_SUPERSEDE = 67121;
!  CREATED FILE SUPERSEDED EXISTING VERSION
literal RMS$_OVRDSKQUOTA = 67177;
!  DISK USAGE EXCEEDS DISK QUOTA
literal RMS$_FILEPURGED = 67193;
!  CREATE FILE CAUSED OLDEST FILE
!  VERSION TO BE PURGED
! +
!
!       WARNING CODES
!
! -
!  BIT 16 = BIT 15 = 1, BIT 14 = 0
literal RMS$_BOF = 98712;
literal RMS$_RNL = 98720;
literal RMS$_RTB = 98728;
literal RMS$_TMO = 98736;
literal RMS$_TNS = 98744;
literal RMS$_BES = 98752;
literal RMS$_PES = 98760;
! +
!
!       ERROR CODES - WITHOUT STV
!
! -
!  BIT 16 = BIT 15 = 1, BIT 14 = 0
literal RMS$_ACT = 98906;
literal RMS$_DEL = 98914;
literal RMS$_INCOMPSHR = 98922;
literal RMS$_DNR = 98930;
literal RMS$_EOF = 98938;
literal RMS$_FEX = 98946;
literal RMS$_FLK = 98954;
literal RMS$_FNF = 98962;
literal RMS$_PRV = 98970;
literal RMS$_REX = 98978;
literal RMS$_RLK = 98986;
literal RMS$_RNF = 98994;
!    (RECORD NEVER WAS IN FILE, OR HAS BEEN DELETED.)
literal RMS$_WLK = 99002;
```

```
0       literal RMS$_EXP = 99010;
0       literal RMS$_NMF = 99018;
0       literal RMS$_SUP = 99026;
0       !  (NOTE THAT SUPPORT HAS REPLACED SUP EXCEPT WHEN RMS CANNOT
0       !  MAP THE DAP STATUS CODE INTO A FAL STATUS CODE.)
0       !  (NOTE ALSO THAT SUP SHOULD HAVE BEEN DEFINED AS A SEVERE
0       !  ERROR CODE. HOWEVER, SUPPORT IS A SEVERE ERROR CODE.)
0       literal RMS$_RSA = 99034;
0       literal RMS$_CRC = 99042;
0       literal RMS$_WCC = 99050;
0       literal RMS$_IDR = 99058;
0       literal RMS$_LWC = 99066;
0       literal RMS$_XCR = 99074;
0       literal RMS$_NOVALPRS = 99082;
0       ! +
0       !
0       !       ERROR CODES - WITH STV ERROR CODE
0       !
0       ! -
0       !  BIT 16 = BIT 15 = BIT 14 = 1
0       literal RMS$_ACC = 114690;
0       literal RMS$_CRE = 114698;
0       literal RMS$_DAC = 114706;
0       literal RMS$_ENT = 114714;
0       literal RMS$_EXT = 114722;
0       literal RMS$_FND = 114730;
0       literal RMS$_MKD = 114738;
0       literal RMS$_DPE = 114746;
0       literal RMS$_SPL = 114754;
0       literal RMS$_DNF = 114762;
0       literal RMS$_JNF = 114770;
0       ! +
0       !
0       !       SEVERE ERROR CODES - WITHOUT STV
0       !
0       ! -
0       !  BIT 16 = BIT 15 = 1, BIT 14 = 0
0       literal RMS$_TEMP0 = 99308;
0       literal RMS$_AID = 99316;
0       literal RMS$_ALN = 99324;
0       literal RMS$_ALQ = 99332;
0       literal RMS$_ANI = 99340;
0       literal RMS$_AOP = 99348;
0       literal RMS$_BKS = 99356;
0       literal RMS$_BKZ = 99364;
0       literal RMS$_BLN = 99372;
0       literal RMS$_BUG = 99380;
0       literal RMS$_BUG_DDI = 99388;
0       literal RMS$_BUG_DAP = 99396;
0       literal RMS$_BUG_XX1 = 99404;
0       literal RMS$_BUG_XX2 = 99412;
0       literal RMS$_BUG_XX3 = 99420;
0       literal RMS$_BUG_XX4 = 99428;
0       literal RMS$_BUG_XX5 = 99436;
0       literal RMS$_BUG_XX6 = 99444;
0       literal RMS$_BUG_XX7 = 99452;
0       literal RMS$_BUG_XX8 = 99460;
```

```
;   0   literal RMS$_BUSY = 99468;
;   0   literal RMS$_CCR = 99476;
;   0   literal RMS$_CHG = 99484;
;   0   literal RMS$_CHK = 99492;
;   0   literal RMS$_COD = 99500;
;   0   literal RMS$_CUR = 99508;
;   0   literal RMS$_DAN = 99516;
;   0   literal RMS$_DEV = 99524;
;   0   literal RMS$_DIR = 99532;
;   0   literal RMS$_DME = 99540;
;   0   literal RMS$_DNA = 99548;
;   0   literal RMS$_DTP = 99556;
;   0   literal RMS$_DUP = 99564;
;   0   literal RMS$_DVI = 99572;
;   0   literal RMS$_ESA = 99580;
;   0   literal RMS$_ESS = 99588;
;   0   literal RMS$_FAB = 99596;
;   0   literal RMS$_FAC = 99604;
;   0   literal RMS$_FLG = 99612;
;   0   literal RMS$_FNA = 99620;
;   0   literal RMS$_FNM = 99628;
;   0   literal RMS$_FSZ = 99636;
;   0   literal RMS$_FOP = 99644;
;   0   literal RMS$_FUL = 99652;
;   0   literal RMS$_IAL = 99660;
;   0   literal RMS$_IAN = 99668;
;   0   literal RMS$_IDX = 99676;
;   0   literal RMS$_IFI = 99684;
;   0   literal RMS$_IMX = 99692;
;   0   literal RMS$_IOP = 99700;
;   0   literal RMS$_IRC = 99708;
;   0   literal RMS$_ISI = 99716;
;   0   literal RMS$_KBF = 99724;
;   0   literal RMS$_KEY = 99732;
;   0   literal RMS$_KRF = 99740;
;   0   literal RMS$_KSZ = 99748;
;   0   literal RMS$_LAN = 99756;
;   0   literal RMS$_TEMP1 = 99764;
;   0   literal RMS$_LNE = 99772;
;   0   literal RMS$_TEMP2 = 99780;
;   0   literal RMS$_MRN = 99788;
;   0   literal RMS$_MRS = 99796;
;   0   literal RMS$_NAM = 99804;
;   0   literal RMS$_NEF = 99812;
;   0   literal RMS$_TEMP3 = 99820;
;   0   literal RMS$_NOD = 99828;
;   0   literal RMS$_NPK = 99836;
;   0   literal RMS$_ORD = 99844;                    ! ! NOT USED AS OF V4
;   0   literal RMS$_ORG = 99852;
;   0   literal RMS$_PBF = 99860;
;   0   literal RMS$_PLG = 99868;
;   0   literal RMS$_POS = 99876;
;   0   literal RMS$_TEMP4 = 99884;
;   0   literal RMS$_QUO = 99892;
;   0   literal RMS$_RAB = 99900;
;   0   literal RMS$_RAC = 99908;
;   0   literal RMS$_RAT = 99916;
```

```
;  0      literal RMS$_RBF = 99924;
;  0      literal RMS$_RFA = 99932;
;  0      literal RMS$_RFM = 99940;
;  0      literal RMS$_RHB = 99948;
;  0      literal RMS$_RLF = 99956;
;  0      literal RMS$_ROP = 99964;
;  0      literal RMS$_RRV = 99972;
;  0      literal RMS$_RVU = 99980;
;  0      literal RMS$_RSS = 99988;
;  0      literal RMS$_RST = 99996;
;  0      literal RMS$_RSZ = 100004;
;  0      literal RMS$_SEQ = 100012;
;  0      literal RMS$_SHR = 100020;
;  0      literal RMS$_SIZ = 100028;
;  0      literal RMS$_SQO = 100036;
;  0      literal RMS$_TEMP5 = 100044;
;  0      literal RMS$_SYN = 100052;
;  0      literal RMS$_TRE = 100060;
;  0      literal RMS$_TYP = 100068;
;  0      literal RMS$_UBF = 100076;
;  0      literal RMS$_USZ = 100084;
;  0      literal RMS$_VER = 100092;
;  0      literal RMS$_XNF = 100100;
;  0      literal RMS$_XAB = 100108;
;  0      literal RMS$_ESL = 100116;
;  0      literal RMS$_TEMP6 = 100124;
;  0      !   (THIS CODE USED TO BE WSF)
;  0      !   WSF          <working set full (cannot lock buffers in working set)>          !! NOT USED AS OF V2
;  0      literal RMS$_ENV = 100132;
;  0      literal RMS$_PLV = 100140;
;  0      literal RMS$_MBC = 100148;
;  0      literal RMS$_RSL = 100156;
;  0      literal RMS$_WLD = 100164;
;  0      literal RMS$_NET = 100172;
;  0      !   (NOTE THAT NETFAIL HAS REPLACED NET EXCEPT WHEN RMS CANNOT
;  0      !   MAP THE DAP STATUS CODE INTO A FAL STATUS CODE.)
;  0      literal RMS$_IBF = 100180;
;  0      literal RMS$_REF = 100188;
;  0      literal RMS$_IFL = 100196;
;  0      literal RMS$_DFL = 100204;
;  0      literal RMS$_KNM = 100212;
;  0      literal RMS$_IBK = 100220;
;  0      literal RMS$_KSI = 100228;
;  0      literal RMS$_LEX = 100236;
;  0      literal RMS$_SEG = 100244;
;  0      literal RMS$_SNE = 100252;                    !! NOT USED AS OF V4
;  0      literal RMS$_SPE = 100260;                    !! NOT USED AS OF V4
;  0      literal RMS$_UPI = 100268;
;  0      literal RMS$_ACS = 100276;
;  0      literal RMS$_STR = 100284;
;  0      literal RMS$_FTM = 100292;
;  0      literal RMS$_GBC = 100300;
;  0      literal RMS$_DEADLOCK = 100308;
;  0      literal RMS$_EXENQLM = 100316;
;  0      literal RMS$_JOP = 100324;
;  0      literal RMS$_RUM = 100332;
;  0      literal RMS$_JNS = 100340;
```

```
0       literal RMS$_NRU = 100348;
0       literal RMS$_IFF = 100356;               ! +
0       !
0       !       SEVERE ERRORS - WITH STV ERROR CODE
0       !
0       !
0       ! -
0       !  BIT 16 = BIT 15 = BIT 14 = 1
0       literal RMS$_ATR = 114892;
0       literal RMS$_ATW = 114900;
0       literal RMS$_CCF = 114908;
0       literal RMS$_CDA = 114916;
0       literal RMS$_CHN = 114924;
0       literal RMS$_RER = 114932;
0       literal RMS$_RMV = 114940;
0       literal RMS$_RPL = 114948;
0       literal RMS$_SYS = 114956;
0       literal RMS$_WER = 114964;
0       literal RMS$_WPL = 114972;
0       literal RMS$_IFA = 114980;
0       literal RMS$_WBE = 114988;
0       literal RMS$_ENQ = 114996;
0       literal RMS$_NETFAIL = 115004;
0       literal RMS$_SUPPORT = 115012;
0       literal RMS$_CRMP = 115020;
0       literal RMS$_NOJ = 115028;
0       literal RMS$_REENT = 115036;
0       literal RMS$_CJF = 115044;
0       literal RMS$_TMR = 115052;
0       !
0       ! ****************************************************************
0       ! *                                                              *
0       ! *  Copyright (c) 1982, 1983                                    *
0       ! *  by DIGITAL Equipment Corporation, Maynard, Mass.            *
0       ! *                                                              *
0       ! *  This software is furnished under a license and may be used and  copied  *
0       ! *  only  in  accordance  with  the  terms  of  such  license and with the  *
0       ! *  inclusion of the above copyright notice.  This software or  any  other  *
0       ! *  copies  thereof may not be provided or otherwise made available to any  *
0       ! *  other person.  No title to and ownership of  the  software  is  hereby  *
0       ! *  transferred.                                                *
0       ! *                                                              *
0       ! *  The information in this software is subject to change  without  notice  *
0       ! *  and  should  not  be  construed  as  a commitment by DIGITAL Equipment  *
0       ! *  Corporation.                                                *
0       ! *                                                              *
0       ! *  DIGITAL assumes no responsibility for the use or  reliability  of  its  *
0       ! *  software on equipment which is not supplied by DIGITAL.     *
0       ! *                                                              *
0       ! ****************************************************************
0       ! ****************************************************************************************
0       !  Created 15-SEP-1984 22:55:02  by  VAX-11 SDL V2.0       Source: 15-SEP-1984 22:49:39 _$255$DUA28:[RMS.SRC]RMSUSR.
0       ! ****************************************************************************************
0
0
0       !*** MODULE $FABDEF ***
0       ! ++++++++++
0       !    the fields thru ctx must not be modified due to
```

```
0       !    commonality between fab/rab/xab
0       literal FAB$C_BID = 3;                  ! code for fab
0       literal FAB$M_PPF_RAT = 16320;
0       literal FAB$M_PPF_IND = 16384;
0       literal FAB$M_MXV = 2;
0       literal FAB$M_SUP = 4;
0       literal FAB$M_TMP = 8;
0       literal FAB$M_TMD = 16;
0       literal FAB$M_DFW = 32;
0       literal FAB$M_SQO = 64;
0       literal FAB$M_RWO = 128;
0       literal FAB$M_POS = 256;
0       literal FAB$M_WCK = 512;
0       literal FAB$M_NEF = 1024;
0       literal FAB$M_RWC = 2048;
0       literal FAB$M_DMO = 4096;
0       literal FAB$M_SPL = 8192;
0       literal FAB$M_SCF = 16384;
0       literal FAB$M_DLT = 32768;
0       literal FAB$M_NFS = 65536;
0       literal FAB$M_UFO = 131072;
0       literal FAB$M_PPF = 262144;
0       literal FAB$M_INP = 524288;
0       literal FAB$M_CTG = 1048576;
0       literal FAB$M_CBT = 2097152;
0       literal FAB$M_RCK = 8388608;
0       literal FAB$M_NAM = 16777216;
0       literal FAB$M_CIF = 33554432;
0       literal FAB$M_ESC = 134217728;
0       literal FAB$M_TEF = 268435456;
0       literal FAB$M_OFP = 536870912;
0       literal FAB$M_KFO = 1073741824;
0       literal FAB$M_PUT = 1;
0       literal FAB$M_GET = 2;
0       literal FAB$M_DEL = 4;
0       literal FAB$M_UPD = 8;
0       literal FAB$M_TRN = 16;
0       literal FAB$M_BIO = 32;
0       literal FAB$M_BRO = 64;
0       literal FAB$M_EXE = 128;
0       literal FAB$M_SHRPUT = 1;
0       literal FAB$M_SHRGET = 2;
0       literal FAB$M_SHRDEL = 4;
0       literal FAB$M_SHRUPD = 8;
0       literal FAB$M_MSE = 16;
0       literal FAB$M_NIL = 32;
0       literal FAB$M_UPI = 64;
0       literal FAB$C_SEQ = 0;                  ! sequential
0       literal FAB$C_REL = 16;                 ! relative
0       literal FAB$C_IDX = 32;                 ! indexed
0       literal FAB$C_HSH = 48;                 ! hashed
0       literal FAB$M_FTN = 1;
0       literal FAB$M_CR = 2;
0       literal FAB$M_PRN = 4;
0       literal FAB$M_BLK = 8;
0       literal FAB$C_RFM_DFLT = 2;             ! var len is default
0       literal FAB$C_UDF = 0;                  ! undefined (also stream binary)
```

