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

```
RRRRRRR   MM      MM  SSSSSSSS   IIIIII   NN      NN  TTTTTTTTTT  DDDDDDD    EEEEEEEEEE  FFFFFFFFFF
RRRRRRRR  MM      MM  SSSSSSSS   IIIIII   NN      NN  TTTTTTTTTT  DDDDDDD    EEEEEEEEEE  FFFFFFFFFF
RR    RR  MMMM  MMMM  SS           II     NN      NN      TT      DD    DD   EE          FF
RR    RR  MMMM  MMMM  SS           II     NN      NN      TT      DD    DD   EE          FF
RR    RR  MM  MM  MM  SS           II     NNNN    NN      TT      DD    DD   EE          FF
RR    RR  MM  MM  MM  SS           II     NNNN    NN      TT      DD    DD   EE          FF
RRRRRRRR  MM      MM    SSSSSS     II     NN  NN  NN      TT      DD    DD   EEEEEEEE    FFFFFFFF
RRRRRRRR  MM      MM    SSSSSS     II     NN  NN  NN      TT      DD    DD   EEEEEEEE    FFFFFFFF
RR  RR    MM      MM        SS     II     NN    NNNN      TT      DD    DD   EE          FF
RR  RR    MM      MM        SS     II     NN    NNNN      TT      DD    DD   EE          FF
RR    RR  MM      MM        SS     II     NN      NN      TT      DD    DD   EE          FF
RR    RR  MM      MM        SS     II     NN      NN      TT      DD    DD   EE          FF
RR    RR  MM      MM  SSSSSSSS   IIIIII   NN      NN      TT      DDDDDDD    EEEEEEEEEE  FF      ....
RR    RR  MM      MM  SSSSSSSS   IIIIII   NN      NN      TT      DDDDDDD    EEEEEEEEEE  FF      ....
                                                                                               ....


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
```

```
0001  0    !++
0002  0    !
0003  0    ! UTLDEF.B32 - UTILITY DEFINITION MACROS FOR BLISS PROCESSING
0004  0    !              OF STARLET DEFINITION MACROS.
0005  0    !       Version 'V04-000'
0006  0    !
0007  0    !--
0008  0    !*******************************************************************
0009  0    !*                                                                 *
0010  0    !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                        *
0011  0    !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.         *
0012  0    !*  ALL RIGHTS RESERVED.                                           *
0013  0    !*                                                                 *
0014  0    !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
0015  0    !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH LICENSE  AND WITH THE   *
0016  0    !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
0017  0    !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
0018  0    !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0019  0    !*  TRANSFERRED.                                                    *
0020  0    !*                                                                 *
0021  0    !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
0022  0    !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0023  0    !*  CORPORATION.                                                    *
0024  0    !*                                                                 *
0025  0    !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
0026  0    !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.         *
0027  0    !*                                                                 *
0028  0    !*                                                                 *
0029  0    !*******************************************************************
0030  0    !
0031  0    !++
0032  0    !
0033  0    ! MODIFIED BY:
0034  0    !
0035  0    !     V02-001 REFORMAT        Maria del C. Nasr        01-Aug-1980
0036  0    !
0037  0    !--
0038  0    !
0039  0    !
0040  0    ! macros to extract offsets, field widths, etc., from field extraction macros
0041  0    !
0042  0    !
0043  0    MACRO   $BYTEOFFSET (OFFSET, POSITION, WIDTH, SIGN) = OFFSET%;
0044  0
0045  0    MACRO   $BITPOSITION (OFFSET, POSITION, WIDTH, SIGN) = POSITION%;
0046  0
0047  0    MACRO   $FIELDWIDTH (OFFSET, POSITION, WIDTH, SIGN) = WIDTH%;
0048  0
0049  0    MACRO   $EXTENSION (OFFSET, POSITION, WIDTH, SIGN) = SIGN%;
0050  0
0051  0 M  MACRO   $FIELDMASK (OFFSET, POSITION, WIDTH, SIGN) =
0052  0            (1^(POSITION+WIDTH) - 1^POSITION)%;
0053  0
0054  0    !
0055  0    ! macro to generate equlst constructs
0056  0    !
0057  0    MACRO
```

```
 M 0058  0          $EQULST(P,G,I,S)[A]=
 MM 0059  0              %NAME(P,GET1ST_ A) =
 MM 0060  0                  %IF NUL2ND_ A
 MM 0061  0                  %THEN (I) + %COUNT*(S)   ! assumes i, s always generated by conversion program
 MM 0062  0                  %ELSE GET2ND_ A
 M 0063  0                   %FI %,
    0064  0
 M 0065  0          GET1ST_(A,B)=
    0066  0              A_%,
 M 0067  0          GET2ND_(A,B)=
    0068  0              B_%,          ! known non-null
 M 0069  0          NUL2ND_(A,B)=
    0070  0              %NULL(B) %;
```

```
0071  0        !    $BEGIN RMSIDXMAC,V04-000
0072  0        !
0073  0        !    MACRO DEFINITIONS FOR RMS-32 INDEX FILE ORGANIZATION
0074  0        !
0075  0        !*************************************************************
0076  0        !*                                                          *
0077  0        !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                 *
0078  0        !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.   *
0079  0        !*  ALL RIGHTS RESERVED.                                    *
0080  0        !*                                                          *
0081  0        !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
0082  0        !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
0083  0        !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
0084  0        !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
0085  0        !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
0086  0        !*  TRANSFERRED.                                            *
0087  0        !*                                                          *
0088  0        !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
0089  0        !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0090  0        !*  CORPORATION.                                            *
0091  0        !*                                                          *
0092  0        !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
0093  0        !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  *
0094  0        !*                                                          *
0095  0        !*                                                          *
0096  0        !*************************************************************
0097  0        !
0098  0        !++
0099  0        !
0100  0        !  FACILITY:    RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
0101  0        !
0102  0        !  ABSTRACT:
0103  0        !
0104  0        !
0105  0        !  ENVIRONMENT:
0106  0        !
0107  0        !        VAX/VMS OPERATING SYSTEM
0108  0        !
0109  0        !--
0110  0        !
0111  0        !  AUTHOR: D. H. Gillespie    CREATION DATE:  17-MAR-1978
0112  0        !         and W. Koenig
0113  0        !
0114  0        !
0115  0        !  MODIFIED BY:
0116  0        !
0117  0        !    V03-002 MCN0003        Maria del C. Nasr      15-Mar-1982
0118  0        !            Use new general linkage for RM$BUG3.  Take out definition
0119  0        !            for R_REC_SIZE, and R_VBN.
0120  0        !
0121  0        !    V03-001 MCN0002        Maria del C. Nasr      25-Mar-1982
0122  0        !            Add macro definition to calculate key buffer address.
0123  0        !
0124  0        !    V02-003 CDS0001        C Saether              9-Dec-1981
0125  0        !            Comment out references to CSH$M_READAHEAD flag in the
0126  0        !            $CSHFLAGS macro.
0127  0        !
```

```
0128  0    !
0129  0    !    V02-002 MCN0001       Maria del C. Nasr       22-Apr-1981
0130  0    !         Add macro definition for determining record identifier size
0131  0    !**
0132  0
0133  0
```

```
0134  0          !
0135  0          ! Define macro for psect attributes
0136  0          !
0137  0          MACRO   PSECT_ATTR = EXECUTE,PIC,GLOBAL %;
0138  0          !
0139  0          ! Define macro to extract size
0140  0          !
0141  0          MACRO   $BYTESIZE(OFFSET,POSITION,WIDTH,SIGN) = WIDTH / 8 %;
0142  0
0143  0          !
0144  0          ! Structure declarations used for system defined structrues to save typing
0145  0          !
0146  0          STRUCTURE
0147  0                  BBLOCK [O, P, S, E; N] =
0148  0                          [N]
0149  0                          (BBLOCK+O)<P,S,E>,
0150  0
0151  0                  BBLOCKVECTOR [I, O, P, S, E; N, BS] =
0152  0                          [N*BS]
0153  0                          (BBLOCKVECTOR+(O+I*BS))<P,S,E>;
0154  0
0155  0          !
0156  0          ! ***** The following two macros violate the BLISS language definition
0157  0          ! ***** in that they make use of the value of SP while building the argument
0158  0          ! ***** list.  It is the opinion of the BLISS maintainers that this usage is safe
0159  0          ! ***** from planned future optimizations.
0160  0          !
0161  0          ! Macro to call the change mode to kernel system service.
0162  0          !
0163  0          ! Macro call format is "KERNEL_CALL( ROUTINE, ARG1, ARG2, ...).
0164  0          !
0165  0          MACRO
0166  0                  KERNEL_CALL (R) =
M 0167  0                          BEGIN
M 0168  0                          EXTERNAL ROUTINE SYS$CMKRNL      : ADDRESSING_MODE (ABSOLUTE):
M 0169  0                          BUILTIN SP;
M 0170  0                          SYS$CMKRNL(4, .SP, %LENGTH-1
M 0171  0                              %IF %LENGTH GTR 1 %THEN, %REMAINING %FI)
0172  0                          END%;
0173  0
0174  0          !
0175  0          ! macro to generate a string descriptor
0176  0          !
0177  0          MACRO
M 0178  0                  DESCRIPTOR (STRING) =
0179  0                          UPLIT (%CHARCOUNT (STRING), UPLIT BYTE (STRING))%;
0180  0
0181  0          !
0182  0          ! macro to return the number of actual parameters supplied to a routine call
0183  0          !
0184  0          MACRO
M 0185  0                  ACTUALCOUNT =
M 0186  0                          BEGIN
M 0187  0                          BUILTIN AP;
M 0188  0                          .(.AP)<0,8>
M 0189  0                          END
0190  0                          %;
```

```
    0191  0
    0192  0        ! macro to generate call to bug check routine
    0193  0
M   0194  0        MACRO BUG_CHECK =
MM  0195  0              ( LINKAGE L_JSB;
MMM 0196  0                EXTERNAL ROUTINE
M   0197  0                  RM$BUG3 :  RL$JSB;
    0198  0                RM$BUG3 ()  ) %;
    0199  0
    0200  0        !
    0201  0        ! Macro used to determine record identifier size (byte or word) depending on
    0202  0        ! prologue version of the file.
    0203  0        !
M   0204  0        MACRO IRC$_ID(RECADR) =
M   0205  0              (IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
M   0206  0              THEN
M   0207  0                  .RECADR[IRC$B_ID]
M   0208  0              ELSE
    0209  0                  .RECADR[IRC$W_ID]) % ;
    0210  0
    0211  0
    0212  0        ! Macro used to determine address of key buffer wanted.  Parameter is
    0213  0        ! the keybuffer number.
    0214  0        !
    0215  0
M   0216  0        MACRO KEYBUF_ADDR(KBUFNO) =
    0217  0              .IRAB[IRB$L_KEYBUF] + .IFAB[IFB$W_KBUFSZ] * ((KBUFNO) - 1) % ;
    0218  0
    0219  0
```

```
  0220   0          !KEEP THESE DEFINITIONS IN ALPHABETICAL ORDER, PLEASE!
  0221   0
  0222   0
  0223   0          !
  0224   0          !  internal register definitions
  0225   0          !