```
0       literal FAB$C_FIX = 1;                          ! fixed length records
0       literal FAB$C_VAR = 2;                          ! variable length records
0       literal FAB$C_VFC = 3;                          ! variable fixed control
0       literal FAB$C_STM = 4;                          ! RMS-11 stream (valid only for sequential org)
0       literal FAB$C_STMLF = 5;                        ! LF stream (valid only for sequential org)
0       literal FAB$C_STMCR = 6;                        ! CR stream (valid only for sequential org)
0       literal FAB$C_MAXRFM = 6;                       ! maximum rfm supported
0       literal FAB$M_RU = 1;
0       literal FAB$M_AI = 2;
0       literal FAB$M_BI = 4;
0       literal FAB$K_BLN = 80;                         ! length of fab
0       literal FAB$C_BLN = 80;                         ! length of fab
0       literal FAB$S_FABDEF = 80;
0       macro FAB$B_BID = 0,0,8,0 %;                    ! block id
0       macro FAB$B_BLN = 1,0,8,0 %;                    ! block len
0       macro FAB$R_IFI_OVERLAY = 2,0,16,0 %;
0       macro FAB$W_IFI = 2,0,16,0 %;                   ! internal file index
0       macro FAB$R_IFI_BITS = 2,0,16,0 %;
0       macro FAB$V_PPF_RAT = 2,6,8,0 %;
0       literal FAB$S_PPF_RAT = 8;                      ! rat value for process-permanent files
0       macro FAB$V_PPF_IND = 2,14,1,0 %;               ! indirect access to process-permanent file
0       !  (i.e., restricted operations)
0       macro FAB$R_FOP_OVERLAY = 4,0,32,0 %;
0       macro FAB$L_FOP = 4,0,32,0 %;                   ! file options
0       macro FAB$R_FOP_BITS = 4,0,32,0 %;
0       macro FAB$V_MXV = 4,1,1,0 %;                    ! maximize version number
0       macro FAB$V_SUP = 4,2,1,0 %;                    ! supersede existing file
0       macro FAB$V_TMP = 4,3,1,0 %;                    ! create temporary file
0       macro FAB$V_TMD = 4,4,1,0 %;                    ! create temp file marked for delete
0       macro FAB$V_DFW = 4,5,1,0 %;                    ! deferred write (rel and idx)
0       macro FAB$V_SQO = 4,6,1,0 %;                    ! sequential access only
0       macro FAB$V_RWO = 4,7,1,0 %;                    ! rewind mt on open
0       macro FAB$V_POS = 4,8,1,0 %;                    ! use next magtape position
0       macro FAB$V_WCK = 4,9,1,0 %;                    ! write checking
0       macro FAB$V_NEF = 4,10,1,0 %;                   ! inhibit end of file positioning
0       macro FAB$V_RWC = 4,11,1,0 %;                   ! rewind mt on close
0       macro FAB$V_DMO = 4,12,1,0 %;                   ! dismount mt on close (not implemented)
0       macro FAB$V_SPL = 4,13,1,0 %;                   ! spool file on close
0       macro FAB$V_SCF = 4,14,1,0 %;                   ! submit command file on close
0       macro FAB$V_DLT = 4,15,1,0 %;                   ! delete sub-option
0       macro FAB$V_NFS = 4,16,1,0 %;                   ! non-file structured operation
0       macro FAB$V_UFO = 4,17,1,0 %;                   ! user file open - no rms operations
0       macro FAB$V_PPF = 4,18,1,0 %;                   ! process permanent file (pio segment)
0       macro FAB$V_INP = 4,19,1,0 %;                   ! process-permanent file is 'input'
0       macro FAB$V_CTG = 4,20,1,0 %;                   ! contiguous extension
0       macro FAB$V_CBT = 4,21,1,0 %;                   ! contiguous best try
0       macro FAB$V_RCK = 4,23,1,0 %;                   ! read checking
0       macro FAB$V_NAM = 4,24,1,0 %;                   ! use name block dvi, did, and/or fid fields for open
0       macro FAB$V_CIF = 4,25,1,0 %;                   ! create if non-existent
0       macro FAB$V_ESC = 4,27,1,0 %;                   ! 'escape' to non-standard function ($modify)
0       macro FAB$V_TEF = 4,28,1,0 %;                   ! truncate at eof on close (write-accessed seq. disk file only)
0       macro FAB$V_OFP = 4,29,1,0 %;                   ! output file parse (only name type sticky)
0       macro FAB$V_KFO = 4,30,1,0 %;                   ! known file open (image activator only release 1)
0       macro FAB$L_STS = 8,0,32,0 %;                   ! status
0       macro FAB$L_STV = 12,0,32,0 %;                  ! status value
0       macro FAB$L_ALQ = 16,0,32,0 %;                  ! allocation quantity
0       macro FAB$W_DEQ = 20,0,16,0 %;                  ! default allocation quantity
```

```
0    macro FAB$R_FAC_OVERLAY = 22,0,8,0 %;
0    macro FAB$B_FAC = 22,0,8,0 %;                 !  file access
0    macro FAB$R_FAC_BITS = 22,0,8,0 %;
0    macro FAB$V_PUT = 22,0,1,0 %;                 !  put access
0    macro FAB$V_GET = 22,1,1,0 %;                 !  get access
0    macro FAB$V_DEL = 22,2,1,0 %;                 !  delete access
0    macro FAB$V_UPD = 22,3,1,0 %;                 !  update access
0    macro FAB$V_TRN = 22,4,1,0 %;                 !  truncate access
0    macro FAB$V_BIO = 22,5,1,0 %;                 !  block i/o access
0    macro FAB$V_BRO = 22,6,1,0 %;                 !  block and record i/o access
0    macro FAB$V_EXE = 22,7,1,0 %;                 !  execute access (caller must be exec or kernel mode,
0    !   ufo must also be set)
0    macro FAB$R_SHR_OVERLAY = 23,0,8,0 %;
0    macro FAB$B_SHR = 23,0,8,0 %;                 !  file sharing
0    macro FAB$R_SHR_BITS = 23,0,8,0 %;
0    macro FAB$V_SHRPUT = 23,0,1,0 %;              !  put access
0    macro FAB$V_SHRGET = 23,1,1,0 %;              !  get access
0    macro FAB$V_SHRDEL = 23,2,1,0 %;              !  delete access
0    macro FAB$V_SHRUPD = 23,3,1,0 %;              !  update access
0    macro FAB$V_MSE = 23,4,1,0 %;                 !  multi-stream connects enabled
0    macro FAB$V_NIL = 23,5,1,0 %;                 !  no sharing
0    macro FAB$V_UPI = 23,6,1,0 %;                 !  user provided interlocking (allows multiple
0    !   writers to seq. files)
0    macro FAB$L_CTX = 24,0,32,0 %;                !  user context
0    ! -----*****
0    macro FAB$B_RTV = 28,0,8,1 %;                 !  retrieval window size
0    macro FAB$R_ORG_OVERLAY = 29,0,8,0 %;
0    macro FAB$B_ORG = 29,0,8,0 %;                 !  file organization
0    macro FAB$R_ORG_BITS = 29,0,8,0 %;
0    macro FAB$V_ORG = 29,4,4,0 %;
0    literal FAB$S_ORG = 4;
0    macro FAB$R_RAT_OVERLAY = 30,0,8,0 %;
0    macro FAB$B_RAT = 30,0,8,0 %;                 !  record format
0    macro FAB$R_RAT_BITS = 30,0,8,0 %;
0    macro FAB$V_FTN = 30,0,1,0 %;                 !  fortran carriage-ctl
0    macro FAB$V_CR = 30,1,1,0 %;                  !  lf-record-cr carriage ctl
0    macro FAB$V_PRN = 30,2,1,0 %;                 !  print-file carriage ctl
0    macro FAB$V_BLK = 30,3,1,0 %;                 !  records don't cross block boundaries
0    macro FAB$B_RFM = 31,0,8,0 %;                 !  record format
0    macro FAB$L_JNL = 32,0,32,0 %;                !  lcb address
0    macro FAB$L_XAB = 36,0,32,0 %;                !  xab address
0    macro FAB$L_NAM = 40,0,32,0 %;                !  nam block address
0    macro FAB$L_FNA = 44,0,32,0 %;                !  file name string address
0    macro FAB$L_DNA = 48,0,32,0 %;                !  default file name string addr
0    macro FAB$B_FNS = 52,0,8,0 %;                 !  file name string size
0    macro FAB$B_DNS = 53,0,8,0 %;                 !  default name string size
0    macro FAB$W_MRS = 54,0,16,0 %;                !  maximum record size
0    macro FAB$L_MRN = 56,0,32,0 %;                !  maximum record number
0    macro FAB$W_BLS = 60,0,16,0 %;                !  blocksize for tape
0    macro FAB$B_BKS = 62,0,8,0 %;                 !  bucket size
0    macro FAB$B_FSZ = 63,0,8,0 %;                 !  fixed header size
0    macro FAB$L_DEV = 64,0,32,0 %;                !  device characteristics
0    macro FAB$L_SDC = 68,0,32,0 %;                !  spooling device characteristics
0    macro FAB$W_GBC = 72,0,16,0 %;                !  Global buffer count
0    macro FAB$R_ACMODES_OVERLAY = 74,0,8,0 %;
0    macro FAB$B_ACMODES = 74,0,8,0 %;             !  agent access modes
0    macro FAB$R_ACMODES_BITS = 74,0,8,0 %;
```

```
0   macro FAB$V_LNM_MODE = 74,0,2,0 %;
0   literal FAB$S_LNM_MODE = 2;              !  ACMODE for log nams
0   macro FAB$V_CHAN_MODE = 74,2,2,0 %;
0   literal FAB$S_CHAN_MODE = 2;             !  ACMODE for channel
0   macro FAB$V_FILE_MODE = 74,4,2,0 %;
0   literal FAB$S_FILE_MODE = 2;             !  ACMODE to use for determining file accessibility
0   macro FAB$R_RCF_OVERLAY = 75,0,8,0 %;    !  recovery control flags
0   macro FAB$B_RCF = 75,0,8,0 %;
0   macro FAB$R_RCF_BITS = 75,0,8,0 %;
0   macro FAB$V_RU = 75,0,1,0 %;             !  recovery unit recovery
0   macro FAB$V_AI = 75,1,1,0 %;             !  after image recovery
0   macro FAB$V_BI = 75,2,1,0 %;             !  before image recovery
0
C   !*** MODULE $RABDEF ***
0   !
0   !           record access block (rab) definitions
0   !
0   !   there is one rab per connected stream
0   !   it is used for all communications between the user
0   !   and rms concerning operations on the stream
0   !
0   ! ++++++++++
0   !   the fields thru ctx cannot be changed due to commonality
0   !   with the fab
0   !
0   literal RAB$C_BID = 1;                   !  code for rab
0   literal RAB$M_PPF_RAT = 16320;
0   literal RAB$M_PPF_IND = 16384;
0   literal RAB$M_ASY = 1;
0   literal RAB$M_TPT = 2;
0   literal RAB$M_REA = 4;
0   literal RAB$M_RRL = 8;
0   literal RAB$M_UIF = 16;
0   literal RAB$M_MAS = 32;
0   literal RAB$M_FDL = 64;
0   literal RAB$M_HSH = 128;
0   literal RAB$M_EOF = 256;
0   literal RAB$M_RAH = 512;
0   literal RAB$M_WBH = 1024;
0   literal RAB$M_BIO = 2048;
0   literal RAB$M_LV2 = 4096;
0   literal RAB$M_LOA = 8192;
0   literal RAB$M_LIM = 16384;
0   literal RAB$M_LOC = 65536;
0   literal RAB$M_WAT = 131072;
0   literal RAB$M_ULK = 262144;
0   literal RAB$M_RLK = 524288;
0   literal RAB$M_NLK = 1048576;
0   literal RAB$M_KGE = 2097152;
0   literal RAB$M_KGT = 4194304;
0   literal RAB$M_NXR = 8388608;
0   literal RAB$M_RNE = 16777216;
0   literal RAB$M_TMO = 33554432;
0   literal RAB$M_CVT = 67108864;
0   literal RAB$M_RNF = 134217728;
0   literal RAB$M_ETO = 268435456;
0   literal RAB$M_PTA = 536870912;
```

```
0      literal RAB$M_PMT = 1073741824;
0      literal RAB$M_CCO = -2147483648;
0      literal RAB$C_SEQ = 0;                       ! sequential access
0      literal RAB$C_KEY = 1;                       ! keyed access
0      literal RAB$C_RFA = 2;                       ! rfa access
0      literal RAB$C_STM = 3;                       ! stream access (valid only for sequential org)
0      literal RAB$K_BLN = 68;                      ! length of rab
0      literal RAB$C_BLN = 68;                      ! length of rab
0      literal RAB$S_RABDEF = 68;
0      macro RAB$B_BID = 0,0,8,0 %;                 ! block id
0      macro RAB$B_BLN = 1,0,8,0 %;                 ! block length
0      macro RAB$R_ISI_OVERLAY = 2,0,16,0 %;
0      macro RAB$W_ISI = 2,0,16,0 %;                ! internal stream index
0      !  (ifi in fab)
0      macro RAB$R_ISI_BITS = 2,0,16,0 %;
0      macro RAB$V_PPF_RAT = 2,6,8,0 %;
0      literal RAB$S_PPF_RAT = 8;                   ! rat value for process-permanent files
0      macro RAB$V_PPF_IND = 2,14,1,0 %;            ! indirect access to process-permanent file
0      !  (i.e., restricted operations)
0      macro RAB$R_ROP_OVERLAY = 4,0,32,0 %;
0      macro RAB$L_ROP = 4,0,32,0 %;                ! record options
0      macro RAB$R_ROP_BITS0 = 4,0,32,0 %;
0      macro RAB$V_ASY = 4,0,1,0 %;                 ! asynchronous operations
0      macro RAB$V_TPT = 4,1,1,0 %;                 ! truncate put - allow sequential put not at
0      macro RAB$V_REA = 4,2,1,0 %;                 ! lock record for read only, allow other readers
0      macro RAB$V_RRL = 4,3,1,0 %;                 ! read record regardless of lock
0      macro RAB$V_UIF = 4,4,1,0 %;                 ! update if existent
0      macro RAB$V_MAS = 4,5,1,0 %;                 ! mass-insert mode
0      macro RAB$V_FDL = 4,6,1,0 %;                 ! fast record deletion
0      macro RAB$V_HSH = 4,7,1,0 %;                 ! use hash code in bkt
0      macro RAB$V_EOF = 4,8,1,0 %;                 ! connect to eof
0      macro RAB$V_RAH = 4,9,1,0 %;                 ! read ahead
0      macro RAB$V_WBH = 4,10,1,0 %;                ! write behind
0      macro RAB$V_BIO = 4,11,1,0 %;                ! connect for bio only
0      macro RAB$V_LV2 = 4,12,1,0 %;                ! level 2 RU lock consistancy
0      macro RAB$V_LOA = 4,13,1,0 %;                ! use bucket fill percentage
0      macro RAB$V_LIM = 4,14,1,0 %;                ! compare for key limit reached on $get/$find seq. (idx only)
0      !
0      !  the following bits are input to
0      !  $find or $get, (see above also REA and RRL)
0      !  (separate byte)
0      !
0      macro RAB$V_LOC = 4,16,1,0 %;                ! use locate mode
0      macro RAB$V_WAT = 4,17,1,0 %;                ! wait if record not available
0      macro RAB$V_ULK = 4,18,1,0 %;                ! manual unlocking
0      macro RAB$V_RLK = 4,19,1,0 %;                ! allow readers for this locked record
0      macro RAB$V_NLK = 4,20,1,0 %;                ! do not lock record
0      macro RAB$V_KGE = 4,21,1,0 %;                ! key > or =
0      macro RAB$V_KGT = 4,22,1,0 %;                ! key greater than
0      macro RAB$V_NXR = 4,23,1,0 %;                ! get non-existent record
0      macro RAB$V_RNE = 4,24,1,0 %;                ! read no echo
0      macro RAB$V_TMO = 4,25,1,0 %;                ! use time-out period
0      macro RAB$V_CVT = 4,26,1,0 %;                ! convert to upper case
0      macro RAB$V_RNF = 4,27,1,0 %;                ! read no filter
0      macro RAB$V_ETO = 4,28,1,0 %;                ! extended terminal operation
0      macro RAB$V_PTA = 4,29,1,0 %;                ! purge type ahead
0      macro RAB$V_PMT = 4,30,1,0 %;                ! use prompt buffer
```

```
macro RAB$V_CCO = 4,31,1,0 %;              !  cancel control o on output
!
!   the following bits are terminal qualifiers only
!   (separate byte)
!
!
!
!     eof, thus truncating file (seq. org only)
!
!   these next two should be in the byte for bits
!   input to $find or $get, but there is no room there
!
!   the following bits may be
!   input to various rab-related
!   operations
macro RAB$R_ROP_FIELDS = 4,0,32,0 %;
macro RAB$B_ROPT = 5,0,8,0 %;              !  various options
macro RAB$B_ROP2 = 6,0,8,0 %;              !  get/find options (use of this field discouraged
!  due to REA and RRL being in a different byte)
macro RAB$B_ROP3 = 7,0,8,0 %;              !  terminal read options
!
macro RAB$L_STS = 8,0,32,0 %;              !  status
macro RAB$R_STV_OVERLAY = 12,0,32,0 %;
macro RAB$L_STV = 12,0,32,0 %;             !  status value
macro RAB$R_STV_FIELDS = 12,0,32,0 %;
macro RAB$W_STV0 = 12,0,16,0 %;            !  low word of stv
macro RAB$W_STV2 = 14,0,16,0 %;            !  high word of stv
macro RAB$R_RFA_OVERLAY = 16,0,0,0 %;
macro RAB$W_RFA = 16,0,0,0 %;
literal RAB$S_RFA = 6;                     !  record's file address
macro RAB$R_RFA_FIELDS = 16,0,0,0 %;
macro RAB$L_RFA0 = 16,0,32,0 %;
macro RAB$W_RFA4 = 20,0,16,0 %;
!   to the rfa field to be a move quad, overwriting
!   this reserved word)
macro RAB$L_CTX = 24,0,32,0 %;             !  user context
! -----*****
macro RAB$B_RAC = 30,0,8,0 %;              !  record access
macro RAB$B_TMO = 31,0,8,0 %;              !  time-out period
macro RAB$W_USZ = 32,0,16,0 %;             !  user buffer size
macro RAB$W_RSZ = 34,0,16,0 %;             !  record buffer size
macro RAB$L_UBF = 36,0,32,0 %;             !  user buffer address
macro RAB$L_RBF = 40,0,32,0 %;             !  record buffer address
macro RAB$L_RHB = 44,0,32,0 %;             !  record header buffer addr
macro RAB$R_KBF_OVERLAY = 48,0,32,0 %;
macro RAB$L_KBF = 48,0,32,0 %;             !  key buffer address
macro RAB$L_PBF = 48,0,32,0 %;             !  prompt buffer addr
macro RAB$R_KSZ_OVERLAY = 52,0,8,0 %;
macro RAB$B_KSZ = 52,0,8,0 %;              !  key buffer size
macro RAB$B_PSZ = 52,0,8,0 %;              !  prompt buffer size
macro RAB$B_KRF = 53,0,8,0 %;              !  key of reference
macro RAB$B_MBF = 54,0,8,1 %;              !  multi-buffer count
macro RAB$B_MBC = 55,0,8,0 %;              !  multi-block count
macro RAB$R_BKT_OVERLAY = 56,0,32,0 %;
macro RAB$L_BKT = 56,0,32,0 %;             !  bucket hash code, vbn, or rrn
macro RAB$L_DCT = 56,0,32,0 %;             !  duplicates count on key accessed on alternate key
```

```
0       macro RAB$L_FAB = 60,0,32,0 %;           !  related fab for connect
0       macro RAB$L_XAB = 64,0,32,0 %;           !  XAB address
0
0       !*** MODULE $NAMDEF ***
0       !
0       !           name block field definitions
0       !
0       !   the nam block is used to communicate optional
0       !   filename-related information
0       !
0       literal NAM$C_BID = 2;                   !  code for nam block
0       literal NAM$C_MAXRSS = 255;              !  maximum resultant name string size (network)
0       literal NAM$C_MAXRSSLCL = 255;           !  maximum resultant name string size (local)
0       literal NAM$M_PWD = 1;
0       literal NAM$M_FILL_1 = 2;
0       literal NAM$M_FILL_2 = 4;
0       literal NAM$M_SYNCHK = 8;
0       literal NAM$M_NOCONCEAL = 16;
0       literal NAM$M_SLPARSE = 32;
0       literal NAM$M_SRCHXABS = 64;
0       literal NAM$C_UFS = 0;                   !  Unknown file system for remote file access or
0       !   not applicable for local file access or
0       !   not applicable for task-to-task operation
0       literal NAM$C_RMS11 = 1;                 !  RMS-11
0       literal NAM$C_RMS20 = 2;                 !  RMS-20
0       literal NAM$C_RMS32 = 3;                 !  RMS-32
0       literal NAM$C_FCS11 = 4;                 !  FCS-11
0       literal NAM$C_RT11FS = 5;                !  RT-11 file system
0       literal NAM$C_TOPS20FS = 7;              !  TOPS-20 file system
0       literal NAM$C_TOPS10FS = 8;              !  TOPS-10 file system
0       literal NAM$C_RMS32S = 10;               !  RMS-32 subset (e.g., VAXELAN)
0       !  ++++++++++++++++++++++++
0       !   the following 3 fields must not be rearranged relative to each other
0       !
0       literal NAM$C_DVI = 16;                  !  length of dvi field
0       !  ++++++++++
0       !   the location of the following fields must not
0       !   be changed due to their commonality with the fib
0       literal NAM$M_IFI = 65536;
0       literal NAM$M_SRCHNMF = 1073741824;
0       literal NAM$M_SVCTX = -2147483648;
0       literal NAM$K_BLN_V2 = 56;               !  Version 2 name block length
0       literal NAM$C_BLN_V2 = 56;               !  Version 2 name block length
0       literal NAM$M_EXP_VER = 1;
0       literal NAM$M_EXP_TYPE = 2;
0       literal NAM$M_EXP_NAME = 4;
0       literal NAM$M_WILD_VER = 8;
0       literal NAM$M_WILD_TYPE = 16;
0       literal NAM$M_WILD_NAME = 32;
0       literal NAM$M_EXP_DIR = 64;
0       literal NAM$M_EXP_DEV = 128;
0       literal NAM$M_WILDCARD = 256;
0       literal NAM$M_SEARCH_LIST = 2048;
0       literal NAM$M_CNCL_DEV = 4096;
0       literal NAM$M_ROOT_DIR = 8192;
0       literal NAM$M_LOWVER = 16384;
0       literal NAM$M_HIGHVER = 32768;
```

```
0    literal NAM$M_PPF = 65536;
0    literal NAM$M_NODE = 131072;
0    literal NAM$M_QUOTED = 262144;
0    literal NAM$M_GRP_MBR = 524288;
0    literal NAM$M_WILD_DIR = 1048576;
0    literal NAM$M_DIR_LVLS = 14680064;
0    literal NAM$M_WILD_UFD = 16777216;
0    literal NAM$M_WILD_SFD1 = 33554432;
0    literal NAM$M_WILD_SFD2 = 67108864;
0    literal NAM$M_WILD_SFD3 = 134217728;
0    literal NAM$M_WILD_SFD4 = 268435456;
0    literal NAM$M_WILD_SFD5 = 536870912;
0    literal NAM$M_WILD_SFD6 = 1073741824;
0    literal NAM$M_WILD_SFD7 = -2147483648;
0    literal NAM$M_WILD_GRP = 16777216;
0    literal NAM$M_WILD_MBR = 33554432;
0    literal NAM$K_BLN_DIRWC = 96;              ! Not documented optional length
0    literal NAM$C_BLN_DIRWC = 96;              ! Not documented optional length
0    literal NAM$K_BLN = 96;                    ! Name block length
0    literal NAM$C_BLN = 96;                    ! Name block length
0    literal NAM$S_NAMDEF = 96;
0    macro NAM$B_BID = 0,0,8,0 %;               ! block id
0    macro NAM$B_BLN = 1,0,8,0 %;               ! block length
0    ! ++++++++++++++++++++++
0    !    the following 3 fields must not be rearranged relative to each other
0    !
0    macro NAM$B_RSS = 2,0,8,0 %;               ! resultant string area size
0    macro NAM$B_RSL = 3,0,8,0 %;               ! resultant string length
0    macro NAM$L_RSA = 4,0,32,0 %;              ! resultant string area address
0    ! ----------------------
0    macro NAM$R_NOP_OVERLAY = 8,0,8,0 %;
0    macro NAM$B_NOP = 8,0,8,0 %;               ! Name options
0    macro NAM$R_NOP_BITS = 8,0,8,0 %;
0    macro NAM$V_PWD = 8,0,1,0 %;               ! Return password if present in nodespec string and any
0    macro NAM$V_FILL_1 = 8,1,1,0 %;            ! unused.  (used to be undocumented ROD)
0    macro NAM$V_FILL_2 = 8,2,1,0 %;            ! unused.  (used to be undocumented SOD)
0    macro NAM$V_SYNCHK = 8,3,1,0 %;            ! Only do syntax check on $parse operation
0    macro NAM$V_NOCONCEAL = 8,4,1,0 %;         ! Do not conceal device/root directory
0    macro NAM$V_SLPARSE = 8,5,1,0 %;           ! Parse search list (not documented) -- used by BACKUP.
0    macro NAM$V_SRCHXABS = 8,6,1,0 %;          ! Fill in attached XABS on $SEARCH operations over the
0    !    network (not documented) -- used by directory.
0    !   other task-specific data of the form /netacp_data"
0    !    (default is to mask out password from expanded and
0    !    resultant name strings and to create a logical name
0    !    whose equivalence string is the unaltered nodespec)
0    macro NAM$B_RFS = 9,0,8,0 %;               ! Remote file system type (currently not documented)
0    ! Note: This field is reserved for use by Digital
0    macro NAM$B_ESS = 10,0,8,0 %;              ! espanded string area size
0    macro NAM$B_ESL = 11,0,8,0 %;              ! expanded string length
0    macro NAM$L_ESA = 12,0,32,0 %;             ! expanded string area address
0    ! ----------------------
0    macro NAM$L_RLF = 16,0,32,0 %;             ! related file nam block addr
0    macro NAM$T_DVI = 20,0,0,0 %;
0    literal NAM$S_DVI = 16;                    ! device id
0    macro NAM$R_FID_OVERLAY = 36,0,0,0 %;
0    macro NAM$W_FID = 36,0,0,0 %;
0    literal NAM$S_FID = 6;                     ! file id
```

```
0    macro NAM$R_FID_FIELDS = 36,0,0,0 %;
0    macro NAM$W_FID_NUM = 36,0,16,0 %;          ! file number
0    macro NAM$W_FID_SEQ = 38,0,16,0 %;          ! sequence number
0    macro NAM$R_FID_RVN_OVERLAY = 40,0,16,0 %;
0    macro NAM$W_FID_RVN = 40,0,16,0 %;          ! relative volume number
0    macro NAM$R_FID_RVN_FIELDS = 40,0,16,0 %;
0    macro NAM$B_FID_RVN = 40,0,8,0 %;           ! alternate format RVN
0    macro NAM$B_FID_NMX = 41,0,8,0 %;           ! alternate format file number extension
0    macro NAM$R_DID_OVERLAY = 42,0,0,0 %;
0    macro NAM$W_DID = 42,0,0,0 %;
0    literal NAM$S_DID = 6;                      ! directory id
0    macro NAM$R_DID_FIELDS = 42,0,0,0 %;
0    macro NAM$W_DID_NUM = 42,0,16,0 %;          ! file number
0    macro NAM$W_DID_SEQ = 44,0,16,0 %;          ! sequence number
0    macro NAM$R_DID_RVN_OVERLAY = 46,0,16,0 %;
0    macro NAM$W_DID_RVN = 46,0,16,0 %;          ! relative volume number
0    macro NAM$R_DID_RVN_FIELDS = 46,0,16,0 %;
0    macro NAM$B_DID_RVN = 46,0,8,0 %;           ! alternate format RVN
0    macro NAM$B_DID_NMX = 47,0,8,0 %;           ! alternate format file number extension
0    macro NAM$R_WCC_OVERLAY = 48,0,32,0 %;
0    macro NAM$L_WCC = 48,0,32,0 %;              ! wild card context
0    macro NAM$R_WCC_BITS = 48,0,32,0 %;
0    macro NAM$V_IFI = 48,16,1,0 %;              ! the first word contains an IFI
0    macro NAM$V_SRCHNMF = 48,30,1,0 %;          ! no-more-files has been encountered on a search
0    macro NAM$V_SVCTX = 48,31,1,0 %;            ! save context across search calls
0    macro NAM$R_FNB_OVERLAY = 52,0,32,0 %;
0    macro NAM$L_FNB = 52,0,32,0 %;              ! file name status bits
0    macro NAM$R_FNB_BITS0 = 52,0,24,0 %;
0    macro NAM$V_EXP_VER = 52,0,1,0 %;           ! version was explicit
0    macro NAM$V_EXP_TYPE = 52,1,1,0 %;          ! type was explicit
0    macro NAM$V_EXP_NAME = 52,2,1,0 %;          ! name was explicit
0    macro NAM$V_WILD_VER = 52,3,1,0 %;          ! version contained a wild card
0    macro NAM$V_WILD_TYPE = 52,4,1,0 %;         ! type contained a wild card
0    macro NAM$V_WILD_NAME = 52,5,1,0 %;         ! name contained a wild card
0    macro NAM$V_EXP_DIR = 52,6,1,0 %;           ! directory was explicit
0    macro NAM$V_EXP_DEV = 52,7,1,0 %;           ! device was explicit
0    macro NAM$V_WILDCARD = 52,8,1,0 %;          ! filename string included a wild card
0    macro NAM$V_SEARCH_LIST = 52,11,1,0 %;      ! search list present
0    macro NAM$V_CNCL_DEV = 52,12,1,0 %;         ! concealed device present
0    macro NAM$V_ROOT_DIR = 52,13,1,0 %;         ! root directory present
0    macro NAM$V_LOWVER = 52,14,1,0 %;           ! lower numbered version(s) of file exist(s)
0    macro NAM$V_HIGHVER = 52,15,1,0 %;          ! higher "
0    macro NAM$V_PPF = 52,16,1,0 %;              ! process-permanent file referenced indirectly
0    macro NAM$V_NODE = 52,17,1,0 %;             ! filename specification included a nodename
0    macro NAM$V_QUOTED = 52,18,1,0 %;           ! filename spec included a quoted string
0    macro NAM$V_GRP_MBR = 52,19,1,0 %;          ! directory spec was of group-member format
0    macro NAM$V_WILD_DIR = 52,20,1,0 %;         ! directory spec included a wild card
0    macro NAM$V_DIR_LVLS = 52,21,3,0 %;
0    literal NAM$S_DIR_LVLS = 3;                 ! number of directory levels (0=ufd only)
0    !
0    !   (inclusive or of other wild card bits)
0    macro NAM$R_FNB_BITS1 = 52,0,32,0 %;
0    macro NAM$V_WILD_UFD = 52,24,1,0 %;         ! ufd included a wild card
0    macro NAM$V_WILD_SFD1 = 52,25,1,0 %;        ! sfd1 included a wild card
0    macro NAM$V_WILD_SFD2 = 52,26,1,0 %;        ! sfd2 included a wild card
0    macro NAM$V_WILD_SFD3 = 52,27,1,0 %;        ! sfd3 included a wild card
0    macro NAM$V_WILD_SFD4 = 52,28,1,0 %;        ! sfd4 included a wild card
```