  0226   0          !      common register definitions
  0227   0
  0228   0          MACRO
M 0229   0              COMMON_FABREG =
  0230   0                      R_FAB, R_IFAB, R_IFAB_FILE, R_IMPURE %,
  0231   0
M 0232   0              COMMON_IOREG =
  0233   0                      R_BDB, R_BKT_ADDR %,
  0234   0
M 0235   0              COMMON_RABREG =
  0236   0                      R_RAB, R_IRAB, R_IFAB, R_IMPURE%,
  0237   0
M 0238   0              COMMON_FAB_STR =
M 0239   0                      R_FAB_STR,
M 0240   0                      R_IFAB_STR,
M 0241   0                      R_IFAB_FILE_STR,
  0242   0                      R_IMPURE_STR %,
  0243   0
M 0244   0              COMMON_IO_STR =
M 0245   0                      R_BDB_STR,
  0246   0                      R_BKT_ADDR_STR %,
  0247   0
M 0248   0              COMMON_RAB_STR =
M 0249   0                      R_RAB_STR,
M 0250   0                      R_IRAB_STR,
M 0251   0                      R_IFAB_STR,
  0252   0                      R_IMPURE_STR%;
  0253   0
  0254   0
  0255   0          !      miscellaneous register definitions
  0256   0          !      the macros associated with registers will follow these conventions:
  0257   0          !              macro r_name =
  0258   0          !                      name = register_number %; (no trailing punctuation)
  0259   0          !              to be used basically in a linkage statement, and
  0260   0          !              macro r_name_str =
  0261   0          !                      r_name : ref bblock %;  (or whatever structure)
  0262   0          !              to be used basically in external or global register declarations
  0263   0          MACRO
M 0264   0              R_BDB =
  0265   0                      BDB        = 4 %,
M 0266   0              R_BKT_ADDR =
  0267   0                      BKT_ADDR = 5 %,
M 0268   0              R_FAB =
  0269   0                      FAB        = 8 %,
M 0270   0              R_IDX_DFN =
  0271   0                      IDX_DFN = 7 %,
M 0272   0              R_IFAB_FILE =
  0273   0                      IFAB_FILE = 9 %,
M 0274   0              R_IMPURE =
  0275   0                      IMPURE  = 11 %,
M 0276   0              R_IRAB =
```

```
  0277  0              IRAB     = 9 %,
M 0278  0          R_RAB =
  0279  0              RAB      = 8 %,
M 0280  0          R_REC_ADDR =
  0281  0              REC_ADDR = 6 %,
M 0282  0          R_IFAB =
  0283  0              IFAB     = 10 %,
  0284  0
  0285  0
M 0286  0          R_BDB_STR =
  0287  0              R_BDB    : REF BBLOCK %,
M 0288  0          R_BKT_ADDR_STR =
  0289  0              R_BKT_ADDR : REF BBLOCK %,
M 0290  0          R_FAB_STR =
  0291  0              R_FAB    : REF BBLOCK %,
M 0292  0          R_ID_STR =
  0293  0              R_ID     : BYTE %,
M 0294  0          R_IDX_DFN_STR =
  0295  0              R_IDX_DFN : REF BBLOCK %,
M 0296  0          R_IFAB_STR =
  0297  0              R_IFAB   : REF BBLOCK %,
M 0298  0          R_IMPURE_STR =
  0299  0              R_IMPURE : REF BBLOCK %,
M 0300  0          R_IRAB_STR =
  0301  0              R_IRAB   : REF BBLOCK %,
M 0302  0          R_RAB_STR =
  0303  0              R_RAB    : REF BBLOCK %,
M 0304  0          R_REC_ADDR_STR =
  0305  0              R_REC_ADDR : REF BBLOCK %,
M 0306  0          R_IFAB_FILE_STR =
  0307  0              R_IFAB_FILE : REF BBLOCK %,
M 0308  0          R_VBN_STR =
  0309  0              R_VBN              %;
  0310  0
```

```
0311  0      !
0312  0      !       macro to make the status codes small
0313  0      !
0314  0      MACRO WORDMASK(CODE) = (
0315  0              (CODE AND %X'FFFF'))
0316  0              %;
0317  0      MACRO RMSERR (CODE) = (
0318  0              %NAME('RMS$_',CODE) AND %X'FFFF')
0319  0              %,
0320  0
0321  0        RMSSUC (CODE) = (
0322  0            %IF %LENGTH EQL 0 %THEN
0323  0                    1
0324  0            %ELSE
0325  0                    %NAME('RMS$_',CODE) AND %X'FFFF'
0326  0            %FI)
0327  0            %;
0328  0
0329  0      !  macro to make constants that are calculated have the <0,16> attribute
0330  0
0331  0
0332  0      !       macros to make the code a little nicer
0333  0
0334  0      MACRO
0335  0      !
0336  0      ! this macro allows you to make a call to another routine
0337  0      !  (and do whatever you want in a block before the call),
0338  0      ! and if an error resulted, do whatever you want and
0339  0      !  then return with the status.
0340  0      !
0341  0        RETURN_ON_ERROR (CALL) [] =
0342  0            (LOCAL STATUS;
0343  0             IF NOT (STATUS = (CALL)) THEN
0344  0                                    (%REMAINING;
0345  0                                     RETURN .STATUS)) %,
0346  0
0347  0      !
0348  0      ! this macro is the same as the one above, except that it
0349  0      !  it returns w/ status whether or not there was an error
0350  0      !  and the caller can supply an 'else' block
0351  0      !  note:  the 'call' part and 'else' part must be separated by a comma
0352  0      !  not a semicolon and the 'else' part must be terminated by a semicolon
0353  0      !
0354  0        RETURN_ELSE (CALL) [] =
0355  0            (LOCAL STATUS;
0356  0             IF NOT (STATUS = (CALL)) THEN RETURN .STATUS
0357  0             ELSE
0358  0                    %REMAINING
0359  0                    RETURN .STATUS)
0360  0            %;
```

```
  0361  0        ! this is an internal cache macro to put the value of the flags into cshtmp
M 0362  0        MACRO IRP(A)[] =
M 0363  0                %ASSIGN (CSHTMP,CSHTMP OR %NAME('CSH$M_',A));
  0364  0                IRP(%REMAINING);
  0365  0                %;
  0366  0        !
  0367  0        ! this is an internal cache macro to verify the cache flags and set them up
M 0368  0        MACRO $CSHFLAGS(FLAGS) =
M 0369  0                COMPILETIME CSHTMP = 0;
M 0370  0                %IF NOT %NULL(FLAGS) %THEN
M 0371  0                        IRP (%REMOVE(FLAGS));
M 0372  0                %FI;
M 0373  0                %IF (CSHTMP AND CSH$M_READAHEAD) NEQ 0 %THEN
M 0374  0                        %ASSIGN (CSHTMP,CSHTMP OR CSH$M_NOWAIT);
M 0375  0                %IF (CSHTMP AND
M 0376  0                        (CSH$M_LOCK OR CSH$M_NOREAD OR CSH$M_NOBUFFER))
M 0377  0                            NEQ 0 %THEN
M 0378  0                                %ERRORMACRO ('INVALID CACHE FLAG COMBINATION');
M 0379  0                    %FI;
M 0380  0                %FI;
M 0381  0                %IF (CSHTMP AND CSH$M_NOBUFFER)  NEQ 0 %THEN
M 0382  0                        %ASSIGN (CSHTMP,CSHTMP OR CSH$M_NOREAD);
M 0383  0                %FI;
  0384  0                %;
  0385  0        !
  0386  0        ! this is a macro to call cache or cachec
  0387  0
M 0388  0        MACRO CACHE (VBN,SIZE,FLAGS,EP) =
M 0389  0            BEGIN
M 0390  0                %IF %NULL(FLAGS) %THEN
M 0391  0                        COMPILETIME CSHTMP = 0
M 0392  0                %ELSE
M 0393  0                $CSHFLAGS(FLAGS)
M 0394  0                %FI;
M 0395  0                %IF %NULL(EP) %THEN
M 0396  0                        RM$CACHE(VBN,SIZE,CSHTMP)
M 0397  0                %ELSE
M 0398  0                        %NAME('RM$CACHE',EP)(VBN,SIZE,CSHTMP)
M 0399  0                %FI
M 0400  0            END
  0401  0                %;
  0402  0        !
  0403  0        ! this is a macro to call getbkt or getbktc
  0404  0
M 0405  0        MACRO GETBKT (VBN,SIZE,FLAGS,EP) =
M 0406  0            BEGIN
M 0407  0                %IF %NULL(FLAGS) %THEN
M 0408  0                        COMPILETIME CSHTMP = 0
M 0409  0                %ELSE
M 0410  0                $CSHFLAGS(FLAGS)
M 0411  0                %FI;
M 0412  0                %IF %NULL(EP) %THEN
M 0413  0                        RM$GETBKT(VBN,SIZE,CSHTMP)
M 0414  0                %ELSE
M 0415  0                        %NAME('RM$GETBKT',EP)(VBN,SIZE,CSHTMP)
M 0416  0                %FI
M 0417  0            END
```

```
0418  0              %;
0419  00
0420  00
0421  0
```

```
0422  0      !
0423  0      ! these are macros to do probing of user structures
0424  0      !
0425  0      MACRO
0426  0          IFNORD (SIZE,ADDR,MODE) [] =
0427  0              (
0428  0              IF NOT PROBER(
0429  0                  %IF %NULL(MODE) %THEN 0 %ELSE MODE %FI,
0430  0                  SIZE,
0431  0                  ADDR)
0432  0              THEN
0433  0                  %REMAINING)
0434  0          %,
0435  0
0436  0          IFNOWRT (SIZE,ADDR,MODE) [] =
0437  0              (
0438  0              IF NOT PROBEW(
0439  0                  %IF %NULL(MODE) %THEN 0 %ELSE MODE %FI,
0440  0                  SIZE,
0441  0                  ADDR)
0442  0              THEN
0443  0                  %REMAINING)
0444  0          %,
0445  0
0446  0          IFRD (SIZE,ADDR,MODE) [] =
0447  0              (
0448  0              IF PROBER(
0449  0                  %IF %NULL(MODE) %THEN 0 %ELSE MODE %FI,
0450  0                  SIZE,
0451  0                  ADDR)
0452  0              THEN
0453  0                  %REMAINING)
0454  0          %,
0455  0      ! macros to do long probes
0456  0          READ_LONG(SIZE,ADDR,MODE) = NOT RM$NOREAD_LONG(SIZE,ADDR,MODE) %,
0457  0
0458  0          WRT_LONG(SIZE,ADDR,MODE) = NOT RM$NOWRT_LONG(SIZE,ADDR,MODE) %;
```

```
0459  0     !
0460  0     ! macro to release a bucket and clear the location where its bdb addr is stored
0461  0     !
0462  0     MACRO RELEASE(B) =
0463  0           BEGIN
0464  0           BDB = .B;
0465  0           B = 0;
0466  0           RM$RLSBKT(0);
0467  0           END%;
0468  0     !
0469  0     ! macro to make sure that an assumption made about the position of symbols
0470  0     ! in a structure is still valid
0471  0     ! the arguments to this macro must be preceded by %quote
0472  0     ! e.g., assume (%quote ifb$b_bid, %quote ifb$b_bln);
0473  0     !
0474  0     MACRO ASSUME (A,B) =
0475  0           %IF $BYTEOFFSET(A) + $BYTESIZE(A) NEQ $BYTEOFFSET(B)
0476  0           %THEN %WARN('WARNING CONSTANT HAS CHANGED')
0477  0           %FI %;
0478  0     !
0479  0     ! this version of assume is good for constants
0480  0     ! e.g. assume (irc$c_fixovhdsz + 2, irc$c_varovhdsz);
0481  0     !
0482  0     MACRO ASSUME_C (A,B) =
0483  0           %IF $BYTEOFFSET(A)   NEQ $BYTEOFFSET(B)
0484  0           %THEN %WARN('WARNING CONSTANT HAS CHANGED')
0485  0           %FI %;
0486  0     !     [ 2 0 1 , 1 0 ] R M S I D X L N K . R 3 2
0487  0     !
0488  0     !     Define subroutine linkage
0489  0     !
0490  0     !
0491  0     !**********************************************************************
0492  0     !*                                                                    *
0493  0     !* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                            *
0494  0     !* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.             *
0495  0     !* ALL RIGHTS RESERVED.                                               *
0496  0     !*                                                                    *
0497  0     !* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0498  0     !* ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE *
0499  0     !* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER *
0500  0     !* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0501  0     !* OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY *
0502  0     !* TRANSFERRED.                                                        *
0503  0     !*                                                                    *
0504  0     !* THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE *
0505  0     !* AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT *
0506  0     !* CORPORATION.                                                        *
0507  0     !*                                                                    *
0508  0     !* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS *
0509  0     !* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.             *
0510  0     !*                                                                    *
0511  0     !*                                                                    *
0512  0     !**********************************************************************
0513  0     !
0514  0     !++
0515  0     !
```

```
0516  0    ! FACILITY:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
0517  0    !
0518  0    ! ABSTRACT:
0519  0    !              This module defines all the routine linkage
0520  0    !
0521  0    !
0522  0    ! ENVIRONMENT:
0523  0    !
0524  0    !              VAX/VMS OPERATING SYSTEM
0525  0    !
0526  0    !--
0527  0    !
0528  0    ! AUTHOR: D. H. Gillespie    CREATION DATE:  17-MAR-1978
0529  0    !           and W. Koenig
0530  0    !
0531  0    ! MODIFIED BY:
0532  0    !
0533  0    !      V03-024  RAS0154          Ron Schaefer          2-May-1983
0534  0    !          Add NOPRESERVE (R2) to L_EXTEND0 linkage.
0535  0    !
0536  0    !      V03-023  MCN0020          Maria del C. Nasr     07-Apr-1983
0537  0    !          Eliminage linkages of RM$NULLKEY,and RM$COMPRESS_KEY.