```
macro NAM$V_WILD_SFD5 = 52,29,1,0 %;       ! sfd5 included a wild card
macro NAM$V_WILD_SFD6 = 52,30,1,0 %;       ! sfd6 included a wild card
macro NAM$V_WILD_SFD7 = 52,31,1,0 %;       ! sfd7 included a wild card
macro NAM$R_FNB_BITS2 = 52,0,32,0 %;
macro NAM$V_WILD_GRP = 52,24,1,0 %;        ! group contained a wild card
macro NAM$V_WILD_MBR = 52,25,1,0 %;        ! member contained a wild card
! -----*****
!  (prior to 40 byte extension)
!
!  Extend the NAM block by 40 bytes.
!
macro NAM$B_NODE = 56,0,8,0 %;             ! Nodespec length
macro NAM$B_DEV = 57,0,8,0 %;              ! Device length
macro NAM$B_DIR = 58,0,8,0 %;              ! Directory length
macro NAM$B_NAME = 59,0,8,0 %;             ! Filename length
macro NAM$B_TYPE = 60,0,8,0 %;             ! Filetype length
macro NAM$B_VER = 61,0,8,0 %;              ! Version number length
macro NAM$L_NODE = 64,0,32,0 %;            ! Nodespec address
macro NAM$L_DEV = 68,0,32,0 %;             ! Device address
macro NAM$L_DIR = 72,0,32,0 %;             ! Directory address
macro NAM$L_NAME = 76,0,32,0 %;            ! Filename address
macro NAM$L_TYPE = 80,0,32,0 %;            ! Filetype address
macro NAM$L_VER = 84,0,32,0 %;             ! Version number address

!*** MODULE $XABDEF ***
!
!               definitions for all xabs
!                       $xabdef
!
!
!
!   the first four fields are shared in common between all xabs
!   and hence are defined only once
!   (the only exception is that the spare word may be used by some xabs)
!
literal XAB$S_XABDEF = 20;
macro XAB$B_COD = 0,0,8,0 %;               ! xab id code
macro XAB$B_BLN = 1,0,8,0 %;               ! block length
macro XAB$L_NXT = 4,0,32,0 %;              ! xab chain link
! WITH POSSIBLE EXCEPTION OF SPARE FIELD
macro XAB$W_RVN = 8,0,16,0 %;
macro XAB$R_RDT_OVERLAY = 12,0,0,0 %;
macro XAB$Q_RDT = 12,0,0,0 %;
literal XAB$S_RDT = 8;
macro XAB$R_RDT_FIELDS = 12,0,0,0 %;
macro XAB$L_RDT0 = 12,0,32,0 %;
macro XAB$L_RDT4 = 16,0,32,1 %;
! COMMON AMONG DAT AND RDT XABS
literal XAB$S_XABDEF1 = 23;
macro XAB$B_BKZ = 22,0,8,0 %;              ! COMMON TO FHC AND ALQ XABS
literal XAB$C_CXT_VER1 = 1;                !  RMS Context Extraction version 1

!*** MODULE $XABFHCDEF ***
! ++
!             file header characteristics xab definitions
!                     $xabfhcdef
!
```

```
        ! +++++++++++
        !   the fields of this xab cannot be rearranged since
        !   they correspond to an on-disk structure
        !
        literal XAB$C_FHC = 29;                  !  xabfhc id code
        literal XAB$K_FHCLEN = 44;               !  length of xabfhc
        literal XAB$C_FHCLEN = 44;               !  length of xabfhc
        literal XAB$S_XABFHCDEF = 44;
        ! THESE 4 FIELDS ARE COMMON TO ALL XABS AND
        ! HAVE BEEN DEFINED BY $XABDEF
        macro XAB$B_RFO = 8,0,8,0 %;             !  record format and file org
        macro XAB$B_ATR = 9,0,8,0 %;             !  record attributes
        macro XAB$W_LRL = 10,0,16,0 %;           !  longest record's length
        macro XAB$R_HBK_OVERLAY = 12,0,32,0 %;
        macro XAB$L_HBK = 12,0,32,0 %;           !  hi vbn allocated
        !  (n.b. reversed on disk!)
        macro XAB$R_HBK_FIELDS = 12,0,32,0 %;
        macro XAB$W_HBK0 = 12,0,16,0 %;
        macro XAB$W_HBK2 = 14,0,16,0 %;
        macro XAB$R_EBK_OVERLAY = 16,0,32,0 %;
        macro XAB$L_EBK = 16,0,32,0 %;           !  eof vbn
        !  (n.b. reversed on disk)
        macro XAB$R_EBK_FIELDS = 16,0,32,0 %;
        macro XAB$W_EBK0 = 16,0,16,0 %;
        macro XAB$W_EBK2 = 18,0,16,0 %;
        macro XAB$W_FFB = 20,0,16,0 %;           !  first free byte in eof block
        !  defined above in $xabdef, since it is shared
        !  by the all xab)
        macro XAB$B_HSZ = 23,0,8,0 %;            !  header size for vfc
        macro XAB$W_MRZ = 24,0,16,0 %;           !  max record size
        macro XAB$W_DXQ = 26,0,16,0 %;           !  default extend quantity
        macro XAB$W_GBC = 28,0,16,0 %;           !  global buffer count
        macro XAB$W_VERLIMIT = 38,0,16,0 %;      !  version limit for file.
        ! -----*****
        macro XAB$L_SBN = 40,0,32,0 %;           !  starting lbn if contiguous

        !*** MODULE $XABALLDEF ***
        ! --
        ! ++
        !
        !          allocation xab definitions
        !                  $xaballdef
        !
        !
        !
        ! ++++++++++
        !   the fields thru bkz cannot be rearranged due to
        !   their commonality with fab
        literal XAB$C_ALL = 20;                  !  xaball id code
        literal XAB$M_HRD = 1;
        literal XAB$M_ONC = 2;
        literal XAB$M_CBT = 32;
        literal XAB$M_CTG = 128;
        literal XAB$C_ANY = 0;                   !  any allocation o.k.
        literal XAB$C_CYL = 1;                   !  cylinder boundary
        literal XAB$C_LBN = 2;                   !  allocate at specified lbn
        literal XAB$C_VBN = 3;                   !  allocate near specified vbn
        literal XAB$C_RFI = 4;                   !  allocate near related file
```

```
literal XAB$K_ALLLEN = 32;                  ! length of xaball
literal XAB$C_ALLLEN = 32;                  ! length of xaball
literal XAB$S_XABALLDEF = 32;
! THESE 4 FIELDS ARE COMMON TO ALL XABS AND
! HAVE BEEN DEFINED BY $XABDEF
macro XAB$R_AOP_OVERLAY = 8,0,8,0 %;
macro XAB$B_AOP = 8,0,8,0 %;                ! allocation options
macro XAB$R_AOP_BITS = 8,0,8,0 %;
macro XAB$V_HRD = 8,0,1,0 %;                ! fail if requested alignment impossible
macro XAB$V_ONC = 8,1,1,0 %;                ! locate allocated space within a cylinder
macro XAB$V_CBT = 8,5,1,0 %;                ! contiguous alllocation, best try
macro XAB$V_CTG = 8,7,1,0 %;                ! contiguous allocation
macro XAB$B_ALN = 9,0,8,0 %;                ! alignment type
macro XAB$W_VOL = 10,0,16,0 %;              ! relative volume no. for allocation
!  (not applicable if aln = vbn or rfi)
macro XAB$L_LOC = 12,0,32,0 %;              ! allocation location
macro XAB$L_ALQ = 16,0,32,0 %;              ! allocation quantity
macro XAB$W_DEQ = 20,0,16,0 %;              ! default allocation quantity
!  defined above in $xabdef,since it is shared by the fhc
!  xab and has the same offset, of course)
! -----*****
macro XAB$B_AID = 23,0,8,0 %;               ! area id number
macro XAB$R_RFI_OVERLAY = 24,0,0,0 %;
macro XAB$W_RFI = 24,0,0,0 %;
literal XAB$S_RFI = 6;                      ! related file id
macro XAB$R_RFI_FIELDS = 24,0,0,0 %;
macro XAB$W_RFI0 = 24,0,16,0 %;             ! file number
macro XAB$W_RFI2 = 26,0,16,0 %;             ! seq number
macro XAB$W_RFI4 = 28,0,16,0 %;             ! rev number

!*** MODULE $XABDATDEF ***
! --
! ++
!
!         date/time xab definitions
!                 $xabdatdef
!
literal XAB$C_DAT = 18;                     ! xabdat id code
literal XAB$K_DATLEN_V2 = 36;               ! Version 2 XABDAT length
literal XAB$C_DATLEN_V2 = 36;               ! Version 2 XABDAT length
literal XAB$K_DATLEN = 44;                  ! length of XABDAT
literal XAB$C_DATLEN = 44;                  ! length of XABDAT
literal XAB$S_XABDATDEF = 44;
! THESE 4 FIELDS ARE COMMON TO ALL XABS AND
! HAVE BEEN DEFINED BY $XABDEF
macro XAB$R_CDT_OVERLAY = 20,0,0,0 %;
macro XAB$Q_CDT = 20,0,0,0 %;
literal XAB$S_CDT = 8;                      ! creation date & time
macro XAB$R_CDT_FIELDS = 20,0,0,0 %;
macro XAB$L_CDT0 = 20,0,32,0 %;
macro XAB$L_CDT4 = 24,0,32,1 %;
macro XAB$R_EDT_OVERLAY = 28,0,0,0 %;
macro XAB$Q_EDT = 28,0,0,0 %;
literal XAB$S_EDT = 8;                      ! expiration date & time
macro XAB$R_EDT_FIELDS = 28,0,0,0 %;
macro XAB$L_EDT0 = 28,0,32,0 %;
macro XAB$L_EDT4 = 32,0,32,1 %;
```

```
0       macro XAB$R_BDT_OVERLAY = 36,0,0,0 %;
0       macro XAB$Q_BDT = 36,0,0,0 %;
0       literal XAB$S_BDT = 8;                      !  backup date and time
0       macro XAB$R_BDT_FIELDS = 36,0,0,0 %;
0       macro XAB$L_BDT0 = 36,0,32,0 %;
0       macro XAB$L_BDT4 = 40,0,32,1 %;
0
0       !*** MODULE $XABRDTDEF ***
0       ! --
0       ! ++
0       !
0       !          revision date/time xab definitions
0       !                  $xabrdtdef
0       literal XAB$C_RDT = 30;                     !  xabrdt id code
0       literal XAB$K_RDTLEN = 20;                  !  length of rdt xab
0       literal XAB$C_RDTLEN = 20;                  !  length of rdt xab
0       literal XAB$S_XABRDTDEF = 20;
0       ! THESE 4 FIELDS ARE COMMON TO ALL XABS AND
0       ! HAVE BEEN DEFINED BY $XABDEF
0
0       !*** MODULE $XABPRODEF ***
0       ! --
0       ! ++
0       !
0       !          protection xab field definitions
0       !                  $xabprodef
0       !
0       !
0       literal XAB$C_PRO = 19;                     !  xabpro id code
0       literal XAB$M_NOREAD = 1;
0       literal XAB$M_NOWRITE = 2;
0       literal XAB$M_NOEXE = 4;
0       literal XAB$M_NODEL = 8;
0       literal XAB$S_XABPRODEF = 1;
0       macro XAB$R_XABPRODEF_BITS = 0,0,8,0 %;
0       macro XAB$V_NOREAD = 0,0,1,0 %;             !  deny read access
0       macro XAB$V_NOWRITE = 0,1,1,0 %;            !  deny write access
0       macro XAB$V_NOEXE = 0,2,1,0 %;              !  deny execution access
0       macro XAB$V_NODEL = 0,3,1,0 %;              !  deny delete access
0       literal XAB$M_PROPAGATE = 1;
0       literal XAB$K_PROLEN_V3 = 16;               !  V3a xabpro length
0       literal XAB$C_PROLEN_V3 = 16;               !  V3a xabpro length
0       literal XAB$K_PROLEN = 88;                  !  xabpro length
0       literal XAB$C_PROLEN = 88;                  !  xabpro length
0       literal XAB$S_XABPRODEF1 = 88;
0       ! THESE 4 FIELDS ARE COMMON TO ALL XABS AND
0       ! HAVE BEEN DEFINED BY $XABDEF
0       macro XAB$R_PRO_OVERLAY = 8,0,16,0 %;
0       macro XAB$W_PRO = 8,0,16,0 %;               !  protection mask
0       macro XAB$R_PRO_BITS = 8,0,16,0 %;
0       macro XAB$V_SYS = 8,0,4,0 %;
0       literal XAB$S_SYS = 4;                      !  system
0       macro XAB$V_OWN = 8,4,4,0 %;
0       literal XAB$S_OWN = 4;                      !  owner
0       macro XAB$V_GRP = 8,8,4,0 %;
0       literal XAB$S_GRP = 4;                      !  group
0       macro XAB$V_WLD = 8,12,4,0 %;
```