0538  0    !          They will be using general linkages.  Modify L_ALLOC3,
0539  0    !          and L_EXTEND0 to use parameters instead of global registers.
0540  0    !
0541  0    !      V03-022  MCN0019          Maria del C. Nasr     05-Apr-1983
0542  0    !          Preserve all registers except R0 and R1 in linkage
0543  0    !          FABREG.  RM$XSUM0 requires a separate linkage because
0544  0    !          it cannot preserve R4.
0545  0    !
0546  0    !      V03-021  TMK0001          Todd M. Katz          26-Mar-1983
0547  0    !          Add the linkage RABREG_4.
0548  0    !
0549  0    !      V03-020  MCN0018          Maria del C. Nasr     24-Mar-1983
0550  0    !          Define new general linkages.  Also, since the linkages
0551  0    !          have changed so much, eliminate all history comments.
0552  0    !
0553  0    !*************************************************************************
```

```
0554  0      !
0555  0      !      This module defines all the routine linkage for RMS-32 index file
0556  0      !      organization.
0557  0      !
0558  0      !      KEEP THESE DEFINITIONS IN ALPHABETICAL ORDER PLEASE
0559  0      !
0560  0      !      The following conventions will be used for linkage macros:
0561  0      !
0562  0      !              MACRO L_NAME =
0563  0      !                   RL$NAME =
0564  0      !                   JSB (REGISTERS) :
0565  0      !                   GLOBAL (REGISTER DEFINITIONS) %;
0566  0      !
0567  0      !              The register definitions are macros of the forms
0568  0      !                   COMMON_FABREG, COMMON_RABREG, COMMON_IOREG, etc.
0569  0      !                   or R_REGNAME as described in RMSIDXMAC.R32
0570  0      !
0571  0      MACRO
0572  0
M 0573  0              L_ALDBUF =
M 0574  0                   RL$ALDBUF =
M 0575  0                   JSB (REGISTER = 5) :
M 0576  0                   GLOBAL (R_IMPURE, R_IFAB)
M 0577  0                   NOPRESERVE (2,3,4)
0578  0                   NOTUSED (8,9) %,
0579  0
M 0580  0              L_ALLOC3 =
M 0581  0                   RL$ALLOC3 =
M 0582  0                   JSB (REGISTER = 7; REGISTER = 1, REGISTER = 2) :
0583  0                   GLOBAL (R_IFAB) %,
0584  0
M 0585  0              L_BDBALLOC =
M 0586  0                   RL$BDBALLOC =
M 0587  0                   JSB (REGISTER = 4, REGISTER = 5) :
M 0588  0                   GLOBAL (COMMON_RABREG)
0589  0                   NOPRESERVE (2,3,4,5,6) %,
0590  0
M 0591  0              L_CACHE =
M 0592  0                   RL$CACHE =
M 0593  0                   JSB (REGISTER = 1, REGISTER = 2, REGISTER = 3) :
M 0594  0                   GLOBAL (COMMON_IOREG)
M 0595  0                   NOPRESERVE (1,2,3)
0596  0                   NOTUSED (8,9,10,11) %,
0597  0
M 0598  0              L_CHECK_SEGMENT=
M 0599  0                   RL$CHECK_SEGMENT =
M 0600  0                   JSB (REGISTER = 0, REGISTER = 4, REGISTER = 2) :
M 0601  0                   GLOBAL (R_IDX_DFN)
M 0602  0                   NOPRESERVE (2,4,5)
0603  0                   PRESERVE (1) %,
0604  0
M 0605  0              L_CHKSUM =
M 0606  0                   RL$CHKSUM =
M 0607  0                   JSB (REGISTER = 5) :
0608  0                   NOPRESERVE (0,1,2) %,
0609  0
M 0610  0              L_COMPARE_KEY =
```

```
M 0611  0              RL$COMPARE_KEY =
M 0612  0                    JSB (REGISTER = 1, REGISTER = 3, REGISTER = 0) :
M 0613  0                    GLOBAL (R_IDX_DFN)
  0614  0                    NOPRESERVE (3) %,
  0615  0
M 0616  0              L_ERROR_LINK1 =
M 0617  0                    RL$ERROR_LINK1 =
M 0618  0                    JSB () :
M 0619  0                    GLOBAL (COMMON_RABREG)
  0620  0                    PRESERVE (0) %,
  0621  0
M 0622  0              L_ERROR_LINK2 =
M 0623  0                    RL$ERROR_LINK2 =
M 0624  0                    JSB () :
M 0625  0                    GLOBAL (COMMON_RABREG, R_IDX_DFN)
  0626  0                    PRESERVE (0) %,
  0627  0
M 0628  0              L_EXTEND0 =
M 0629  0                    RL$EXTEND0 =
M 0630  0                    JSB (REGISTER = 5, REGISTER = 6; REGISTER = 1, REGISTER = 6) :
M 0631  0                    GLOBAL (COMMON_FABREG)
  0632  0                    NOPRESERVE (2,3,4,5) %,
  0633  0
M 0634  0              L_FABREG =
M 0635  0                    RL$FABREG =
M 0636  0                    JSB () :
M 0637  0                    GLOBAL (COMMON_FABREG)
  0638  0                    NOPRESERVE (0,1) %,
  0639  0
M 0640  0              L_FABREG_7 =
M 0641  0                    RL$FABREG_7 =
M 0642  0                    JSB () :
  0643  0                    GLOBAL (COMMON_FABREG, R_IDX_DFN) %,
  0644  0
M 0645  0              L_GETSPC =
M 0646  0                    RL$GETSPC =
M 0647  0                    JSB (REGISTER = 1, REGISTER = 2; REGISTER = 1) :
M 0648  0                    GLOBAL (R_IMPURE)
M 0649  0                    NOPRESERVE (2,3,4)
  0650  0                    NOTUSED (8,9,10) %,
  0651  0
M 0652  0              L_JSB =
M 0653  0                    RL$JSB =
  0654  0                    JSB () %,
  0655  0
M 0656  0              L_JSB01 =
M 0657  0                    RL$JSB01 =
M 0658  0                    JSB (REGISTER = 0, REGISTER = 1) :
M 0659  0                    GLOBAL (R_BKT_ADDR, R_REC_ADDR, R_IDX_DFN, R_IRAB, R_IFAB)
  0660  0                    NOPRESERVE (0,1) %,
  0661  0
M 0662  0              L_LINK_7_10_11 =
M 0663  0                    RL$LINK_7_10_11 =
M 0664  0                    JSB () :
M 0665  0                    GLOBAL (R_IDX_DFN, R_IFAB, R_IMPURE)
  0666  0                    NOPRESERVE (0,1) %,
  0667  0
```

```
M 0668   0              L_PRESERVE1 =
M 0669   0                      RL$PRESERVE1 =
M 0670   0                      JSB () :
M 0671   0                      GLOBAL (COMMON_RABREG, R_BDB, R_REC_ADDR, R_IDX_DFN)
  0672   0                      PRESERVE (1) %,
  0673   0
M 0674   0              L_QUERY_AND_LOCK =
M 0675   0                      RL$QUERY_AND_LOCK =
M 0676   0                      JSB (REGISTER = 1, REGISTER = 2) :
M 0677   0                      GLOBAL (COMMON_RABREG)
  0678   0                      NOPRESERVE (3) %,
  0679   0
M 0680   0              L_RABREG =
M 0681   0                      RL$RABREG =
M 0682   0                      JSB () :
M 0683   0                      GLOBAL (COMMON_RABREG)
M 0684   0                      NOPRESERVE (0,T) %,
  0685   0
M 0686   0              L_RABREG_4 =
M 0687   0                      RL$RABREG_4 =
M 0688   0                      JSB () :
M 0689   0                      GLOBAL (COMMON_RABREG, R_BDB)
  0690   0                      NOPRESERVE (0,T) %,
  0691   0
M 0692   0              L_RABREG_4567 =
M 0693   0                      RL$RABREG_4567 =
M 0694   0                      JSB () :
M 0695   0                      GLOBAL (COMMON_RABREG, COMMON_IOREG, R_REC_ADDR, R_IDX_DFN)
  0696   0                      NOPRESERVE (0,T) %,
  0697   0
M 0698   0              L_RABREG_457 =
M 0699   0                      RL$RABREG_457 =
M 0700   0                      JSB () :
M 0701   0                      GLOBAL (COMMON_RABREG, COMMON_IOREG, R_IDX_DFN)
  0702   0                      NOPRESERVE (0,T) %,
  0703   0
M 0704   0              L_RABREG_467 =
M 0705   0                      RL$RABREG_467 =
M 0706   0                      JSB () :
M 0707   0                      GLOBAL (COMMON_RABREG, R_BDB, R_REC_ADDR, R_IDX_DFN)
  0708   0                      NOPRESERVE (0,T) %,
  0709   0
M 0710   0              L_RABREG_567 =
M 0711   0                      RL$RABREG_567 =
M 0712   0                      JSB () :
M 0713   0                      GLOBAL (COMMON_RABREG, R_BKT_ADDR, R_REC_ADDR, R_IDX_DFN)
  0714   0                      NOPRESERVE (0,T) %,
  0715   0
M 0716   0              L_RABREG_67 =
M 0717   0                      RL$RABREG_67 =
M 0718   0                      JSB () :
M 0719   0                      GLOBAL (COMMON_RABREG, R_REC_ADDR, R_IDX_DFN)
  0720   0                      NOPRESERVE (0,T) %,
  0721   0
M 0722   0              L_RABREG_7 =
M 0723   0                      RL$RABREG_7 =
M 0724   0                      JSB () :
```

```
M 0725  0              GLOBAL (COMMON_RABREG, R_IDX_DFN)
  0726  0              NOPRESERVE (0,T) %,
  0727  0
M 0728  0          L_REC_OVHD =
M 0729  0              RL$REC_OVHD =
M 0730  0              JSB (REGISTER = 1; REGISTER = 1) :
  0731  0              GLOBAL (R_REC_ADDR, R_IDX_DFN, R_IFAB) %,
  0732  0
M 0733  0          L_RELEASE =
M 0734  0              RL$RELEASE =
M 0735  0              JSB (REGISTER = 3) :
M 0736  0              GLOBAL (R_BDB, R_IRAB, R_IFAB, R_IMPURE)
M 0737  0              NOPRESERVE (1,2)
  0738  0              NOTUSED (8) %,
  0739  0
M 0740  0          L_RELEASE_FAB =
M 0741  0              RL$RELEASE_FAB =
M 0742  0              JSB (REGISTER = 3) :
M 0743  0              GLOBAL (R_BDB, R_IFAB, R_IFAB_FILE, R_IMPURE)
M 0744  0              NOPRESERVE (1,2)
  0745  0              NOTUSED(8) %,
  0746  0
M 0747  0          L_RETSPC =
M 0748  0              RL$RETSPC =
M 0749  0              JSB (REGISTER= 2, REGISTER = 3, REGISTER = 4) :
M 0750  0              GLOBAL (R_IMPURE)
M 0751  0              NOPRESERVE (2,3,5)
  0752  0              NOTUSED (8,9,10) %,
  0753  0
M 0754  0          L_SIDR_FIRST =
M 0755  0              RL$SIDR_FIRST =
M 0756  0              JSB (STANDARD; REGISTER = 1, REGISTER = 2) :
  0757  0              GLOBAL (R_REC_ADDR, R_IDX_DFN, COMMON_RABREG) %,
  0758  0
M 0759  0          L_XSUMO =
M 0760  0              RL$XSUMO =
M 0761  0              JSB () :
M 0762  0              GLOBAL (COMMON_FABREG)
  0763  0              NOPRESERVE (0,T,4) %;
```

```
0782  0    ! *  DIGITAL assumes no responsibility for the use or  reliability  of  its  *
0783  0    ! *  software on equipment which is not supplied by DIGITAL.                 *
0784  0    ! *                                                                          *
0785  0    ! ********************************************************************************
0786  0    ! ********************************************************************************
0787  0    ! 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
0788  0    ! ********************************************************************************
0789  0
0790  0
0791  0    !*** MODULE $IFBDEF ***
0792  0    !
0793  0    !   NOTE:  The fields thru JNLBDB inclusive are common between the ifb and irb
0794  0    !