```
0       literal XAB$S_WLD = 4;                          !  world
0       macro XAB$B_MTACC = 10,0,8,0 %;                 !  Magtape access control char.
0       macro XAB$R_PROT_OPT_OVERLAY = 11,0,8,0 %;
0       macro XAB$B_PROT_OPT = 11,0,8,0 %;              !  XABPRO options field
0       macro XAB$R_PROT_OPT_FIELDS = 11,0,8,0 %;
0       macro XAB$V_PROPAGATE = 11,0,1,0 %;             !  Propagate security attributes on $ENTER and $RENAME
0       macro XAB$R_UIC_OVERLAY = 12,0,32,0 %;
0       macro XAB$L_UIC = 12,0,32,0 %;                  !  uic code
0       macro XAB$R_UIC_FIELDS = 12,0,32,0 %;
0       macro XAB$W_MBM = 12,0,16,0 %;                  !  member code
0       macro XAB$W_GRP = 14,0,16,0 %;                  !  group code
0       macro XAB$R_PROT_MODE_OVERLAY = 16,0,0,0 %;  !  RWED/mode protection for file
0       macro XAB$Q_PROT_MODE = 16,0,0,0 %;
0       literal XAB$S_PROT_MODE = 8;                    !  eventually may be a quadword
0       macro XAB$R_PROT_MODE_FIELDS = 16,0,8,0 %;
0       macro XAB$B_PROT_MODE = 16,0,8,0 %;             !  but currently only a byte
0       macro XAB$L_ACLBUF = 24,0,32,0 %;               !  address of user's ACL buffer
0       macro XAB$W_ACLSIZ = 28,0,16,0 %;               !  size of user's ACL buffer
0       macro XAB$W_ACLLEN = 30,0,16,0 %;               !  return length of entire ACL
0       macro XAB$L_ACLCTX = 32,0,32,0 %;               !  ACL context field
0       macro XAB$L_ACLSTS = 36,0,32,0 %;               !  ACL return err status
0
0       !*** MODULE $XABTRMDEF ***
0       !  --
0       !  ++
0       !
0       !           terminal control xab field definitions
0       !                   $xabtrmdef
0       !
0       !
0       !
0       literal XAB$C_TRM = 31;                         ! XABTRM ID CODE
0       literal XAB$K_TRMLEN = 36;                      !  length of xab of type terminal control
0       literal XAB$C_TRMLEN = 36;                      !  length of xab of type terminal control
0       literal XAB$S_XABTRMDEF = 36;
0       ! THESE 4 FIELDS ARE COMMON TO ALL XABS AND
0       ! HAVE BEEN DEFINED BY $XABDEF
0       macro XAB$L_ITMLST = 8,0,32,0 %;                !  item list address
0       macro XAB$W_ITMLST_LEN = 12,0,16,0 %;           !  item list length
0
0       !*** MODULE $XABSUMDEF ***
0       !  --
0       !  ++
0       !
0       !           summary xab field definitions
0       !                   $xabsumdef
0       !
0       !
0       literal XAB$C_SUM = 22;                         !  xabsum id code
0       literal XAB$K_SUMLEN = 12;                      !  xabsum length
0       literal XAB$C_SUMLEN = 12;                      !  xabsum length
0       literal XAB$S_XABSUMDEF = 12;
0       ! THESE 4 FIELDS ARE COMMON TO ALL XABS AND
0       ! HAVE BEEN DEFINED BY $XABDEF
0       macro XAB$B_NOA = 8,0,8,0 %;                    !  number of defined areas for index file
0       macro XAB$B_NOK = 9,0,8,0 %;                    !  number of defined keys for index file
0       macro XAB$W_PVN = 10,0,16,0 %;                  !  prologue version number (relative and index files)
```

```
0
0       !*** MODULE $XABKEYDEF ***
0       ! --
0       ! ++
0       !
0       !        key definition xab field definitions
0       !               $xabkeydef
0       !
0       !
0       literal XAB$C_KEY = 21;                   !  xabkey id code
0       literal XAB$M_DUP = 1;
0       literal XAB$M_CHG = 2;
0       literal XAB$M_NUL = 4;
0       literal XAB$M_IDX_NCMPR = 8;
0       literal XAB$M_KEY_NCMPR = 64;
0       literal XAB$M_DAT_NCMPR = 128;
0       literal XAB$C_STG = 0;                    !  string
0       literal XAB$C_IN2 = 1;                    !  signed 15 bit integer (2 bytes)
0       literal XAB$C_BN2 = 2;                    !  2 byte binary
0       literal XAB$C_IN4 = 3;                    !  signed 31 bit integer (4 bytes)
0       literal XAB$C_BN4 = 4;                    !  4 byte binary
0       literal XAB$C_PAC = 5;                    !  packed decimal (1-16 bytes)
0       literal XAB$C_IN8 = 6;                    !  signed 63 bit integer (4 bytes)
0       literal XAB$C_BN8 = 7;                    !  8 byte binary
0       literal XAB$C_MAXDTP = 7;                 !  max. legal data type
0       literal XAB$K_KEYLEN_V2 = 64;             !  old xabkey length
0       literal XAB$C_KEYLEN_V2 = 64;             !  old xabkey length
0       !
0       !  Additions for prologue 3 files
0       !
0       literal XAB$C_PRG3 = 3;                   !  Prologue version three
0       literal XAB$C_PRG2 = 2;                   !  Prologue version two
0       literal XAB$C_PRG1 = 1;                   !  Prologue versoin one
0       literal XAB$K_KEYLEN = 76;                !  xabkey length
0       literal XAB$C_KEYLEN = 76;                !  xabkey length
0       ! --
0       ! ++
0       literal XAB$S_XABKEYDEF = 76;
0       ! THESE 4 FIELDS ARE COMMON TO ALL XABS AND
0       ! HAVE BEEN DEFINED BY $XABDEF
0       !
0       !  the field layout of the key xab is such that it matchs as
0       !  closely as possible the layout of a key decriptor in the
0       !  index file prologue. this is so the contents may be moved
0       !  between the two structures as efficiently as possible.
0       !
0       macro XAB$B_IAN = 8,0,8,0 %;              !  index level area number
0       macro XAB$B_LAN = 9,0,8,0 %;              !  lowest index level area number
0       macro XAB$B_DAN = 10,0,8,0 %;             !  data level area number
0       macro XAB$B_LVL = 11,0,8,0 %;             !  level of root bucket
0       macro XAB$B_IBS = 12,0,8,0 %;             !  size of index buckets in virtual blocks
0       macro XAB$B_DBS = 13,0,8,0 %;             !  size of data buckets in virtual blocks
0       macro XAB$L_RVB = 14,0,32,0 %;            !  root bucket start vbn
0       macro XAB$R_FLG_OVERLAY = 18,0,8,0 %;
0       macro XAB$B_FLG = 18,0,8,0 %;             !  key option flags
0       macro XAB$R_FLG_BITS0 = 18,0,8,0 %;
0       macro XAB$V_DUP = 18,0,1,0 %;             !  duplicate key values allowed
```

```
0       macro XAB$V_CHG = 18,1,1,0 %;          ! alt key only --key field may change on update
0       macro XAB$V_NUL = 18,2,1,0 %;          ! alt key only --null key value enable
0       macro XAB$V_IDX_NCMPR = 18,3,1,0 %;    ! indicate index records for given key are not compressed
0       macro XAB$V_KEY_NCMPR = 18,6,1,0 %;    ! indicates key is not compressed in data record
0       macro XAB$R_FLG_BITS1 = 18,0,8,0 %;
0       macro XAB$V_DAT_NCMPR = 18,7,1,0 %;    ! data record is not compressed
0       macro XAB$B_DTP = 19,0,8,0 %;          ! key field data type
0       macro XAB$B_NSG = 20,0,8,0 %;          ! number of key segments
0       macro XAB$B_NUL = 21,0,8,0 %;          ! nul key character
0       macro XAB$B_TKS = 22,0,8,0 %;          ! total key field size (bytes)
0       macro XAB$B_REF = 23,0,8,0 %;          ! key of reference (0=prim key,
0       !  1-254 = alternate keys)
0       macro XAB$W_MRL = 24,0,16,0 %;         ! minimun record length to contain key field
0       macro XAB$W_IFL = 26,0,16,0 %;         ! index bucket fill size (bytes)
0       macro XAB$W_DFL = 28,0,16,0 %;         ! data bucket fil size (bytes)
0       macro XAB$R_POS_OVERLAY = 30,0,0,0 %;
0       macro XAB$W_POS = 30,0,0,0 %;
0       literal XAB$S_POS = 16;                ! key field record offset positions
0       macro XAB$R_POS_FIELDS = 30,0,0,0 %;
0       macro XAB$W_POS0 = 30,0,16,0 %;        ! segment 0
0       macro XAB$W_POS1 = 32,0,16,0 %;        ! segment 1
0       macro XAB$W_POS2 = 34,0,16,0 %;        ! segment 2
0       macro XAB$W_POS3 = 36,0,16,0 %;        ! segment 3
0       macro XAB$W_POS4 = 38,0,16,0 %;        ! segment 4
0       macro XAB$W_POS5 = 40,0,16,0 %;        ! segment 5
0       macro XAB$W_POS6 = 42,0,16,0 %;        ! segment 6
0       macro XAB$W_POS7 = 44,0,16,0 %;        ! segment 7
0       macro XAB$R_SIZ_OVERLAY = 46,0,0,0 %;
0       macro XAB$B_SIZ = 46,0,0,0 %;
0       literal XAB$S_SIZ = 8;                 ! key field segment sizes
0       macro XAB$R_SIZ_FIELDS = 46,0,0,0 %;
0       macro XAB$B_SIZ0 = 46,0,8,0 %;         ! segment 0
0       macro XAB$B_SIZ1 = 47,0,8,0 %;         ! segment 1
0       macro XAB$B_SIZ2 = 48,0,8,0 %;         ! segment 2
0       macro XAB$B_SIZ3 = 49,0,8,0 %;         ! segment 3
0       macro XAB$B_SIZ4 = 50,0,8,0 %;         ! segment 4
0       macro XAB$B_SIZ5 = 51,0,8,0 %;         ! segment 5
0       macro XAB$B_SIZ6 = 52,0,8,0 %;         ! segment 6
0       macro XAB$B_SIZ7 = 53,0,8,0 %;         ! segment 7
0       !
0       !   the positions of the above fields are dictated by the key descriptor
0       !   record layout in the index file prologue.
0       !
0       macro XAB$L_KNM = 56,0,32,0 %;         ! pointer to 32 character key name buffer
0       macro XAB$L_DVB = 60,0,32,0 %;         ! first data bucket start vbn
0       macro XAB$R_TYP_OVERLAY = 64,0,0,0 %;
0       macro XAB$B_TYP = 64,0,0,0 %;
0       literal XAB$S_TYP = 8;                 ! key field segment types
0       macro XAB$R_TYP_FIELDS = 64,0,0,0 %;
0       macro XAB$B_TYP0 = 64,0,8,0 %;         ! segment 0
0       macro XAB$B_TYP1 = 65,0,8,0 %;         ! segment 1
0       macro XAB$B_TYP2 = 66,0,8,0 %;         ! segment 2
0       macro XAB$B_TYP3 = 67,0,8,0 %;         ! segment 3
0       macro XAB$B_TYP4 = 68,0,8,0 %;         ! segment 4
0       macro XAB$B_TYP5 = 69,0,8,0 %;         ! segment 5
0       macro XAB$B_TYP6 = 70,0,8,0 %;         ! segment 6
0       macro XAB$B_TYP7 = 71,0,8,0 %;         ! segment 7
```

```
macro XAB$B_PROLOG = 72,0,8,0 %;          ! indicate prologue version desired (primary key only)

!*** MODULE $XABCXFDEF ***
!
!        RMS Context XAB associated with the FAB
!                $xabcxfdef
!
!
literal XAB$C_CXF = 32;                   ! XABCXF id code
literal XAB$M_CXFRST = 1;
literal XAB$K_CXFLEN = 60;                ! length of xab type CXF
literal XAB$C_CXFLEN = 60;                ! length of xab type CXF
literal XAB$S_XABCXFDEF = 60;
!  UP TILL NOW COMMON AMONG ALL XABS
!
!        Following in common with the CXR block, too.
!        Do not rearrange without changing both.
!
macro XAB$L_CXFSTS = 8,0,32,0 %;          ! Status of the last file operation.
macro XAB$L_CXFSTV = 12,0,32,0 %;         ! Status Value of the last file operation.
!
!  Top four bits of the options longword are reserved for the XABCXR.  These
!  bits describe the version of the key buffer.
!
macro XAB$R_CXFCOP_OVERLAY = 16,0,32,0 %;
macro XAB$L_CXFCOP = 16,0,32,0 %;         ! Context Options.
macro XAB$R_CXFCOP_BITS = 16,[illegible],0 %;
macro XAB$V_CXFRST = 16,0,1[illegible] %;         ! Restore file state - use context blk as input.
macro XAB$L_CXFBKP = 20,0,32,0 %;         ! Bookkeeping bits
macro XAB$W_CXFIFI = 24,0,16,0 %;         ! Internal File Identifier
macro XAB$B_CXFVER = 26,0,8,0 %;          ! prologue version num
!
!        Up Till now in common with XABCXR, too.
!
!        The following fields correspond to those in the FAB or IFB
!        They should not be rearranged as their order is assumed for
!        purposes of moving large chunks of data rather than a byte
!        or word at a time.  Note: ASSUME is used in the actual code
!
macro XAB$W_CXFDEQ = 32,0,16,0 %;         ! Default extention quantity
macro XAB$B_CXFFAC = 34,0,8,0 %;          ! File access
macro XAB$B_CXFSHR = 35,0,8,0 %;          ! File Sharing
macro XAB$W_CXFRTE = 36,0,16,0 %;         ! (Not used)
macro XAB$B_CXFORG = 39,0,8,0 %;          ! file organization
macro XAB$W_CXFGBC = 40,0,16,0 %;         ! global buffer count
macro XAB$B_CXFRTV = 42,0,8,0 %;          ! retrieval window

!*** MODULE $XABCXRDEF ***
!
!        RMS Context XAB associated with the RAB
!                $xabcxrdef
!
!
literal XAB$C_CXR = 33;                   ! XABCXR id code
literal XAB$M_CXRRST = 1;
literal XAB$C_CXB_VER1 = 1;
literal XAB$C_CXRBLEN = 512;              ! Length of CXRBUF (bytes)
```