0795  0    literal IFB$C_BID = 11;                    !  ifab id code
0796  0    literal IFB$M_PUT = 1;
0797  0    literal IFB$M_GET = 2;
0798  0    literal IFB$M_DEL = 4;
0799  0    literal IFB$M_UPD = 8;
0800  0    literal IFB$M_TRN = 16;
0801  0    literal IFB$M_BIO = 32;
0802  0    literal IFB$M_BRO = 64;
0803  0    literal IFB$M_EXE = 128;
0804  0    literal IFB$C_SEQ = 0;                     !  sequential
0805  0    literal IFB$C_REL = 1;                     !  relative
0806  0    literal IFB$C_IDX = 2;                     !  indexed
0807  0    literal IFB$C_DIR = 3;                     !  direct
0808  0    literal IFB$C_MAXORG = 2;                  !  release 1.5 maximum
0809  0    literal IFB$K_FHAEND = 102;                !  end of file header attributes
0810  0    literal IFB$C_FHAEND = 102;                !  end of file header attributes
0811  0    literal IFB$C_KBUFNUM = 6;                 !  constant - the number of key buffers allocated
0812  0    literal IFB$M_ONLY_RU = 1;
0813  0    literal IFB$M_RU = 2;
0814  0    literal IFB$M_BI = 4;
0815  0    literal IFB$M_AI = 8;
0816  0    literal IFB$M_AT = 16;
0817  0    literal IFB$M_NEVER_RU = 32;
0818  0    literal IFB$M_RU_RECVR = 1;
0819  0    literal IFB$M_AI_RECVR = 2;
0820  0    literal IFB$M_BI_RECVR = 4;
0821  0    literal IFB$M_VALID_AT = 1;
0822  0    literal IFB$M_JNL = 2;
0823  0    literal IFB$M_RUP = 4;
0824  0    literal IFB$M_RU_RLK = 8;
0825  0    literal IFB$M_DONE_ASS_JNL = 16;
0826  0    literal IFB$K_BLN_SEQ = 172;
0827  0    literal IFB$C_BLN_SEQ = 172;
0828  0    !  --
0829  0    !
0830  0    !    organization-dependent fields
0831  0    !
0832  0    !    the following fields are used differently
0833  0    !    depending upon the file's organization
0834  0    !
0835  0    !  ++
0836  0    !
0837  0    !    relative org specific fields
0838  0    !
```

```
0839  0          literal IFB$S_IFBDEF = 172;
0840  0          !   (but have definitions that allow them to
0841  0          !    be referenced from the start of the ifab)
0842  0          !   ++
0843  0          !    the following bits are defined in
0844  0          !    common with the irab
0845  0          !
0846  0          macro IFB$V_BUSY = 4,0,1,0 %;           ! stream busy
0847  0          macro IFB$V_EOF = 4,1,1,0 %;            ! file positioned at eof
0848  0          macro IFB$V_PPF_IMAGE = 4,2,1,0 %;      ! flag for indirect processing of process-
0849  0          ! permanent files (restricts allowable operations)
0850  0          macro IFB$V_ASYNC = 4,3,1,0 %;          ! async i/o flag (must be zero for ifab)
0851  0          macro IFB$V_ASYNCWAIT = 4,4,1,0 %;      ! wait on async i/o (must be zero for ifab)
0852  0          !   --
0853  0          !
0854  0          !    ifab specific bits
0855  0          !
0856  0          macro IFB$V_ACCESSED = 4,5,1,0 %;       ! file is accessed
0857  0          macro IFB$V_ANSI_D = 4,6,1,0 %;         ! ansi d variable records
0858  0          macro IFB$V_RWC = 4,7,1,0 %;            ! copy of fop bit from open
0859  0          macro IFB$V_DMO = 4,8,1,0 %;            ! copy of fop bit from open
0860  0          macro IFB$V_SPL = 4,9,1,0 %;            ! copy of fop bit from open
0861  0          macro IFB$V_SCF = 4,10,1,0 %;           ! copy of fop bit from open
0862  0          macro IFB$V_DLT = 4,11,1,0 %;           ! copy of fop bit from open
0863  0          macro IFB$V_DFW = 4,12,1,0 %;           ! deferred write (copy of fop bit from $open)
0864  0          macro IFB$V_SQO = 4,13,1,0 %;           ! sequential operations only
0865  0          macro IFB$V_PPF_INPUT = 4,14,1,0 %;     ! this is command 'input' stream
0866  0          macro IFB$V_NFS = 4,15,1,0 %;           ! non-file structured flag
0867  0          macro IFB$V_WRTACC = 4,16,1,0 %;        ! logical or of fac bits:
0868  0          !    put, upd, del, trn
0869  0          macro IFB$V_MSE = 4,17,1,0 %;           ! multi-streams enabled
0870  0          macro IFB$V_CREATE = 4,18,1,0 %;        ! set if doing create (may be "create if")
0871  0          macro IFB$V_NORECLK = 4,19,1,0 %;       ! record locking not required
0872  0          !   (i.e., no shared access or multi-stream)
0873  0          macro IFB$V_RW_ATTR = 4,20,1,0 %;       ! set if file attributes must be re-written
0874  0          macro IFB$V_TMP = 4,21,1,0 %;           ! temporary file (i.e., no directory entry)
0875  0          macro IFB$V_TEF = 4,22,1,0 %;           ! truncate at eof due to large auto extend
0876  0          macro IFB$V_STALL_LOCK = 4,23,1,0 %;    ! RMS is stalled for file lock
0877  0          macro IFB$V_SEQFIL = 4,24,1,0 %;        ! this is really a sequential file being shared
0878  0          macro IFB$V_SEARCH = 4,25,1,0 %;        ! search ifab - left during wildcard operations
0879  0          macro IFB$V_RMS_STALL = 4,26,1,0 %;     ! RMS is stalled on this file operation
0880  0          macro IFB$V_RESTART = 4,27,1,0 %;       ! Reopen or recreate operation in progress
0881  0          macro IFB$V_FILEFOUND = 4,28,1,0 %;     ! A file was found on a search operation
0882  0          macro IFB$V_DAP_OPEN = 4,29,1,0 %;      ! open/create function was performed via dap
0883  0          macro IFB$V_DAP = 4,30,1,0 %;           ! data access protocol transmission
0884  0          macro IFB$V_NSP = 4,31,1,0 %;           ! network services protocol transmission
0885  0          macro IFB$L_PRIM_DEV = 0,0,32,0 %;      ! device characteristics bits
0886  0          !   (for primary device - bit encoding same as for fab)
0887  0          macro IFB$L_BKPBITS = 4,0,32,0 %;       ! bookkeeping bits
0888  0          !
0889  0          macro IFB$B_BID = 8,0,8,0 %;            ! block id
0890  0          macro IFB$B_BLN = 9,0,8,0 %;            ! block length in longwords
0891  0          macro IFB$B_MODE = 10,0,8,0 %;          ! caller's mode
0892  0          macro IFB$B_EFN = 11,0,8,0 %;           ! event flag used for synchronous qio
0893  0          macro IFB$L_IOS = 12,0,32,0 %;          ! internal i/o status block
0894  0          macro IFB$L_BWB = 12,0,32,0 %;          ! bucket wait block for inter stream waiting
0895  0          macro IFB$W_IOS2 = 14,0,16,0 %;         ! high word of io status block
```

```
0896   0        macro IFB$L_IOS4 = 16,0,32,0 %;          | 2nd longword of io status block
0897   0        macro IFB$L_ASBADDR = 20,0,32,0 %;       | address of asynchronous context block
0898   0        macro IFB$L_ARGLST = 24,0,32,0 %;        | user call parameters addr
0899   0        macro IFB$L_IRAB_LNK = 28,0,32,0 %;      | pointer to irab(s)
0900   0        macro IFB$W_CHNL = 32,0,16,0 %;          | i/o channel number
0901   0        macro IFB$B_FAC = 34,0,8,0 %;            | file access
0902   0        macro IFB$V_PUT = 34,0,1,0 %;            | (same as in fab's fac field)
0903   0        macro IFB$V_GET = 34,1,1,0 %;
0904   0        macro IFB$V_DEL = 34,2,1,0 %;
0905   0        macro IFB$V_UPD = 34,3,1,0 %;
0906   0        macro IFB$V_TRN = 34,4,1,0 %;
0907   0        macro IFB$V_BIO = 34,5,1,0 %;
0908   0        macro IFB$V_BRO = 34,6,1,0 %;
0909   0        macro IFB$V_EXE = 34,7,1,0 %;
0910   0        !   note: if both bio and bro set, implies block i/o
0911   0        !         access only allowed for this connect, resets
0912   0        !         to bro on disconnect (seq. file org. only).
0913   0        !
0914   0        macro IFB$B_ORGCASE = 35,0,8,0 %;        | copy of org for case dispatching
0915   0        macro IFB$L_LAST_FAB = 36,0,32,0 %;      | address of fab for last operation
0916   0        macro IFB$W_IFI = 40,0,16,0 %;           | Internal file Identifier, the one we gave to the user
0917   0        macro IFB$W_ECHO_ISI = 42,0,16,0 %;      | ISI of stream to echo records from SYS$INPUT
0918   0        macro IFB$L_ATJNLBUF = 44,0,32,0 %;      | address of IFAB audit trail buffer
0919   0        macro IFB$L_JNLBDB = 48,0,32,0 %;        | address of Journaling BDB for FAB operations
0920   0        ! -----*****
0921   0        macro IFB$L_EXTJNLBUF = 52,0,32,0 %;     | pointer to buffer to contain extend journal record
0922   0        macro IFB$L_FWA_PTR = 56,0,32,0 %;       | pointer to file work area control block
0923   0        macro IFB$L_NWA_PTR = 60,0,32,0 %;       | pointer to network work area control block
0924   0        macro IFB$L_BDB_FLNK = 64,0,32,0 %;      | pointer to bdb(s)
0925   0        macro IFB$L_BDB_BLNK = 68,0,32,0 %;      | bdb backward link
0926   0        macro IFB$L_DEVBUFSIZ = 72,0,32,0 %;     | device default (or bls if mt) buff size
0927   0        macro IFB$W_RTDEQ = 76,0,16,0 %;         | run-time default extend quantity
0928   0        macro IFB$B_SHR = 78,0,8,0 %;            | File sharing bits from users FAB
0929   0        macro IFB$B_AGENT_MODE = 79,0,8,0 %;     | User's FAB$V_FILE_MODE field, maximized with mode of caller
0930   0        !
0931   0        !   ++++++++++
0932   0        !
0933   0        !   the following fields must remain as is since
0934   0        !   they correspond to the rms attributes stored
0935   0        !   in the file header
0936   0        !
0937   0        macro IFB$B_RFMORG = 80,0,8,0 %;         | organization and record format
0938   0        macro IFB$V_RFM = 80,0,4,0 %;
0939   0        literal IFB$S_RFM = 4;                   | record format (n.b. constant values defined in rfm field of fab)
0940   0        macro IFB$V_ORG = 80,4,4,0 %;
0941   0        literal IFB$S_ORG = 4;                   | file organization
0942   0        macro IFB$B_RAT = 81,0,8,0 %;            | record attributes (n.b. bit offsets defined in rat field of fab)
0943   0        macro IFB$W_LRL = 82,0,16,0 %;           | longest record's length (or fixed record length)
0944   0        macro IFB$L_HBK_DISK = 84,0,32,0 %;      | hi vbn allocated (note: disk format!)
0945   0        macro IFB$L_EBK_DISK = 88,0,32,0 %;      | eof vbn (note: disk format!)
0946   0        macro IFB$W_FFB = 92,0,16,0 %;           | first free byte in eof block
0947   0        macro IFB$B_BKS = 94,0,8,0 %;            | bucket size (! vbns)
0948   0        macro IFB$B_FSZ = 95,0,8,0 %;            | record header size for vfc
0949   0        macro IFB$W_MRS = 96,0,16,0 %;           | max record size allowable
0950   0        macro IFB$W_DEQ = 98,0,16,0 %;           | default extend quantity
0951   0        macro IFB$W_GBC = 100,0,16,0 %;          | global buffer count
0952   0        ! -----*****
```

```
0953  0        macro IFB$B_DRT_REHIT = 104,0,8,0 %;        ! hit count for local dirty buffers.
0954  0        macro IFB$B_GBL_REHIT = 105,0,8,0 %;        ! rehit count for gbl buffers.
0955  0        macro IFB$L_RNS_LEN = 108,0,32,0 %;         ! resultant name string length (used as a temp field by $search)
0956  0        macro IFB$L_LOCK_BDB = 108,0,32,0 %;        ! lock bdb address (used by $extend for rel. file)
0957  0        macro IFB$L_HBK = 112,0,32,0 %;             ! hi vbn allocated.
0958  0        macro IFB$L_EBK = 116,0,32,0 %;             ! eof vbn.
0959  0        macro IFB$L_SFSB_PTR = 120,0,32,0 %;        ! pointer to shared file synchronization block
0960  0        macro IFB$L_GBSB_PTR = 124,0,32,0 %;        ! pointer to global buffer synchronization block.
0961  0        macro IFB$L_PAR_LOCK_ID = 128,0,32,0 %;     ! Parent lock ID for bucket locks (get from SFSB.)
0962  0        macro IFB$W_AVLCL = 132,0,16,0 %;           ! local buffers available.
0963  0        macro IFB$W_AVGBPB = 134,0,16,0 %;          ! gbl ptr blocks available.
0964  0        macro IFB$L_GBH_PTR = 136,0,32,0 %;         ! pointer to global header.
0965  0        macro IFB$L_AS_DEV = 140,0,32,0 %;          ! assigned device characteristics
0966  0        !                    AS_DEV and ASDEVBSIZ *)
0967  0        macro IFB$L_ASDEVBSIZ = 148,0,32,0 %;       ! assigned device buffer size
0968  0        macro IFB$L_BLBFLNK = 152,0,32,0 %;         ! Forward link to BLB chain.
0969  0        macro IFB$L_BLBBLNK = 156,0,32,0 %;         ! Back link to BLB chain.
0970  0        macro IFB$B_JNLFLG = 160,0,8,0 %;           ! journaling attribute flags
0971  0        macro IFB$V_ONLY_RU = 160,0,1,0 %;          ! Recovery Unit journaling, no access outside RU
0972  0        macro IFB$V_RU = 160,1,1,0 %;               ! Recovery Unit journaling
0973  0        macro IFB$V_BI = 160,2,1,0 %;               ! Before Image journaling
0974  0        macro IFB$V_AI = 160,3,1,0 %;               ! After Image jouraling
0975  0        macro IFB$V_AT = 160,4,1,0 %;               ! Audit Trail journaling
0976  0        macro IFB$V_NEVER_RU = 160,5,1,0 %;         ! never do RU journaling
0977  0        macro IFB$B_RECVRFLGS = 161,0,8,0 %;        ! Recovery flags
0978  0        macro IFB$V_RU_RECVR = 161,0,1,0 %;         ! Recovery Unit Rollback in progress
0979  0        macro IFB$V_AI_RECVR = 161,1,1,0 %;         ! AI Roll Forward Recovery in progress
0980  0        macro IFB$V_BI_RECVR = 161,2,1,0 %;         ! BI Roll Backward Recovery in progress
0981  0        macro IFB$B_JNLFLG2 = 162,0,8,0 %;          ! Secondary journaling flags (generally operation specific)
0982  0        macro IFB$V_VALID_AT = 162,0,1,0 %;         ! AT entry in IFB buffer is valid and should be written
0983  0        macro IFB$V_JNL = 162,1,1,0 %;              ! Journaling Initialized for this file
0984  0        macro IFB$V_RUP = 162,2,1,0 %;              ! Recovery Unit in progress
0985  0        macro IFB$V_RU_RLK = 162,3,1,0 %;           ! Fake record locking during recovery unit
0986  0        macro IFB$V_DONE_ASS_JNL = 162,4,1,0 %;     ! Journal channels already assigned
0987  0        macro IFB$L_RJB = 164,0,32,0 %;             ! RMS Journaling Block address
0988  0        macro IFB$W_BUFFER_OFFSET = 168,0,16,0 %;   !  ANSI buffer offset
0989  0        literal IFB$K_BLN_REL = 180;
0990  0        literal IFB$C_BLN_REL = 180;
0991  0        ! --
0992  0        ! ++
0993  0        !
0994  0        !    indexed org specific fields
0995  0        !
0996  0        literal IFB$S_IFBDEF1 = 180;
0997  0        macro IFB$L_MRN = 172,0,32,0 %;             ! (rel) max record number
0998  0        macro IFB$L_DVBN = 176,0,32,0 %;            ! (rel) first data bucket vbn
0999  0        literal IFB$K_BLN_IDX = 184;
1000  0        literal IFB$C_BLN_IDX = 184;
1001  0        literal IFB$K_BLN = 184;                    ! ifab length
1002  0        literal IFB$C_BLN = 184;                    ! ifab length
1003  0        ! --
1004  0        literal IFB$S_IFBDEF2 = 184;
1005  0        macro IFB$L_IDX_PTR = 172,0,32,0 %;         ! (idx) pointer to primary key index descriptor
1006  0        macro IFB$B_AVBN = 176,0,8,0 %;             ! (idx) vbn of 1st area descriptor
1007  0        macro IFB$B_AMAX = 177,0,8,0 %;             ! (idx) total number of area descriptors
1008  0        macro IFB$B_NUM_KEYS = 178,0,8,0 %;         ! (idx) ! of keys in file
1009  0        macro IFB$B_UBUFSZ = 179,0,8,0 %;           ! (idx) update buffer size for keys
```

```
1010   0        macro IFB$W_KBUFSZ = 180,0,16,0 %;          ! (idx) key buffer size
1011   0        macro IFB$B_EXTRABUF = 182,0,8,0 %;         ! (idx) number of extra buffers for 'cache'ing
1012   0        macro IFB$B_PLG_VER = 183,0,8,0 %;          ! (idx) prologue version number
1013   0
1014   0        !*** MODULE $IRBDEF ***
1015   0
1016   0        !          IRB field definitions
1017   0        !
1018   0        !          Internal rab (irb)
1019   0        !
1020   0        !          There is 1 irab per connected record access stream
1021   0        !
1022   0        !
1023   0        !    NOTE: The fields thru JNLBDB inclusive are common between the irb and ifb
1024   0        !
1025   0        literal IRB$C_BID = 10;                     ! irab code
1026   0        literal IRB$M_POSINSERT = 1;
1027   0        literal IRB$M_SRCHGT = 2;
1028   0        literal IRB$M_POSDELETE = 4;
1029   0        literal IRB$M_NEW_IDX = 8;
1030   0        literal IRB$M_SRCAGE = 16;
1031   0        literal IRB$M_NORLS_RNF = 32;
1032   0        literal IRB$M_FIRST_TIM = 64;
1033   0        literal IRB$M_PRM = 128;
1034   0        literal IRB$M_DUP_KEY = 256;
1035   0        literal IRB$M_DEL_SEEN = 512;
1036   0        literal IRB$M_LAST_GT = 1024;
1037   0        literal IRB$M_BKT_NO_LO = 1;
1038   0        literal IRB$M_NEW_BKTS = 6;
1039   0        literal IRB$M_REC_W_LO = 8;
1040   0        literal IRB$M_CONT_BKT = 16;
1041   0        literal IRB$M_CONT_R = 32;
1042   0        literal IRB$M_EMPTY_BKT = 64;
1043   0        literal IRB$M_DUPS_SEEN = 128;
1044   0        literal IRB$M_BKT_NO = 3;
1045   0        literal IRB$M_BIG_SPLIT = 4;
1046   0        literal IRB$M_SPL_IDX = 1;
1047   0        literal IRB$M_EMPT_SEEN = 2;
1048   0        literal IRB$M_VALID_AT = 1;
1049   0        literal IRB$K_BLN_REL = 100;
1050   0        literal IRB$C_BLN_REL = 100;
1051   0        ! ++
1052   0        !
1053   0        !    sequential org specific fields
1054   0        !
1055   0        literal IRB$K_BLN_SEQ = 108;
1056   0        literal IRB$C_BLN_SEQ = 108;
1057   0        literal IRB$S_IRBDEF = 108;
1058   0        !   to apply from start of irab
1059   0        ! ++
1060   0        !
1061   0        !    the following bits are defined in common
1062   0        !    with the ifab
1063   0        !
1064   0        macro IRB$V_BUSY = 4,0,1,0 %;               ! file busy
1065   0        macro IRB$V_EOF = 4,1,1,0 %;                ! stream positioned at eof
1066   0        macro IRB$V_PPF_IMAGE = 4,2,1,0 %;          ! flag for indirect processing of process-
```

```
1067  0        !   permanent file
1068  0        macro IRB$V_ASYNC = 4,3,1,0 %;              !  asynchronous i/o request
1069  0        macro IRB$V_ASYNCWAIT = 4,4,1,0 %;          !  $wait issued for asynchronous i/o request
1070  0        ! --
1071  0        !
1072  0        !     irab specific bits
1073  0        !
1074  0        macro IRB$V_FIND_LAST = 4,5,1,0 %;          !  last operation was a find
1075  0        macro IRB$V_PUTS_LAST = 4,6,1,0 %;          !  last operation was a put sequential
1076  0        macro IRB$V_BIO_LAST = 4,7,1,0 %;           !  this/last operation is/was a block i/o operation
1077  0        !  note: this bit is set only if mixed block and record
1078  0        !        operations (bro access).  after call to rm$rset
1079  0        !        refers to the current operation and bro_sw gives
1080  0        !        type of last operation.
1081  0        macro IRB$V_BRO_SW = 4,8,1,0 %;             !  switched from record operation to block i/o operation
1082  0        macro IRB$V_FIND = 4,9,1,0 %;               !  operation is a find
1083  0        macro IRB$V_RAHWBH = 4,10,1,0 %;            !  read ahead or write behind processing
1084  0        macro IRB$V_SKIP_NEXT = 4,11,1,0 %;         !  skip to next record flag for index fo
1085  0        macro IRB$V_DUP = 4,12,1,0 %;               !  duplicate records seen
1086  0        macro IRB$V_UNLOCK_RP = 4,13,1,0 %;         !  release lock on current (rp) record
1087  0        macro IRB$V_PPF_EOF = 4,14,1,0 %;           !  give one-shot rms$_eof error on sys$input
1088  0        macro IRB$V_PPF_SKIP = 4,15,1,0 %;          !  skip sys$input record ($deck), redoing $get
1089  0        !  or $find on next record
1090  0        macro IRB$V_PPF_FNDSV = 4,16,1,0 %;         !  save value for find bit when ppf_skip set
1091  0        macro IRB$V_IDX_ERR = 4,17,1,0 %;           !  index update error occurred
1092  0        macro IRB$V_RRV_ERR = 4,18,1,0 %;           !  rrv update error occurred
1093  0        macro IRB$V_UPDATE = 4,19,1,0 %;            !  operation is an update (indexed)
1094  0        macro IRB$V_UPDATE_IF = 4,20,1,0 %;         !  operation was a $PUT -> $UPDATE
1095  0        macro IRB$V_ABOVELCKD = 4,21,1,0 %;         !  level above was locked by search_tree
1096  0        macro IRB$V_GBLBUFF = 4,22,1,0 %;           !  global buffers are in use.
1097  0        macro IRB$V_CON_EOF = 4,23,1,0 %;           !  file positioned at EOF by $CONNECT (isam)
1098  0        macro IRB$V_NO_Q_WAIT = 4,24,1,0 %;         !  do not wait for enqueues on query locks
1099  0        macro IRB$V_PPF_ECHO = 4,25,1,0 %;          !  echo SYS$INPUT records to SYS$OUTPUT
1100  0        macro IRB$V_RMS_STALL = 4,26,1,0 %;         !  RMS is stalled on this record operation
1101  0        macro IRB$V_RESTART = 4,27,1,0 %;           !  Reconnect operation in progress
1102  0        macro IRB$V_DAP_CONN = 4,28,1,0 %;          !  connect function was performed via dap
1103  0        macro IRB$V_RU_DELETE = 4,29,1,0 %;         !  recovery unit deletion in progress
1104  0        macro IRB$V_RU_UNDEL = 4,30,1,0 %;          !  recovery unit un-deletion in progress
1105  0        macro IRB$V_RU_UPDATE = 4,31,1,0 %;         !  place new record is special RU UPDATE format
1106  0        !
1107  0        !   the following are alternate definitions for alternate
1108  0        !   (non-conflicting) use of the above bits
1109  0        !
1110  0        macro IRB$V_WRITE = 4,9,1,0 %;              !  operation is a write
1111  0        macro IRB$L_IFAB_LNK = 0,0,32,0 %;          !  pointer to ifab
1112  0        macro IRB$L_BKPBITS = 4,0,32,0 %;           !  bookkeeping status bits
1113  0        !