```
literal XAB$K_CXRLEN = 84;              ! Length of XAB type CXR
literal XAB$C_CXRLEN = 84;              ! Length of XAB type CXR
literal XAB$S_XABCXRDEF = 84;
!   UP TILL NOW COMMON AMONG ALL XABS
!
!        Following in common with the CXF block, too.
!        Do not rearrange without changing it.
!
macro XAB$L_CXRSTS = 8,0,32,0 %;        ! Status of the last record operation.
macro XAB$L_CXRSTV = 12,0,32,0 %;       ! Status Value of the last record operation.
macro XAB$R_CXRCOP_OVERLAY = 16,0,32,0 %;
macro XAB$L_CXRCOP = 16,0,32,0 %;       ! Context Options.
macro XAB$R_CXRCOP_BITS = 16,0,32,0 %;
macro XAB$V_CXRRST = 16,0,1,0 %;        ! Restore file/record state - use context blk as input.
macro XAB$V_CXRBVER = 16,28,4,0 %;
literal XAB$S_CXRBVER = 4;              ! Version of Key buffer
macro XAB$L_CXRBKP = 20,0,32,0 %;       ! Bookkeeping bits
macro XAB$W_CXRISI = 24,0,16,0 %;       ! Internal Record Identifier
macro XAB$B_CXRVER = 26,0,8,0 %;        ! prologue version num.
!
!        Up Till now in common with XABCXF, too.
!
!
!        The following elements are arranged such that large amounts of
!        data can be moved at a time rather than words or bytes.  Do not
!        rearrange them without this consideration in mind.
!
!        The following elements are stream dependent regardless of file org.
!
macro XAB$B_CXRMBF = 32,0,8,0 %;        ! Multibuffer count
macro XAB$B_CXRMBC = 33,0,8,0 %;        ! Multiblock count
macro XAB$W_CXRBFZ = 34,0,16,0 %;       ! sz in byte of CXRBUF
!
!     The following elements are necessary for saving the NRP context for
!     Sequential and Relative files.
!
macro XAB$L_CXRVBN = 36,0,32,0 %;       ! NRP VBN
macro XAB$W_CXROFF = 40,0,16,0 %;       ! NRP offset in VBN
macro XAB$W_FILL_8 = 42,0,16,0 %;       ! mbz - longword align
!
!     The following elements are necessary for saving the NRP context for
!     ISAM files.
!
macro XAB$L_CXRPOS0 = 44,0,32,0 %;      ! Primary Positioning RFA
macro XAB$W_CXRPOS4 = 48,0,16,0 %;
macro XAB$L_CXRCUR0 = 52,0,32,0 %;      ! Current Positioning RFA
macro XAB$W_CXRCUR4 = 56,0,16,0 %;
macro XAB$L_CXRSID0 = 60,0,32,0 %;      ! SIDR positioning RFA
macro XAB$W_CXRSID4 = 64,0,16,0 %;
macro XAB$W_CXRCNT = 68,0,16,0 %;       ! SIDR array count
macro XAB$B_CXRKREF = 70,0,8,0 %;       ! Cur Key of Reference
macro XAB$B_CXRKLEN = 71,0,8,0 %;       ! Length of key
macro XAB$L_CXRBUF = 72,0,32,0 %;       ! address of key buf

!*** MODULE $XABJNLDEF ***
! ++
!
```

```
0	!	Journal XAB definitions
0	!		$xabjnldef
0	!
0	!
0	literal XAB$C_JNL = 34;			!  xabjnl id code
0	literal XAB$M_ONLY_RU = 1;
0	literal XAB$M_RU = 2;
0	literal XAB$M_BI = 4;
0	literal XAB$M_AI = 8;
0	literal XAB$M_AT = 16;
0	literal XAB$M_NEVER_RU = 32;
0	literal XAB$K_MAXJNLNAM = 16;		!  max size of ascii string journal name
0	literal XAB$C_MAXJNLNAM = 16;		!  max size of ascii string journal name
0	literal XAB$K_JNLLEN = 60;
0	literal XAB$C_JNLLEN = 60;
0	literal XAB$S_XABJNLDEF = 60;
0	! THESE 4 FIELDS ARE COMMON TO ALL XABS AND
0	! HAVE BEEN DEFINED BY $XABDEF
0	macro XAB$R_JOP_OVERLAY = 8,0,16,0 %;
0	macro XAB$W_JOP = 8,0,16,0 %;		!  journaling flags
0	macro XAB$R_JOP_BITS = 8,0,8,0 %;
0	macro XAB$V_ONLY_RU = 8,0,1,0 %;	!  Recovery-unit only access
0	macro XAB$V_RU = 8,1,1,0 %;		!  Recovery unit
0	macro XAB$V_BI = 8,2,1,0 %;		!  Before Image
0	macro XAB$V_AI = 8,3,1,0 %;		!  After Image
0	macro XAB$V_AT = 8,4,1,0 %;		!  Audit Trail
0	macro XAB$V_NEVER_RU = 8,5,1,0 %;	!  Never journal in Recovery-unit
0	macro XAB$B_BIS = 12,0,8,0 %;		!  BI journal name buffer size
0	macro XAB$B_BIL = 13,0,8,0 %;		!  BI journal name return size
0	macro XAB$L_BIA = 16,0,32,0 %;		!  BI journal name buffer address
0	macro XAB$B_AIS = 20,0,8,0 %;		!  AI journal name buffer size
0	macro XAB$B_AIL = 21,0,8,0 %;		!  AI journal name return size
0	macro XAB$L_AIA = 24,0,32,0 %;		!  AI journal name buffer address
0	macro XAB$B_ATS = 28,0,8,0 %;		!  AT journal name buffer size
0	macro XAB$B_ATL = 29,0,8,0 %;		!  AT journal name return size
0	macro XAB$L_ATA = 32,0,32,0 %;		!  AT journal name buffer address
0
0	!*** MODULE $FSCNDEF ***
0	! ++
0	!
0	!	Descriptor codes for SYS$FILESCAN
0	!
0	!
0	literal FSCN$M_NODE = 1;
0	literal FSCN$M_DEVICE = 2;
0	literal FSCN$M_ROOT = 4;
0	literal FSCN$M_DIRECTORY = 8;
0	literal FSCN$M_NAME = 16;
0	literal FSCN$M_TYPE = 32;
0	literal FSCN$M_VERSION = 64;
0	literal FSCN$S_FLDFLAGS = 1;
0	macro FSCN$V_NODE = 0,0,1,0 %;		!  Node name present
0	macro FSCN$V_DEVICE = 0,1,1,0 %;	!  Device name present
0	macro FSCN$V_ROOT = 0,2,1,0 %;		!  Root directory present
0	macro FSCN$V_DIRECTORY = 0,3,1,0 %;	!  Directory present
0	macro FSCN$V_NAME = 0,4,1,0 %;		!  File name present
0	macro FSCN$V_TYPE = 0,5,1,0 %;		!  File type present
```

```
macro FSCN$V_VERSION = 0,6,1,0 %;           ! File version present
literal FSCN$_FILESPEC = 1;                 ! complete filespec
literal FSCN$_NODE = 2;                     ! node:: field
literal FSCN$_DEVICE = 3;                   ! device: field
literal FSCN$_ROOT = 4;                     ! [root.] field
literal FSCN$_DIRECTORY = 5;                ! [directory] field
literal FSCN$_NAME = 6;                     ! name field
literal FSCN$_TYPE = 7;                     ! .typ field
literal FSCN$_VERSION = 8;                  ! ;version field
literal FSCN$S_ITEM_LEN = 8;
literal FSCN$S_FSCNDEF = 8;
macro FSCN$W_LENGTH = 0,0,16,0 %;           ! return length word
macro FSCN$W_ITEM_CODE = 2,0,16,0 %;        ! item code value
macro FSCN$L_ADDR = 4,0,32,0 %;             ! return length pointer

!*** MODULE $RMEDEF ***
!
!                   rms escape definitions
!
!    the following values identify various requests for non-standard rms
!    functions.  they are currently input to the $modify function in the
!    ctx field of the fab only if the esc bit is set in fop.  incorrect
!    use of these capabilties could cause rms to fail, hence great caution
!    should be exercised in their use.
!
literal RME$C_SETRFM = 1;                   ! change rfm, mrs, and fsz (if vfc) in ifab only
literal RME$C_PPFECHO = 2;                  ! enable echo of SYS$INPUT to SYS$OUTPUT
!
! ****************************************************************************
! *                                                                          *
! *  Copyright (c) 1982, 1983                                                *
! *  by DIGITAL Equipment Corporation, Maynard, Mass.                        *
! *                                                                          *
! *  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.                 *
! *                                                                          *
! ****************************************************************************
! *********************************************************************************************************
!  Created 15-SEP-1984 22:55:15  by  VAX-11 SDL V2.0       Source: 15-SEP-1984 22:50:36 _$255$DUA28:[RMS.SRC]RMSFILS
! *********************************************************************************************************


!*** MODULE $PLGDEF ***
literal PLG$C_VER_NO = 1;                   ! current prolog version number
literal PLG$C_VER_IDX = 2;                  ! new plg for indexed files
literal PLG$C_VER_3 = 3;                    ! new plg for compression, space reclamation (plg 3)
```

```
;   0    literal PLG$K_BLN = 120;
;   0    literal PLG$C_BLN = 120;
;   0    literal PLG$S_PLGDEF = 120;
;   0    macro PLG$B_DBKTSIZ = 11,0,8,0 %;          !  data bucket size
;   0    macro PLG$R_FLAGS_OVERLAY = 16,0,8,0 %;
;   0    macro PLG$B_FLAGS = 16,0,8,0 %;            !  flag bits
;   0    macro PLG$R_FLAGS_BITS = 16,0,8,0 %;
;   0    macro PLG$V_NOEXTEND = 16,0,1,0 %;         !  no extend allowed (rel)
;   0    macro PLG$B_AVBN = 102,0,8,0 %;            !  vbn of first area descriptor
;   0    macro PLG$B_AMAX = 103,0,8,0 %;            !  maximum number of areas
;   0    macro PLG$W_DVBN = 104,0,16,0 %;           !  first data bucket vbn
;   0    macro PLG$L_MRN = 108,0,32,0 %;            !  maximum record number (rel)
;   0    macro PLG$L_EOF = 112,0,32,0 %;            !  eof vbn (rel)
;   0    macro PLG$W_VER_NO = 116,0,16,0 %;         !  version number
;   0    macro PLG$W_GBC = 118,0,16,0 %;            !  default global buffer count
;   0
;   0    !*** MODULE $DLCDEF ***
;   0    !
;   0    !
;   0    !    relative file deletion control byte bit definitions
;   0    !
;   0    literal DLC$M_DELETED = 4;
;   0    literal DLC$M_REC = 8;
;   0    literal DLC$S_DLCDEF = 1;
;   0    macro DLC$R_DLCDEF_BITS = 0,0,8,0 %;
;   0    macro DLC$V_DELETED = 0,2,1,0 %;           !  record deleted
;   0    macro DLC$V_REC = 0,3,1,0 %;               !  record exists (but may have been deleted)
;   0
;   0    !*** MODULE $BKTDEF ***
;   0    !
;   0    !   index bucket definition
;   0    !
;   0    !   this is the bucket format for RMS-11/RMS-32 index files.
;   0    !
;   0    literal BKT$K_OVERHDSZ = 14;               !  length of bucket overhead
;   0    literal BKT$C_OVERHDSZ = 14;               !  length of bucket overhead
;   0    literal BKT$M_LASTBKT = 1;
;   0    literal BKT$M_ROOTBKT = 2;
;   0    literal BKT$M_PTR_SZ = 24;
;   0    literal BKT$C_ENDOVHD = 4;                 !  end of bucket overhead
;   0    literal BKT$C_DATBKTOVH = 2;               !  end of bucket overhead for data buckets
;   0    literal BKT$C_DUPBKTOVH = 4;               !  additional end of data bucket overhead
;   0    !    when duplicates are allowed (LCB pointer
;   0    literal BKT$C_MAXBKTSIZ = 63;              !  maximum bucket size
;   0    literal BKT$S_BKTDEF = 14;
;   0    macro BKT$B_CHECKCHAR = 0,0,8,0 %;         !  bucket check character
;   0    macro BKT$R_AREANO_OVERLAY = 1,0,8,0 %;
;   0    macro BKT$B_AREANO = 1,0,8,0 %;            !  area number form which bucket was allocated
;   0    macro BKT$B_INDEXNO = 1,0,8,0 %;           !  index to which this bucket belongs (plg 3)
;   0    macro BKT$W_ADRSAMPLE = 2,0,16,0 %;        !  address sample - low 16 bits of first vbn in bucket
;   0    macro BKT$R_FREESPACE_OVERLAY = 4,0,16,0 %;
;   0    macro BKT$W_FREESPACE = 4,0,16,0 %;        !  displacement in bucket of first free byte
;   0    macro BKT$W_KEYFRESPC = 4,0,16,0 %;        !  pointer to key's free space (plg 3)
;   0    macro BKT$R_NXTRECID_OVERLAY = 6,0,16,0 %;
;   0    macro BKT$W_NXTRECID = 6,0,16,0 %;         !  next available word record id (plg 3)
;   0    macro BKT$R_NXTRECID_FIELDS = 6,0,16,0 %;
;   0    macro BKT$B_NXTRECID = 6,0,8,0 %;          !  next available record id
```

```
macro BKT$B_LSTRECID = 7,0,8,0 %;         ! last id in range
macro BKT$L_NXTBKT = 8,0,32,0 %;          ! vbn of next bucket
macro BKT$B_LEVEL = 12,0,8,0 %;           ! bucket level number
macro BKT$R_BKTCB_OVERLAY = 13,0,8,0 %;
macro BKT$B_BKTCB = 13,0,8,0 %;           ! bucket control bits
macro BKT$R_BKTCB_BITS = 13,0,8,0 %;
macro BKT$V_LASTBKT = 13,0,1,0 %;         ! last bucket in horizontal chain
macro BKT$V_ROOTBKT = 13,1,1,0 %;         ! root bucket
macro BKT$V_PTR_SZ = 13,3,2,0 %;
literal BKT$S_PTR_SZ = 2;                 ! size of vbn pointers in this bucket

!*** MODULE $IRCDEF ***
!
!  index record definition
!
!  this is the definition of RMS-11/RMS-32 index file record formats
!
literal IRC$M_PTRSZ = 3;
literal IRC$M_RECORDCB = 252;
literal IRC$M_DELETED = 4;
literal IRC$M_NOPTRSZ = 16;
literal IRC$M_FIRST_KEY = 128;
literal IRC$M_RRV = 8;
literal IRC$M_NODUPCNT = 16;
literal IRC$M_RU_DELETE = 32;
literal IRC$M_RU_UPDATE = 64;
literal IRC$C_IDXPTRBAS = 2;              ! used to determine size of pointer in index
literal IRC$C_IDXOVHDSZ = 1;              ! includes record control byte
!
!  data bucket record
!
literal IRC$S_IRCDEF = 1;
macro IRC$B_CONTROL = 0,0,8,0 %;          ! record control byte
macro IRC$R_CONTROL_BITS0 = 0,0,8,0 %;
macro IRC$V_PTRSZ = 0,0,2,0 %;
literal IRC$S_PTRSZ = 2;                  ! size of pointer
macro IRC$V_RECORDCB = 0,2,6,0 %;
literal IRC$S_RECORDCB = 6;               ! record control bits
!
!  record control bits used only in primary data record and SIDR array element
!  control bytes
!
macro IRC$R_CONTROL_BITS1 = 0,0,8,0 %;
macro IRC$V_DELETED = 0,2,1,0 %;          ! record is deleted
macro IRC$V_NOPTRSZ = 0,4,1,0 %;          ! no RRV
macro IRC$V_FIRST_KEY = 0,7,1,0 %;
!
!  record control bits used only in primary data record control bytes
!
macro IRC$R_CONTROL_BITS2 = 0,0,8,0 %;
macro IRC$V_RRV = 0,3,1,0 %;              ! rrv record
!
!  record control bits used only in prologue 2 SIDR record control bytes
!
macro IRC$R_CONTROL_BITS3 = 0,0,8,0 %;
macro IRC$V_NODUPCNT = 0,4,1,0 %;         ! DUP_CNT field absent
!
```

```
!    record control bits used only in prologue 3 RRV, UDR and SIDR record control
!    bytes of RU journalled files. (RU_UPDATE is set only in UDR record control
!    bytes)
!
macro IRC$R_CONTROL_BITS4 = 0,0,8,0 %;
macro IRC$V_RU_DELETE = 0,5,1,0 %;          !  record is RU deleted
macro IRC$V_RU_UPDATE = 0,6,1,0 %;          !  record is RU updated
!
!    record control bits reserved for RMS-11 use only (these may not be re-defined
!    except for prologue 3 records)
!
!    Bit number 5
!    Bit number 6
!
!
!    index bucket record
!
macro IRC$R_CONTROL_FIELDS4 = 0,0,8,0 %;
macro IRC$T_BUCKETPTR = 1,0,0,0 %;          !  bucket pointer (not referenced in the code,
!    just present for consistency)
literal IRC$S_IRCDEF1 = 3;
macro IRC$B_ID = 1,0,8,0 %;                 !  record id
macro IRC$B_RRV_ID = 2,0,8,0 %;             !  rrv's id -- always in the same place
!
!    prologue 3 data bucket record
!
literal IRC$C_DATSZFLD = 2;                 !  size of size field in variable length records
literal IRC$C_DATPTRBAS = 3;                !  used to determine size of RRV in data buckets
literal IRC$C_DCNTSZFLD = 4;                !  size of duplicate count field in Plg 2 SIDRs
literal IRC$C_DATOVHDSZ = 2;                !  includes the record control byte, and the id
literal IRC$C_FIXOVHDSZ = 7;                !  the record overhead for fixed record
literal IRC$C_VAROVHDSZ = 9;                !  record overhead for variable records
literal IRC$C_RRVOVHDSZ = 7;                !  size of RRV
!
!    prologue 3 constants
!
literal IRC$C_DATPTRBS3 = 4;                !  used to determine size of RRV in data buckets
literal IRC$C_DATOVHSZ3 = 3;                !  record control byte, and id
literal IRC$C_FIXOVHSZ3 = 9;                !  record overhead for fixed length records
literal IRC$C_VAROVHSZ3 = 11;               !  record overhead for variable length records
literal IRC$C_RRVOVHSZ3 = 9;                !  size of RRV
literal IRC$C_SDROVHSZ3 = 2;                !  record overhead for SIDRs
literal IRC$C_KEYCMPOVH = 2;                !  key compression overhead
literal IRC$C_DATCMPOVH = 3;                !  data compression overhead
literal IRC$S_IRCDEF2 = 5;
macro IRC$W_ID = 1,0,16,0 %;                !  record id
macro IRC$W_RRV_ID = 3,0,16,0 %;            !  rrv's id -- always in the same place
!
!    constants
!

!*** MODULE $KEYDEF ***
!
!    definitions for the key descriptors in the prologue
!
!    these definitions are associated w/ the plg and area definitions
!
```

```
;   0       literal KEY$M_DUPKEYS = 1;
;   0       literal KEY$M_CHGKEYS = 2;
;   0       literal KEY$M_NULKEYS = 4;
;   0       literal KEY$M_IDX_COMPR = 8;
;   0       literal KEY$M_INITIDX = 16;
;   0       literal KEY$M_KEY_COMPR = 64;
;   0       literal KEY$M_REC_COMPR = 128;
;   0       literal KEY$C_MAX_DAT = 10;                 !  (PLG3) Maximum size of a non-compressed data
;   0       !    record
;   0       literal KEY$C_MAX_PRIMARY = 6;              !  (PLG3) Maximum size of a non-compressed
;   0       !    primary key
;   0       literal KEY$C_MAX_INDEX = 6;                !  (PLG3) Maximum size of a non-compressed
;   0       !    index and SIDR key
;   0       literal KEY$C_STRING = 0;                   !  string data type
;   0       literal KEY$C_SGNWORD = 1;                  !  signed binary word
;   0       literal KEY$C_UNSGNWORD = 2;                !  unsigned binary word
;   0       literal KEY$C_SGNLONG = 3;                  !  signed binary long word
;   0       literal KEY$C_UNSGNLONG = 4;                !  unsigned binary long word
;   0       literal KEY$C_PACKED = 5;                   !  packed decimal
;   0       literal KEY$C_SGNQUAD = 6;                  !  signed binary quadword
;   0       literal KEY$C_UNSGNQUAD = 7;                !  unsigned binary quadword
;   0       literal KEY$C_MAX_DATA = 7;                 !  maximum data type value allowed
;   0       literal KEY$K_BLN = 96;                     !  length of key descriptor in the prologue (plg 3)
;   0       literal KEY$C_BLN = 96;                     !  length of key descriptor in the prologue (plg 3)
;   0       literal KEY$C_SPARE = 6;                    !  these are spare words in key block (plg 3)
;   0       literal KEY$S_KEYDEF = 96;
;   0       macro KEY$L_IDXFL = 0,0,32,0 %;             !  vbn for next key descriptor
;   0       macro KEY$W_NOFF = 4,0,16,0 %;              !  offset to next key descriptor
;   0       macro KEY$B_IANUM = 6,0,8,0 %;              !  index area number
;   0       macro KEY$B_LANUM = 7,0,8,0 %;              !  level 1 area number
;   0       macro KEY$B_DANUM = 8,0,8,0 %;              !  data area number
;   0       macro KEY$B_ROOTLEV = 9,0,8,0 %;            !  root level
;   0       macro KEY$B_IDXBKTSZ = 10,0,8,0 %;          !  index bucket size
;   0       macro KEY$B_DATBKTSZ = 11,0,8,0 %;          !  data bucket size
;   0       macro KEY$L_ROOTVBN = 12,0,32,0 %;          !  root bucket pointer
;   0       macro KEY$R_FLAGS_OVERLAY = 16,0,8,0 %;
;   0       macro KEY$B_FLAGS = 16,0,8,0 %;             !  flag bits
;   0       macro KEY$R_FLAGS_BITS0 = 16,0,8,0 %;
;   0       macro KEY$V_DUPKEYS = 16,0,1,0 %;           !  duplicate key values allowed
;   0       macro KEY$V_CHGKEYS = 16,1,1,0 %;           !  key value may change on $update operation
;   0       macro KEY$V_NULKEYS = 16,2,1,0 %;           !  null key character enabled
;   0       macro KEY$V_IDX_COMPR = 16,3,1,0 %;         !  index is compressed
;   0       macro KEY$V_INITIDX = 16,4,1,0 %;           !  index must be initialized
;   0       macro KEY$V_KEY_COMPR = 16,6,1,0 %;         !  (PLG3) key is compressed in data record
;   0       macro KEY$R_FLAGS_BITS1 = 16,0,8,0 %;
;   0       macro KEY$V_REC_COMPR = 16,7,1,0 %;         !  (PLG3) Data record is compressed
;   0       macro KEY$B_DATATYPE = 17,0,8,0 %;          !  data type for key
;   0       macro KEY$B_SEGMENTS = 18,0,8,0 %;          !  number of segments in key
;   0       macro KEY$B_NULLCHAR = 19,0,8,0 %;          !  "null" character
;   0       macro KEY$B_KEYSZ = 20,0,8,0 %;             !  total key size
;   0       macro KEY$B_KEYREF = 21,0,8,0 %;            !  key of reference
;   0       macro KEY$W_MINRECSZ = 22,0,16,0 %;         !  minimum record length
;   0       macro KEY$W_IDXFILL = 24,0,16,0 %;          !  index fill quantity
;   0       macro KEY$W_DATFILL = 26,0,16,0 %;          !  data fill quantity
;   0       macro KEY$R_POSITION_OVERLAY = 28,0,16,0 %;
;   0       macro KEY$W_POSITION = 28,0,16,0 %;         !  key seg position
;   0       macro KEY$W_POSITION0 = 28,0,16,0 %;        !  another name for position 0
```

```
;  0   macro KEY$W_POSITION1 = 30,0,16,0 %;          !  position 1
;  0   macro KEY$W_POSITION2 = 32,0,16,0 %;          !  position 2
;  0   macro KEY$W_POSITION3 = 34,0,16,0 %;          !  position 3
;  0   macro KEY$W_POSITION4 = 36,0,16,0 %;          !  position 4
;  0   macro KEY$W_POSITION5 = 38,0,16,0 %;
;  0   macro KEY$W_POSITION6 = 40,0,16,0 %;
;  0   macro KEY$W_POSITION7 = 42,0,16,0 %;
;  0   macro KEY$R_SIZE_OVERLAY = 44,0,8,0 %;
;  0   macro KEY$B_SIZE = 44,0,8,0 %;                !  key segment size
;  0   macro KEY$B_SIZE0 = 44,0,8,0 %;               !  another name for size
;  0   macro KEY$B_SIZE1 = 45,0,8,0 %;               !  size 1
;  0   macro KEY$B_SIZE2 = 46,0,8,0 %;
;  0   macro KEY$B_SIZE3 = 47,0,8,0 %;
;  0   macro KEY$B_SIZE4 = 48,0,8,0 %;
;  0   macro KEY$B_SIZE5 = 49,0,8,0 %;
;  0   macro KEY$B_SIZE6 = 50,0,8,0 %;
;  0   macro KEY$B_SIZE7 = 51,0,8,0 %;
;  0   macro KEY$T_KEYNAM = 52,0,0,0 %;
;  0   literal KEY$S_KEYNAM = 32;                    !  key name
;  0   macro KEY$L_LDVBN = 84,0,32,0 %;              !  first data bucket
;  0   macro KEY$R_TYPE_OVERLAY = 88,0,8,0 %;
;  0   macro KEY$B_TYPE = 88,0,8,0 %;                !  key segment datatype (plg 3)
;  0   macro KEY$B_TYPE0 = 88,0,8,0 %;               !  another name for first datatype (plg 3)
;  0   macro KEY$B_TYPE1 = 89,0,8,0 %;               !  (plg 3)
;  0   macro KEY$B_TYPE2 = 90,0,8,0 %;               !  (plg 3)
;  0   macro KEY$B_TYPE3 = 91,0,8,0 %;               !  (plg 3)
;  0   macro KEY$B_TYPE4 = 92,0,8,0 %;               !  (plg 3)
;  0   macro KEY$B_TYPE5 = 93,0,8,0 %;               !  (plg 3)
;  0   macro KEY$B_TYPE6 = 94,0,8,0 %;               !  (plg 3)
;  0   macro KEY$B_TYPE7 = 95,0,8,0 %;               !  (plg 3)
;  0
;  0   !*** MODULE $AREADEF ***
;  0   !
;  0   !
;  0   !   definitions for the area descriptor in the prologue
;  0   !
;  0   !
;  0   literal AREA$C_CYL = 1;                       !  cylinded alignment
;  0   literal AREA$C_LBN = 2;                       !  logical block alignment
;  0   literal AREA$C_VBN = 3;                       !  virtual block alignment
;  0   literal AREA$C_RFI = 4;                       !  allocate close to related file by fid
;  0   literal AREA$M_HARD = 1;
;  0   literal AREA$M_ONC = 2;
;  0   literal AREA$M_CBT = 32;
;  0   literal AREA$M_CTG = 128;
;  0   literal AREA$K_BLN = 64;                      !  length of area descriptor in the prologue
;  0   literal AREA$C_BLN = 64;                      !  length of area descriptor in the prologue
;  0   literal AREA$S_AREADEF = 64;
;  0   macro AREA$B_FLAGS = 1,0,8,0 %;               !  not currently used
;  0   macro AREA$B_AREAID = 2,0,8,0 %;              !  area id
;  0   macro AREA$B_ARBKTSZ = 3,0,8,0 %;             !  bucket size for area
;  0   macro AREA$W_VOLUME = 4,0,16,0 %;             !  relative volume number
;  0   macro AREA$B_ALN = 6,0,8,0 %;                 !  extend allocation alignment
;  0   macro AREA$R_AOP_OVERLAY = 7,0,8,0 %;
;  0   macro AREA$B_AOP = 7,0,8,0 %;                 !  alignment options
;  0   macro AREA$R_AOP_BITS = 7,0,8,0 %;
;  0   macro AREA$V_HARD = 7,0,1,0 %;                !  absolute alignment or nothing
```

```
;  0   macro AREA$V_ONC = 7,1,1,0 %;           !  locate on cylinder
;  0   macro AREA$V_CBT = 7,5,1,0 %;           !  contiguous best try
;  0   macro AREA$V_CTG = 7,7,1,0 %;           !  contiguous
;  0   macro AREA$L_AVAIL = 8,0,32,0 %;        !  available (returned) buckets
;  0   macro AREA$L_CVBN = 12,0,32,0 %;        !  start vbn for current extent
;  0   macro AREA$L_CNBLK = 16,0,32,0 %;       !  number of blocks in current extent
;  0   macro AREA$L_USED = 20,0,32,0 %;        !  number of blocks used
;  0   macro AREA$L_NXTVBN = 24,0,32,0 %;      !  next vbn to use
;  0   macro AREA$L_NXT = 28,0,32,0 %;         !  start vbn for next extent
;  0   macro AREA$L_NXBLK = 32,0,32,0 %;       !  number of blocks in next extent
;  0   macro AREA$W_DEQ = 36,0,16,0 %;         !  default extend quantity
;  0   macro AREA$L_LOC = 40,0,32,0 %;         !  start lbn on volume
;  0   macro AREA$W_RFI = 44,0,0,0 %;
;  0   literal AREA$S_RFI = 6;                 !  related file id
;  0   macro AREA$L_TOTAL_ALLOC = 50,0,32,0 %; !  total block allocation
;  0   macro AREA$W_CHECK = 62,0,16,0 %;       !  checksum
;  0
;  0   !*** MODULE $RJRDEF ***
;  0   !
;  0   !
;  0   !  definitions for the journaling records in RMS journals
;  0   !       for performance reasons, the BKT and BLK forms
;  0   !       should be an integral number of quadwords.
;  0   !
;  0   !
;  0   literal RJR$C_VER1 = 1;                 !  journal version 1
;  0   literal RJR$C_VER2 = 2;                 !  journal version 2
;  0   literal RJR$C_MAXVER = 2;               !  version limit
;  0   literal RJR$C_NOENT = 0;                !  null type
;  0   literal RJR$C_MAPPING = 1;              !  mapping entry
;  0   literal RJR$C_FILENAME = 1;             !  mapping entry synonym
;  0   literal RJR$C_RECORD = 2;               !  record entry
;  0   literal RJR$C_BLOCK = 3;                !  block IO entry (at, etc...)
;  0   literal RJR$C_BUCKET = 4;               !  ISAM bucket
;  0   literal RJR$C_EXTEND = 5;               !  extend (AT, AI)
;  0   literal RJR$C_AT_RECORD = 6;            !  audit trail record
;  0   literal RJR$C_MAXTYP = 6;               !  entry-type limit
;  0   literal RJR$C_SEQ = 0;                  !  sequential file org
;  0   literal RJR$C_REL = 1;                  !  relative file org
;  0   literal RJR$C_IDX = 2;                  !  indexed file org
;  0   literal RJR$C_HSH = 3;                  !  hashed file org
;  0   literal RJR$C_MAXORG = 3;               !  org limit
;  0   literal RJR$_NOOP = 0;                  !  null operation
;  0   literal RJR$_BUCKET = 1;                !  bucket-level I/O
;  0   literal RJR$_CLOSE = 2;                 !  close
;  0   literal RJR$_CONNECT = 3;               !  connect
;  0   literal RJR$_CREATE = 4;                !  create
;  0   literal RJR$_DELETE = 5;                !  delete
;  0   literal RJR$_DISCONNECT = 6;            !  disconnect
;  0   literal RJR$_DISPLAY = 7;               !  display
;  0   literal RJR$_ENTER = 8;                 !  enter
;  0   literal RJR$_ERASE = 9;                 !  erase
;  0   literal RJR$_EXTEND = 10;               !  extend
;  0   literal RJR$_FIND = 11;                 !  find
;  0   literal RJR$_FLUSH = 12;                !  flush
;  0   literal RJR$_FREE = 13;                 !  free
;  0   literal RJR$_GET = 14;                  !  get
```