1114  0        macro IRB$B_BID = 8,0,8,0 %;                !  block id
1115  0        macro IRB$B_BLN = 9,0,8,0 %;                !  block length in longwords
1116  0        macro IRB$B_MODE = 10,0,8,0 %;              !  caller's mode
1117  0        macro IRB$B_EFN = 11,0,8,0 %;               !  event flag for synchronous io
1118  0        macro IRB$L_IOS = 12,0,32,0 %;              !  internal i/o status block
1119  0        macro IRB$L_BWB = 12,0,32,0 %;              !  bucket wait block for inter stream locking
1120  0        macro IRB$W_IOS2 = 14,0,16,0 %;             !  high word of io status block
1121  0        macro IRB$L_IOS4 = 16,0,32,0 %;             !  io status block (2nd longword)
1122  0        macro IRB$L_ASBADDR = 20,0,32,0 %;          !  address of permanent asynchronous context block
1123  0        macro IRB$L_ARGLST = 24,0,32,0 %;           !  user arg list address
```

```
1124  0       ! if async, points to copy at head
1125  0       ! of async context block
1126  0       macro IRB$L_IRAB_LNK = 28,0,32,0 %;       ! pointer to next irab
1127  0       macro IRB$L_CURBDB = 32,0,32,0 %;         ! current bdb address
1128  0       macro IRB$L_LAST_RAB = 36,0,32,0 %;       ! address of rab for last operation
1129  0       macro IRB$W_ISI = 40,0,16,0 %;            ! Internal stream Identifier, the one we gave to the user
1130  0       macro IRB$L_ATJNLBUF = 44,0,32,0 %;       ! address of IRAB audit trail journaling buffer
1131  0       macro IRB$L_JNLBDB = 48,0,32,0 %;         ! address of journaling BDB for RAB operations
1132  0       !-----*****
1133  0       macro IRB$L_IDENT = 52,0,32,0 %;          ! process unique identifier for the IRB
1134  0       macro IRB$L_RLB_LNK = 56,0,32,0 %;        ! pointer to RLBs
1135  0       macro IRB$L_NXTBDB = 60,0,32,0 %;         ! next bdb address
1136  0       macro IRB$L_NRP = 64,0,32,0 %;            ! next record pointer (relative record number)
1137  0       macro IRB$L_NRP_VBN = 64,0,32,0 %;        ! next record pointer (relative)
1138  0       macro IRB$B_CACHEFLGS = 64,0,8,0 %;       ! cacheflags for calls to getbkt,cache, etc. (indexed)
1139  0       macro IRB$B_STOPLEVEL = 65,0,8,0 %;       ! level to stop at on tree search (indexed)
1140  0       macro IRB$W_SRCHFLAGS = 66,0,16,0 %;      ! search flags (indexed)
1141  0       macro IRB$V_POSINSERT = 66,0,1,0 %;       ! position for insert
1142  0       macro IRB$V_SRCHGT = 66,1,1,0 %;          ! approximate search gt
1143  0       macro IRB$V_POSDELETE = 66,2,1,0 %;       ! position for delete
1144  0       macro IRB$V_NEW_IDX = 66,3,1,0 %;         ! need to read in new idx dsc from file
1145  0       macro IRB$V_SRCAGE = 66,4,1,0 %;          ! approximate search ge
1146  0       macro IRB$V_NORLS_RNF = 66,5,1,0 %;       ! don't release bkt on rnf error, if set
1147  0       macro IRB$V_FIRST_TIM = 66,6,1,0 %;       ! flag to indicate 1st time for seq. processing
1148  0       macro IRB$V_PRM = 66,7,1,0 %;             ! flag to indicate that the permanence bit in the bdb
1149  0       macro IRB$V_DUP_KEY = 66,8,1,0 %;         ! a duplicate key seen on scan of any data bucket
1150  0       macro IRB$V_DEL_SEEN = 66,9,1,0 %;        ! a deleted record has been encountered between current
1151  0       macro IRB$V_LAST_GT = 66,10,1,0 %;        ! result of last search of compressed key bucket was GT
1152  0       ! and a next record during a $GET/$FIND
1153  0       !   should be set
1154  0       macro IRB$L_NRP_OFF = 68,0,32,0 %;        ! next record pointer offset (relative)
1155  0       macro IRB$L_CURVBN = 68,0,32,0 %;         ! vbn of current record (relative)
1156  0       macro IRB$W_NRP_OFF = 68,0,16,0 %;        ! ''
1157  0       macro IRB$B_SPL_BITS = 68,0,8,0 %;        ! bits for splitting (indexed)
1158  0       macro IRB$V_BKT_NO_LO = 68,0,1,0 %;       ! low bit of bucket number processing
1159  0       literal IRB$S_NEW_BKTS = 2;               ! number of new buckets (0-3)
1160  0       macro IRB$V_REC_W_LO = 68,3,1,0 %;        ! if splitting at pos_insert than rec goes w/ lo
1161  0       macro IRB$V_CONT_BKT = 68,4,1,0 %;        ! middle bucket is a continuation bkt
1162  0       macro IRB$V_CONT_R = 68,5,1,0 %;          ! right bucket is a continuation bkt
1163  0       macro IRB$V_EMPTY_BKT = 68,6,1,0 %;       ! bucket contains no data records
1164  0       macro IRB$V_DUPS_SEEN = 68,7,1,0 %;       ! dups seen on scan of bucket, any key
1165  0       macro IRB$V_BKT_NO = 68,0,2,0 %;
1166  0       literal IRB$S_BKT_NO = 2;
1167  0       macro IRB$V_BIG_SPLIT = 68,2,1,0 %;
1168  0       macro IRB$V_SPL_IDX = 68,0,1,0 %;         ! split up new index record and swing pointer
1169  0       macro IRB$V_EMPT_SEEN = 68,1,1,0 %;       ! empty bucket passed over on posinsert
1170  0       macro IRB$L_RP = 72,0,32,0 %;             ! record pointer (relative record !)
1171  0       macro IRB$L_RP_VBN = 72,0,32,0 %;         ! record pointer (relative)
1172  0       macro IRB$W_POS_INS = 72,0,16,0 %;        ! offset for position for insert for put (indexed)
1173  0       macro IRB$W_SPLIT = 74,0,16,0 %;          ! first split point (indexed)
1174  0       macro IRB$L_RP_OFF = 76,0,32,0 %;         ! record pointer offset
1175  0       macro IRB$L_LST_REC = 76,0,32,0 %;        ! last record address (indexed)
1176  0       macro IRB$L_PTR_VBN = 76,0,32,0 %;        ! pointer vbn used by find_by_rrv (indexed)
1177  0       macro IRB$W_RP_OFF = 76,0,16,0 %;         ! record pointer offset
1178  0       macro IRB$W_SPLIT_1 = 76,0,16,0 %;        ! second split point -- 3-bkt split (indexed)
1179  0       macro IRB$W_SPLIT_2 = 78,0,16,0 %;        ! third split point -- 4-bkt split (indexed)
1180  0
```

```
1181   0        macro IRB$L_OWNER_ID = 80,0,32,0 %;          ! owner id used for record locks
1182   0        macro IRB$W_OWN_ID = 80,0,16,0 %;            ! index part of process id (pid)
1183   0        macro IRB$W_OWN_ISI = 82,0,16,0 %;           ! isi value for this irab
1184   0        macro IRB$B_PPF_ISI = 82,0,8,0 %;            ! isi value for this process-permanent irab
1185   0        macro IRB$B_BCNT = 84,0,8,0 %;               ! i/o buffer count
1186   0        macro IRB$B_MBC = 85,0,8,0 %;                ! multi-block count
1187   0        macro IRB$W_RSZ = 86,0,16,0 %;               ! record size from user
1188   0        macro IRB$L_RBF = 88,0,32,0 %;               ! user record buffer address
1189   0        macro IRB$B_MBF = 92,0,8,0 %;                ! Multi-buffer count from user's RAB
1190   0        macro IRB$B_JNLFLG3 = 93,0,8,0 %;            ! IRB journaling flags
1191   0        macro IRB$V_VALID_AT = 93,0,1,0 %;           ! IRB MJB contains valid AT entry to write
1192   0          ++
1193   0          !
1194   0          !   start of organization dependent fields
1195   0          !
1196   0          ++
1197   0          !
1198   0          !   used by sequential and relative files
1199   0          !
1200   0        macro IRB$W_CSIZ = 98,0,16,0 %;              ! current record size (seq)
1201   0          ++
1202   0          !
1203   0          !   relative org specific fields
1204   0          !
1205   0        macro IRB$L_TEMP0 = 100,0,32,0 %;
1206   0        macro IRB$W_ROVHDSZ = 100,0,16,0 %;          ! overhead size for record
1207   0        macro IRB$B_PRE_CCTL = 100,0,8,0 %;          ! 'pre' carriage control
1208   0        macro IRB$B_POST_CCTL = 101,0,8,0 %;         ! 'post' carriage control
1209   0        macro IRB$W_RTOTLSZ = 102,0,16,0 %;          ! total size for record
1210   0        macro IRB$L_TEMP1 = 104,0,32,0 %;
1211   0        macro IRB$B_NVBNS = 104,0,8,0 %;             ! number of vbns transferred (nxtblk1)
1212   0          !
1213   0          !   indexed org specific fields
1214   0          !
1215   0        literal IRB$K_BLN_IDX = 196;
1216   0        literal IRB$C_BLN_IDX = 196;
1217   0        literal IRB$S_IRBDEF1 = 196;
1218   0        macro IRB$L_KEYBUF = 96,0,32,0 %;            ! address of internal key buffer & update buffer
1219   0        macro IRB$L_UPDBUF = 100,0,32,0 %;           ! address of internal update buffer
1220   0        macro IRB$L_RECBUF = 104,0,32,0 %;           ! address of internal record buffer
1221   0        macro IRB$L_OLDBUF = 108,0,32,0 %;           ! address of internal old record buffer (updates only)
1222   0        macro IRB$L_RFA_VBN = 112,0,32,0 %;          ! save record vbn for nrp data
1223   0        macro IRB$L_UPD_BDB = 112,0,32,0 %;          ! save current bdb during insert operation
1224   0        macro IRB$L_LAST_VBN = 112,0,32,0 %;         ! last vbn at data level for update
1225   0        macro IRB$W_RFA_ID = 116,0,16,0 %;           ! save record id for search data
1226   0        macro IRB$W_LAST_ID = 116,0,16,0 %;          ! id for udr during update (plg 3)
1227   0        macro IRB$W_SAVE_POS = 118,0,16,0 %;         ! save duplicate position for nrp data
1228   0        macro IRB$L_NEXT_VBN = 120,0,32,0 %;         ! save next user data record VBN for nrp data
1229   0        macro IRB$L_PUTUP_VBN = 120,0,32,0 %;        ! RFA VBN of $PUT/$UPDATE record
1230   0        macro IRB$L_FIRST_VBN = 124,0,32,0 %;        ! save SIDR first element VBN for search NRP data
1231   0        macro IRB$W_NEXT_ID = 128,0,16,0 %;          ! save next user data record ID for nrp data
1232   0        macro IRB$W_PUTUP_ID = 128,0,16,0 %;         ! ID of $PUT/$UPDATE record
1233   0        macro IRB$W_FIRST_ID = 130,0,16,0 %;         ! save SIDR first element ID for search NRP data
1234   0        macro IRB$L_LOCK_BDB = 132,0,32,0 %;         ! lock bdb addr of level below on splits
1235   0        macro IRB$L_VBN_LEFT = 136,0,32,0 %;         ! left vbn of split
1236   0        macro IRB$L_MIDX_TMP1 = 136,0,32,0 %;        ! temporary one for make index
1237   0        macro IRB$L_VBN_RIGHT = 140,0,32,0 %;        ! right vbn of split
```

```
1238  0      macro IRB$L_MIDX_TMP2 = 140,0,32,0 %;         ! temporary two for make index
1239  0      macro IRB$L_VBN_MID = 144,0,32,0 %;           ! middle vbn of split
1240  0      macro IRB$L_MIDX_TMP3 = 144,0,32,0 %;         ! temporary three for make index
1241  0      macro IRB$L_NEXT_DOWN = 144,0,32,0 %;         ! used by search_tree
1242  0      macro IRB$L_REC_COUNT = 148,0,32,0 %;         ! number of current record in this bucket (plg 3)
1243  0      macro IRB$L_LST_NCMP = 152,0,32,0 %;          ! address of last key with zero front compression (plg 3)
1244  0      macro IRB$L_SPL_COUNT = 156,0,32,0 %;         ! number of the first record to be moved into new bucket
1245  0      !    when splitting indexes and SIDRs
1246  0      macro IRB$W_NID_RIGHT = 160,0,16,0 %;         ! Next record ID of the right bucket
1247  0      macro IRB$W_NID_MID = 162,0,16,0 %;           ! Next record ID of the middle bucket
1248  0      macro IRB$W_RFA_NID = 164,0,16,0 %;           ! Next record ID of the RFA bucket
1249  0      macro IRB$B_KEYSZ = 166,0,8,0 %;              ! size of key in keybuffer !2
1250  0      macro IRB$L_CUR_VBN = 168,0,32,0 %;           ! VBN of current record (primary/SIDR)
1251  0      macro IRB$L_POS_VBN = 172,0,32,0 %;           ! VBN of primary data record for NRP positioning
1252  0      macro IRB$L_UDR_VBN = 176,0,32,0 %;           ! VBN of current primary data record
1253  0      macro IRB$L_SIDR_VBN = 180,0,32,0 %;          ! SIDR array first element VBN of current record (SIDR)
1254  0      macro IRB$W_CUR_ID = 184,0,16,0 %;            ! ID of current record (primary)
1255  0      macro IRB$W_POS_ID = 186,0,16,0 %;            ! ID of primary data record for NRP positioning
1256  0      macro IRB$W_UDR_ID = 188,0,16,0 %;            ! ID of current primary data record
1257  0      macro IRB$W_SIDR_ID = 190,0,16,0 %;           ! SIDR array first element ID of current record (SIDR)
1258  0      macro IRB$W_CUR_COUNT = 192,0,16,0 %;         ! SIDR array count of current record (SIDR)
1259  0      macro IRB$B_RP_KREF = 194,0,8,0 %;            ! Key of reference by which next record is retrieved
1260  0      macro IRB$B_CUR_KREF = 195,0,8,0 %;           ! Key of reference of current record (primary/SIDR)
1261  0
1262  0      !*** MODULE $ASBDEF ***
1263  0
1264  0      !          ASB field definitions
1265  0      !