```
;  0     literal RJR$_MODIFY = 15;                    !  modify
;  0     literal RJR$_NXTVOL = 16;                    !  next volume
;  0     literal RJR$_OPEN = 17;                      !  open
;  0     literal RJR$_PARSE = 18;                     !  parse
;  0     literal RJR$_PUT = 19;                       !  put
;  0     literal RJR$_READ = 20;                      !  block I/O read
;  0     literal RJR$_RELEASE = 21;                   !  release
;  0     literal RJR$_REMOVE = 22;                    !  remove
;  0     literal RJR$_RENAME = 23;                    !  rename
;  0     literal RJR$_REWIND = 24;                    !  rewind
;  0     literal RJR$_SEARCH = 25;                    !  search
;  0     literal RJR$_SPACE = 26;                     !  block I/O space
;  0     literal RJR$_TRUNCATE = 27;                  !  truncate
;  0     literal RJR$_UPDATE = 28;                    !  update
;  0     literal RJR$_WAIT = 29;                      !  wait
;  0     literal RJR$_WRITE = 30;                     !  block I/O write
;  0     literal RJR$_TPT = 31;                       !  truncate on PUT
;  0     literal RJR$_MAXOPER = 31;                   !  oper limit
;  0     literal RJR$C_NOJNL = 0;                     !  null jnl type
;  0     literal RJR$C_RMS_AI = 1;                    !  after-image journal
;  0     literal RJR$C_RMS_BI = 2;                    !  before-image journal
;  0     literal RJR$C_RMS_RU = 3;                    !  recovery unit
;  0     literal RJR$C_MAXJNL = 3;                    !  jnl type limit
;  0     literal RJR$K_HDRLEN = 56;                   !  common header len
;  0     literal RJR$C_HDRLEN = 56;                   !  common header len
;  0     literal RJR$K_C_FIBLEN = 64;
;  0     literal RJR$C_C_FIBLEN = 64;
;  0     literal RJR$K_RECATRLEN = 32;
;  0     literal RJR$C_RECATRLEN = 32;
;  0     literal RJR$K_FILNAMLEN = 452;
;  0     literal RJR$C_FILNAMLEN = 452;
;  0     literal RJR$K_RECLEN = 72;                   !  record entry len
;  0     literal RJR$C_RECLEN = 72;                   !  record entry len
;  0     literal RJR$K_BLKLEN = 68;                   !  block i/o entry len
;  0     literal RJR$C_BLKLEN = 68;                   !  block i/o entry len
;  0     literal RJR$K_EXTLEN = 122;                  !  extend entry len
;  0     literal RJR$C_EXTLEN = 122;                  !  extend entry len
;  0     literal RJR$K_BKTLEN = 68;                   !  bucket entry len
;  0     literal RJR$C_BKTLEN = 68;                   !  bucket entry len
;  0     literal RJR$K_AT_RECLEN = 76;
;  0     literal RJR$C_AT_RECLEN = 76;
;  0     literal RJR$K_BLN = 452;                     !  length of RJR descriptor in the prologue
;  0     literal RJR$C_BLN = 452;                     !  length of RJR descriptor in the prologue
;  0     literal RJR$S_RJRDEF = 452;
;  0     macro RJR$R_FLAGS_OVERLAY = 0,0,16,0 %;      !  control flags
;  0     macro RJR$W_FLAGS = 0,0,16,0 %;
;  0     macro RJR$R_FLAGS_BITS = 0,0,16,0 %;
;  0     macro RJR$B_VERSION = 2,0,8,0 %;             !  RMS journal version #
;  0     macro RJR$B_ENTRY_TYPE = 3,0,8,0 %;          !  journal entry type
;  0     macro RJR$B_ORG = 4,0,8,0 %;                 !  file organization
;  0     macro RJR$B_OPER = 5,0,8,0 %;                !  RMS operation id
;  0     macro RJR$B_JNL_TYPE = 6,0,8,0 %;            !  journaling type
;  0     macro RJR$T_JNLID = 8,0,0,0 %;
;  0     literal RJR$S_JNLID = 28;                    !
;  0     macro RJR$T_VOLNAM = 8,0,0,0 %;
;  0     literal RJR$S_VOLNAM = 12;                   !  volume name
;  0     macro RJR$T_FID = 20,0,0,0 %;
```

```
;      0       literal RJR$S_FID = 6;                       !  file id
;      0       macro RJR$Q_ID_DATE = 28,0,0,0 %;
;      0       literal RJR$S_ID_DATE = 8;                   !  time
;      0       macro RJR$L_AT_STS = 36,0,32,0 %;            !  status of operation
;      0       macro RJR$L_AT_STV = 40,0,32,0 %;            !  secondary status
;      0       macro RJR$L_AT_CTX = 44,0,32,0 %;            !  user FAB/RAB CTX field
;      0       macro RJR$R_ENTRY_OVERLAY = 56,0,0,0 %;
;      0       macro RJR$R_FILENAME_ENTRY = 56,0,0,0 %;
;      0       literal RJR$S_FILENAME_ENTRY = 396;
;      0       macro RJR$R_ATR_FLAGS_OVERLAY = 60,0,32,0 %;
;      0       macro RJR$L_ATR_FLAGS = 60,0,32,0 %;
;      0       macro RJR$R_ATR_FLAGS_BITS = 60,0,8,0 %;
;      0       macro RJR$V_ATR_UCHAR = 60,0,1,0 %;          !  UCHAR attribute present
;      0       macro RJR$V_ATR_PROT = 60,1,1,0 %;           !  PROT attribute present
;      0       macro RJR$V_ATR_UIC = 60,2,1,0 %;            !  UIC attribute present
;      0       macro RJR$V_ATR_REC = 60,3,1,0 %;            !  RECORD attributes present
;      0       macro RJR$V_ATR_EXPIRE = 60,4,1,0 %;         !  EXPIRATION present
;      0       macro RJR$L_UIC = 64,0,32,0 %;               !  owner UIC
;      0       macro RJR$L_PROT = 68,0,32,0 %;              !  prot mask
;      0       macro RJR$L_ALLOC = 72,0,32,0 %;             !  initial allocation (audit)
;      0       macro RJR$L_UCHAR = 76,0,32,0 %;             !  user characteristics (create)
;      0       macro RJR$Q_EXPIRE = 80,0,0,0 %;
;      0       literal RJR$S_EXPIRE = 8;                    !  expiration date (create)
;      0       macro RJR$B_FNS = 88,0,8,0 %;                !  size of file name
;      0       macro RJR$B_FAC = 90,0,8,0 %;                !  file access (audit)
;      0       macro RJR$B_SHR = 91,0,8,0 %;                !  sharing allowed (audit)
;      0       macro RJR$W_DID = 92,0,0,0 %;
;      0       literal RJR$S_DID = 6;                       !  directory ID (create, volume recovery)
;      0       macro RJR$T_C_FIB = 100,0,0,0 %;
;      0       literal RJR$S_C_FIB = 64;                    !  FIB (create)
;      0       macro RJR$T_REC_ATTR = 164,0,0,0 %;
;      0       literal RJR$S_REC_ATTR = 32;                 !  record attributes (create)
;      0       macro RJR$T_FILENAME = 196,0,0,0 %;
;      0       literal RJR$S_FILENAME = 256;                !  full filename
;      0       macro RJR$R_RECORD_ENTRY = 56,0,0,0 %;
;      0       literal RJR$S_RECORD_ENTRY = 16;             !  record entry
;      0       macro RJR$L_CHKSUM = 60,0,32,0 %;            !  checksum of old record
;      0       macro RJR$R_RFA_OVERLAY = 64,0,0,0 %;
;      0       macro RJR$W_RFA = 64,0,0,0 %;
;      0       literal RJR$S_RFA = 6;                       !  RFA of record
;      0       macro RJR$R_RFA_FIELDS = 64,0,0,0 %;
;      0       macro RJR$L_RFA0 = 64,0,32,0 %;              !  alternate RFA def
;      0       macro RJR$W_RFA4 = 68,0,16,0 %;
;      0       macro RJR$L_RRN = 64,0,32,0 %;               !  relative record number
;      0       macro RJR$W_RSIZE = 70,0,16,0 %;             !  record size
;      0       macro RJR$T_RIMAGE = 72,0,0,0 %;             !  record date
;      0       !
;      0       !  The block entry is common to both AT and block IO journaling.
;      0       !
;      0       macro RJR$R_BLOCK_ENTRY = 56,0,0,0 %;
;      0       literal RJR$S_BLOCK_ENTRY = 12;              !  block entry
;      0       macro RJR$L_BLOCK_VBN = 60,0,32,0 %;         !  vbn of block
;      0       macro RJR$L_BLOCK_SIZE = 64,0,32,0 %;        !  transfer size (AT)
;      0       macro RJR$T_BLOCK = 68,0,0,0 %;              !  block data
;      0       !
;      0       !  The extend entry is common to both AT and AI journaling.
;      0       !
```

```
macro RJR$R_EXTEND_ENTRY = 56,0,0,0 %;
literal RJR$S_EXTEND_ENTRY = 66;            !  extend entry
macro RJR$T_FILL_EXT2 = 60,0,0,0 %;
literal RJR$S_FILL_EXT2 = 32;               !  no longer FIB
!  currently unused
macro RJR$R_EXT_FLAGS_OVERLAY = 92,0,32,0 %;
macro RJR$L_EXT_FLAGS = 92,0,32,0 %;
macro RJR$R_EXT_FLAGS_BITS = 92,0,8,0 %;
macro RJR$V_EXT_USE_XAB = 92,0,1,0 %;       !  ALL XAB fields present
!
!  Fields EXT_AOP (unused) through EXT_RFI are in same relative locations as
!  the same fields in allocation XAB.
!
macro RJR$B_EXT_AOP = 100,0,8,0 %;          !  align options
macro RJR$B_EXT_ALN = 101,0,8,0 %;          !  alignment boundary
macro RJR$W_EXT_VOL = 102,0,16,0 %;         !  relative volume number
macro RJR$L_EXT_LOC = 104,0,32,0 %;         !  location
macro RJR$L_EXT_ALQ = 108,0,32,0 %;         !  allocation quantity
macro RJR$W_EXT_DEQ = 112,0,16,0 %;         !  default extension
macro RJR$B_EXT_BKZ = 114,0,8,0 %;          !  bucket size
macro RJR$B_EXT_AID = 115,0,8,0 %;          !  area ID
macro RJR$W_EXT_RFI = 116,0,0,0 %;
literal RJR$S_EXT_RFI = 6;                  !  related file IFI
macro RJR$T_EXT_ENDALL = 122,0,0,0 %;       !  end of all info
macro RJR$R_BUCKET_ENTRY = 56,0,0,0 %;
literal RJR$S_BUCKET_ENTRY = 12;            !  BUCKET entry
macro RJR$L_BKT_VBN = 60,0,32,0 %;          !  bucket vbn
macro RJR$W_BKT_SIZE = 64,0,16,0 %;         !  bucket size
macro RJR$W_JBKT_SIZE = 66,0,16,0 %;        !  actual size of
!  journaled bucket data
macro RJR$T_BUCKET = 68,0,0,0 %;            !  bucket data
macro RJR$R_AT_RECORD = 56,0,0,0 %;
literal RJR$S_AT_RECORD = 20;
macro RJR$L_AT_ROP = 60,0,32,0 %;           !  record options
macro RJR$B_AT_KRF = 64,0,8,0 %;            !  key of reference
macro RJR$B_AT_KSZ = 65,0,8,0 %;            !  key size
macro RJR$B_AT_RAC = 66,0,8,0 %;            !  record access mode
macro RJR$R_AT_RFA_OVERLAY = 68,0,0,0 %;
macro RJR$W_AT_RFA = 68,0,0,0 %;
literal RJR$S_AT_RFA = 6;                   !  RFA of record
macro RJR$R_AT_RFA_FIELDS = 68,0,0,0 %;
macro RJR$L_AT_RFA0 = 68,0,32,0 %;          !  alternate RFA def
macro RJR$W_AT_RFA4 = 72,0,16,0 %;
macro RJR$L_AT_RRN = 68,0,32,0 %;           !  relative record number
macro RJR$T_AT_KEY = 76,0,0,0 %;            !  key if used
!
!  The FILENAME entry is used to describe filename information required to:
!
!  1.   Identify a stream of journal entries with a particular file.
!       Used in particular for AI volume recovery or roll back RUs on remount.
!
!  2.   Record the information required to re-create a file for AI journaling.
!
!  3.   Record all the required audit-trail information for a $CREATE.
!
```

```
;                               COMMAND QUALIFIERS

;       BLISS/LIB=LIB$:RMS.L32/LIS=LIS$:RMS.LST LIB$:RMS.R32

; Run Time:          00:37.5
; Elapsed Time:      00:42.1
; Lines/CPU Min:      8451
; Lexemes/CPU-Min: 76835
; Memory Used:  590 pages
; Library Precompilation Complete
```

0314 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RMSCALLS
MAR

RMSIDXLNK
R32

RMS
LST

RMS32MAC
MAR

RMSMSCMAC
MAR

UTLDEF
R32

RMSIDXMAC
R32

UTLDEFUND
R32

RMSINTDEF
LST

RMSIDXDEF
R32

NT0MACROS
MAR