1266  0      !          Asynchronous context block (asb)
1267  0      !
1268  0      !          There is one asb per irab pointed to by irb$l_asbaddr allocated at
1269  0      !          connect and one per ifab which is dynamicaly allocated at stall
1270  0      !
1271  0      !          The asb$l_arglst is pointed to by the arglst field of the
1272  0      !          irab if the irb$v_async bookkeeping bit is set
1273  0      !
1274  0      !          All of the asb$c_bln_xxx must be longword aligned
1275  0      !
1276  0      literal ASB$C_BID = 13;                       ! asb id = 13
1277  0      literal ASB$K_BLN_FIX = 48;                   ! block length of fixed asb
1278  0      literal ASB$C_BLN_FIX = 48;                   ! block length of fixed asb
1279  0      !    regs 4 and 5 are saved on stack
1280  0      literal ASB$K_BLN_SEQ = 188;                  ! block length for seq org irab operations
1281  0      literal ASB$C_BLN_SEQ = 188;                  ! block length for seq org irab operations
1282  0      literal ASB$K_BLN_REL = 192;                  ! block length for rel org irab operations
1283  0      literal ASB$C_BLN_REL = 192;                  ! block length for rel org irab operations
1284  0      literal ASB$K_BLN_FAB = 352;                  ! block length for fab-related operations
1285  0      literal ASB$C_BLN_FAB = 352;                  ! block length for fab-related operations
1286  0      literal ASB$K_BLN_IDX = 512;
1287  0      literal ASB$C_BLN_IDX = 512;
1288  0      literal ASB$S_ASBDEF = 512;
1289  0      macro ASB$W_STKLEN = 0,0,16,0 %;              ! save stack length (must be first word in block)
1290  0      !    STKLEN = BLN org - BLN_FIX
1291  0      macro ASB$W_STKSIZ = 2,0,16,0 %;              ! size of saved stack in bytes
1292  0      macro ASB$B_BID = 8,0,8,0 %;                  ! block id
1293  0      macro ASB$B_BLN = 9,0,8,0 %;                  ! block length in longwords
1294  0      macro ASB$L_ARGLST = 12,0,0,0 %;
```

```
1295  0        literal ASB$S_ARGLST = 16;                        !  saved argument list on async irab operations
1296  0        macro ASB$B_ARGCNT = 12,0,8,0 %;                  !  argument count
1297  0        !  value will be 0, 1, 2, or 3
1298  0        macro ASB$L_FABRAB = 16,0,32,0 %;                 !  fab or rab address
1299  0        macro ASB$L_ERR = 20,0,32,0 %;                    !  err routine addr
1300  0        macro ASB$L_SUC = 24,0,32,0 %;                    !  suc routine addr
1301  0        macro ASB$L_REGS = 28,0,0,0 %;
1302  0        literal ASB$S_REGS = 20;                          !  save register area for regs 6, 7, 8, 10 and 11
1303  0        macro ASB$L_STK = 48,0,0,0 %;
1304  0        literal ASB$S_STK = 140;                          !  saved stack area
1305  0
1306  0        !*** MODULE $BDBDEF ***
1307  0
1308  0        !           BDB field definitions
1309  0        !
1310  0        !           buffer descriptor block (bdb)
1311  0        !
1312  0        !           there is one bdb per i/o buffer
1313  0        !           ( the i/o buffers exist in separate pages, page aligned)
1314  0        !
1315  0        literal BDB$C_BID = 12;                           !  bdb id code
1316  0        literal BDB$M_VAL = 1;
1317  0        literal BDB$M_DRT = 2;
1318  0        literal BDB$M_IOP = 4;
1319  0        literal BDB$M_PRM = 8;
1320  0        literal BDB$M_NOLOCATE = 16;
1321  0        literal BDB$M_WFO = 32;
1322  0        literal BDB$M_AST_DCL = 64;
1323  0        literal BDB$S_BDBDEF = 80;
1324  0        macro BDB$L_FLINK = 0,0,32,0 %;                   !  forward link
1325  0        macro BDB$L_BLINK = 4,0,32,0 %;                   !  backward link
1326  0        macro BDB$B_BID = 8,0,8,0 %;                      !  block id
1327  0        macro BDB$B_BLN = 9,0,8,0 %;                      !  block length in longwords
1328  0        macro BDB$B_FLGS = 10,0,8,0 %;                    !  bdb flags
1329  0        macro BDB$V_VAL = 10,0,1,0 %;                     !  buffer contents valid
1330  0        macro BDB$V_DRT = 10,1,1,0 %;                     !  buffer content dirty
1331  0        macro BDB$V_IOP = 10,2,1,0 %;                     !  buffer has i/o in progress
1332  0        macro BDB$V_PRM = 10,3,1,0 %;                     !  buffer has permanence factor
1333  0        macro BDB$V_NOLOCATE = 10,4,1,0 %;                !  buffer shared - no locate mode
1334  0        macro BDB$V_WFO = 10,5,1,0 %;                     !  other streams awaiting
1335  0        macro BDB$V_AST_DCL = 10,6,1,0 %;                 !  ast has been declared for
1336  0        !   waiting stream
1337  0        !   the releasing of this bdb
1338  0        !   (set/cleared by rm$cache)
1339  0        macro BDB$B_CACHE_VAL = 11,0,8,0 %;               !  relative value of buffer in cache
1340  0        macro BDB$B_VERTYP = 11,0,8,0 %;                  !  version type (1 = wild)
1341  0        macro BDB$W_USERS = 12,0,16,0 %;                  !  number of streams referencing this buffer
1342  0        macro BDB$W_BUFF_ID = 14,0,16,0 %;                !  buffer identification number
1343  0        macro BDB$L_BLB_PTR = 16,0,32,0 %;                !  pointer to BLB chain for this BDB
1344  0        macro BDB$W_NUMB = 20,0,16,0 %;                   !  of bytes of buffer in use
1345  0        macro BDB$W_DIRSEQ = 20,0,16,0 %;                 !  UCB$W_DIRSEQ at directory read time
1346  0        macro BDB$W_SIZE = 22,0,16,0 %;                   !  bytes in buffer
1347  0        macro BDB$L_ADDR = 24,0,32,0 %;                   !  address of buffer
1348  0        macro BDB$L_VBN = 28,0,32,0 %;                    !  1st vbn in buffer
1349  0        macro BDB$L_VBNSEQNO = 32,0,32,0 %;               !  vbn seq number of validity check vs. bcb copy
1350  0        macro BDB$L_LAST = 32,0,32,0 %;                   !  address of last directory record
1351  0        macro BDB$L_WAIT = 36,0,32,0 %;                   !  wait thread (irab addr)
```

```
1352   0       !  (for inter-stream intra-
1353   0       !   process locking only)
1354   0       macro BDB$L_VERCOUNT = 36,0,32,0 %;          ! negative count of version entries scanned
1355   0       macro BDB$L_ALLOC_ADDR = 40,0,32,0 %;        ! buffer allocation addr
1356   0       macro BDB$W_ALLOC_SIZE = 44,0,16,0 %;        ! buffer allocation size
1357   0       macro BDB$L_BI_BDB = 48,0,32,0 %;            ! address of isam/block i/o bi journaling BDB
1358   0       macro BDB$L_AI_BDB = 52,0,32,0 %;            ! address of isam/block i/o ai journaling BDB
1359   0       macro BDB$T_JNLSEQ = 56,0,0,0 %;
1360   0       literal BDB$S_JNLSEQ = 16;                   ! Journaling Sequence Number Block
1361   0       macro BDB$L_WR1 = 72,0,32,0 %;               ! work area
1362   0       macro BDB$B_REL_VBN = 72,0,8,0 %;            ! current vbn rel to start of buffer
1363   0       macro BDB$B_VAL_VBNS = 73,0,8,0 %;           ! ! of valid vbns in buffer
1364   0       macro BDB$B_PRE_CCTL = 74,0,8,0 %;           ! unit record carriage control byte ('pre')
1365   0       macro BDB$B_POST_CCTL = 75,0,8,0 %;          ! unit record carriage control byte ('post')
1366   0       macro BDB$L_CURBUFADR = 76,0,32,0 %;         ! current buffer addr
1367   0       literal BDB$K_BLN = 80;                      ! length of bdb block
1368   0       literal BDB$C_BLN = 80;                      ! length of bdb block
1369   0       literal BDB$S_BDBDEF1 = 80;
1370   0       macro BDB$L_IOSB = 72,0,0,0 %;
1371   0       literal BDB$S_IOSB = 8;                      ! i/o status block for buffer
1372   0       macro BDB$L_VERSION = 72,0,32,0 %;           ! addr of current/next directory version entry
1373   0       macro BDB$L_RECORD = 76,0,32,0 %;            ! address of current/next directory record
1374   0
1375   0       !*** MODULE $GBPBDEF ***
1376   0       !
1377   0       !          GBPB field definitions
1378   0       !
1379   0       !          Global Buffer Pointer Block (GBPB)
1380   0       !
1381   0       !          The GBPB is the process local structure used in conjunction with
1382   0       !          shared global i/o buffers.  In order to minimize the impact of
1383   0       !          global buffers on existing code, the GBPB is identical to a BDB
1384   0       !          in those fields which are referenced outside of the RM$CACHE and
1385   0       !          RM$RELEASE routines.
1386   0       !
1387   0       literal GBPB$C_BID = 21;                     ! gbpb id code
1388   0       literal GBPB$K_BLN = 40;                     ! Length of GBPB block
1389   0       literal GBPB$C_BLN = 40;                     ! Length of GBPB block
1390   0       literal GBPB$S_GBPBDEF = 40;
1391   0       macro GBPB$L_FLINK = 0,0,32,0 %;             ! forward link
1392   0       macro GBPB$L_BLINK = 4,0,32,0 %;             ! backward link
1393   0       macro GBPB$B_BID = 8,0,8,0 %;                ! block id
1394   0       macro GBPB$B_BLN = 9,0,8,0 %;                ! block length in longwords
1395   0       macro GBPB$B_FLGS = 10,0,8,0 %;              ! gbpb flags (use BDB flgs definitions)
1396   0       macro GBPB$B_CACHE_VL = 11,0,8,0 %;          ! relative cache value of this buffer
1397   0       macro GBPB$W_USERS = 12,0,16,0 %;            ! number of streams referencing this buffer
1398   0       macro GBPB$W_BUFF_ID = 14,0,16,0 %;          ! buffer identification number
1399   0       macro GBPB$L_BLB_PTR = 16,0,32,0 %;          ! pointer to BLB chain for this GBPB
1400   0       macro GBPB$W_NUMB = 20,0,16,0 %;             ! ! of bytes of buffer in use
1401   0       macro GBPB$W_SIZE = 22,0,16,0 %;             ! ! bytes in buffer
1402   0       macro GBPB$L_ADDR = 24,0,32,0 %;             ! address of buffer
1403   0       macro GBPB$L_VBN = 28,0,32,0 %;              ! 1st vbn in buffer
1404   0       macro GBPB$L_VBNSEQNO = 32,0,32,0 %;         ! sequence number field.
1405   0       macro GBPB$L_GBD_PTR = 36,0,32,0 %;          ! Pointer to the GBD for this buffer.
1406   0
1407   0       !*** MODULE $RLBDEF ***
1408   0       !
```

```
1409  0   !          RLB field definitions
1410  0   !
1411  0   !          record lock block (rlb)
1412  0   !
1413  0   !          The rlb describes one locked record for a particular
1414  0   !          process-record stream (rab/irab). if the owner field
1415  0   !          is 0 then the rlb is available for use. otherwise, it
1416  0   !          describes a locked record. note: when owner is 0 the
1417  0   !          record rfa fields are zeroed (0).
1418  0   !
1419  0   !
1420  0   !          +-------------------------------------------+
1421  0   !   rlb:   !                 link                      !
1422  0   !          +-------------------------+-----------------+
1423  0   !          !                         !     rfa4 id     !
1424  0   !          +-------+--------+--------+--------+--------+
1425  0   !          ! flags !reserved!  bln   !   bid  !
1426  0   !          +-------+--------+--------+--------+--------+
1427  0   !          !                 rfa0                     !
1428  0   !          +-------------------------------------------+
1429  0   !          !                 owner                     !
1430  0   !          +-------------------------------------------+
1431  0   !   lksb:  ! Still to be def- !  VMS status code      !
1432  0   !          ! ined status bits !                       !
1433  0   !          +-------------------------------------------+
1434  0   !          ! Lock Id. (Returned for new locks,         !
1435  0   !          !          input for conversions)           !
1436  0   !          +-------------------------------------------+
1437  0   !
1438  0   literal RLB$M_TIMER_INPROG = 1;
1439  0   literal RLB$C_BID = 14;                        ! rlb code
1440  0   literal RLB$M_WAIT = 1;
1441  0   literal RLB$M_CR = 2;
1442  0   literal RLB$M_PW = 4;
1443  0   literal RLB$M_PR = 8;
1444  0   literal RLB$M_CONV = 16;
1445  0   literal RLB$M_LV2 = 32;
1446  0   literal RLB$M_FAKE = 64;
1447  0   literal RLB$M_TMO = 128;
1448  0   literal RLB$K_BLN = 28;                        ! length of rlb
1449  0   literal RLB$C_BLN = 28;                        ! length of rlb
1450  0   literal RLB$S_RLBDEF = 28;
1451  0   macro RLB$L_LNK = 0,0,32,0 %;                  ! link to next rlb
1452  0   macro RLB$L_MISC = 4,0,32,0 %;                 ! longword definition to optimize clearing field
1453  0   macro RLB$W_FLAGS2 = 4,0,16,0 %;               ! more flag bits
1454  0   macro RLB$V_TIMER_INPROG = 4,0,1,0 %;          ! Timer queued.
1455  0   macro RLB$W_RFA4 = 6,0,16,0 %;                 ! 3'rd word of records rfa
1456  0   !   offset for seq f.o. (bits 0:14)
1457  0   !   always 0 for rel f.o. (bits 0:14)
1458  0   macro RLB$W_ID = 6,0,16,0 %;                   ! id for idx f.o.
1459  0   macro RLB$B_BID = 8,0,8,0 %;                   ! block id
1460  0   macro RLB$B_BLN = 9,0,8,0 %;                   ! block length in longwords
1461  0   macro RLB$B_TMO = 10,0,8,0 %;                  ! propagation of ROP TMO field
1462  0   macro RLB$B_FLAGS = 11,0,8,0 %;                ! various locking flags
1463  0   macro RLB$V_WAIT = 11,0,1,0 %;                 ! propagation of ROP WAT bit
1464  0   macro RLB$V_CR = 11,1,1,0 %;                   ! defines lock manager mode "concurrent read"
1465  0   macro RLB$V_PW = 11,2,1,0 %;                   ! allow reader access to locked record flag
```
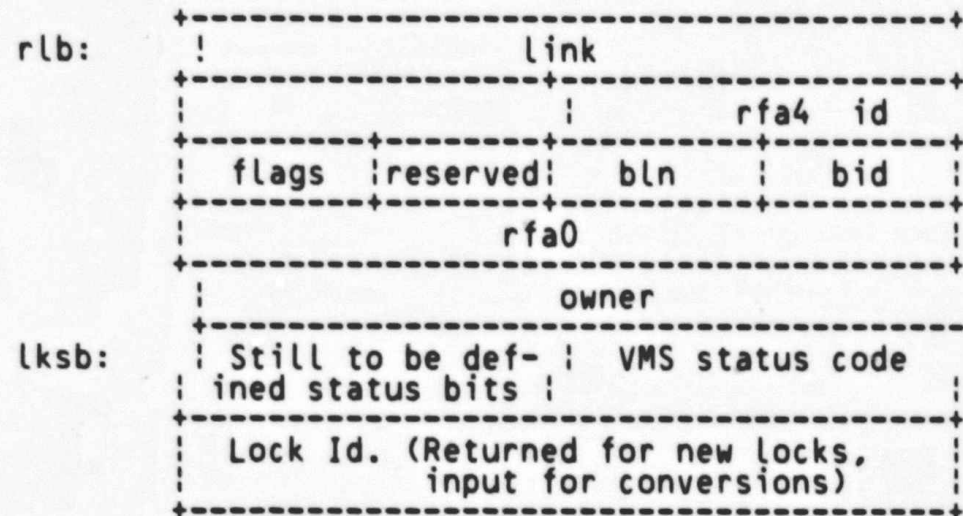
```
1466  0       macro RLB$V_PR = 11,3,1,0 %;                    ! used to query lock database
1467  0       macro RLB$V_CONV = 11,4,1,0 %;                  ! defines lock manager option "convert"
1468  0       macro RLB$V_LV2 = 11,5,1,0 %;                   ! sets lock as "level 2" RU consistancy
1469  0       macro RLB$V_FAKE = 11,6,1,0 %;                  ! this RLB contains no lock.
1470  0       macro RLB$V_TMO = 11,7,1,0 %;                   ! propagation of ROP TMO bit
1471  0       ! indicate "lock for write, allow readers"
1472  0       ! used to query lock database for records
1473  0       macro RLB$L_RFA0 = 12,0,32,0 %;                 ! 1'st and 2'nd words of record's rfa
1474  0       ! seq f.o. vbn
1475  0       ! rel f.o. relative record number
1476  0       ! idx f.o. start vbn
1477  0       macro RLB$L_OWNER = 16,0,32,0 %;                ! identification of owning stream
1478  0       macro RLB$L_LKSB = 20,0,32,0 %;                 ! first longword of lock status block
1479  0       macro RLB$W_STATUS = 20,0,16,0 %;               ! VMS status code
1480  0       macro RLB$W_S_BITS = 22,0,16,0 %;               ! various status bits
1481  0       macro RLB$L_LOCK_ID = 24,0,32,0 %;              ! second longword of lksb is lock_id
1482  0
1483  0       !*** MODULE $FLBDEF ***
1484  0       !
1485  0       !    file lock block definitions
1486  0       !
1487  0       literal FLB$C_BID = 23;
1488  0       literal FLB$K_BLN = 20;
1489  0       literal FLB$C_BLN = 20;
1490  0       literal FLB$S_FLBDEF = 20;
1491  0       macro FLB$L_FLB_LNK = 0,0,32,0 %;               ! pointer to next FLB
1492  0       macro FLB$L_RLB_LNK = 4,0,32,0 %;               ! pointer to RLBs
1493  0       macro FLB$B_BID = 8,0,8,0 %;                    ! block id
1494  0       macro FLB$B_BLN = 9,0,8,0 %;                    ! block length
1495  0       macro FLB$L_IFB_PTR = 12,0,32,0 %;              ! IFAB address
1496  0       macro FLB$L_LOCK_ID = 16,0,32,0 %;              ! lock id
1497  0
1498  0       !*** MODULE $DRCDEF ***
1499  0       !
1500  0       !    directory cache node definitions
1501  0       !
1502  0       literal DRC$K_BLN = 62;                         ! length of directory cache node
1503  0       literal DRC$C_BLN = 62;                         ! length of directory cache node
1504  0       literal DRC$S_DRCDEF = 62;
1505  0       macro DRC$L_NXTFLNK = 0,0,32,0 %;               ! link to next entry, this level
1506  0       macro DRC$L_NXTBLNK = 4,0,32,0 %;               ! link to previous entry, this level
1507  0       macro DRC$L_LVLFLNK = 8,0,32,0 %;               ! link to first entry, next lower level
1508  0       macro DRC$L_LVLBLNK = 12,0,32,0 %;              ! link to last entry, next lower level
1509  0       ! note: the links are maintained in lru order
1510  0       macro DRC$T_NAME = 16,0,0,0 %;
1511  0       literal DRC$S_NAME = 40;                        ! directory name or device and unit
1512  0       ! note: stored as counted string counting count itself
1513  0       macro DRC$W_DID = 56,0,0,0 %;
1514  0       literal DRC$S_DID = 6;                          ! file id for directory
1515  0       macro DRC$W_DIRSEQ = 58,0,16,0 %;               ! directory sequence ! for device node
1516  0
1517  0       !*** MODULE $RLSDEF ***
1518  0       !
1519  0       !                  release option flag definitions
1520  0       !
1521  0       literal RLS$M_RETURN = 1;
1522  0       literal RLS$M_WRT_THRU = 2;
```

```
1523  0         literal RLS$M_KEEP_LOCK = 4;
1524  0         literal RLS$M_DEQ = 8;
1525  0         literal RLS$S_RLSDEF = 1;
1526  0         macro RLS$V_RETURN = 0,0,1,0 %;            ! return buffer and bdb to free space lists
1527  0         macro RLS$V_WRT_THRU = 0,1,1,0 %;          ! write buffer if dirty
1528  0         macro RLS$V_KEEP_LOCK = 0,2,1,0 %;         ! keep bdb locked
1529  0         macro RLS$V_DEQ = 0,3,1,0 %;               ! always release lock
1530  0
1531  0         !*** MODULE $CSHDEF ***
1532  0         !
1533  0         !               cache option flag definitions
1534  0         !
1535  0         literal CSH$M_LOCK = 1;
1536  0         literal CSH$M_NOWAIT = 2;
1537  0         literal CSH$M_NOREAD = 4;
1538  0         literal CSH$M_NOBUFFER = 8;
1539  0         literal CSH$S_CSHDEF = 1;
1540  0         macro CSH$V_LOCK = 0,0,1,0 %;              ! obtain exclusive access to block
1541  0         macro CSH$V_NOWAIT = 0,1,1,0 %;            ! do not wait for block on access interlock
1542  0         macro CSH$V_NOREAD = 0,2,1,0 %;            ! do not read in block
1543  0         macro CSH$V_NOBUFFER = 0,3,1,0 %;          ! obtain access to block but don't allocate
1544  0         !   a buffer for it and don't read it
1545  0         !   collision
1546  0
1547  0         !*** MODULE $PIODEF ***
1548  0         !
1549  0         !
1550  0         !    rms overall status bit definitions
1551  0         !
1552  0         literal PIO$S_PIODEF = 1;
1553  0         macro PIO$V_INHAST = 0,0,1,0 %;            ! set if asts implicitly inhibited
1554  0         !   if reset by disabled ast, ast must be re-
1555  0         !   enabled
1556  0         macro PIO$V_EOD = 0,1,1,0 %;               ! set if searching for 'eod' string on 'input'
1557  0         macro PIO$V_SYNC1 = 0,2,1,0 %;             ! sync stalled operation using efn 27
1558  0         macro PIO$V_SYNC2 = 0,3,1,0 %;             ! sync stalled operation using efn 28
1559  0
1560  0         !*** MODULE $FTLDEF ***
1561  0         !
1562  0         !          definitions for rms debug failure codes
1563  0         !
1564  0         !
1565  0         !    the following codes are for temporary bug check tests, and are
1566  0         !    internal to rms.  all of the codes are negative, implying that they
1567  0         !    do not return to the caller, probably killing the process (if not
1568  0         !    the entire system).
1569  0         !
1570  0         literal FTL$_SETPRTFAIL = -1;              ! set protection system service failed (rm0bufmgr)
1571  0         literal FTL$_STKTOOBIG = -2;               ! stack too big for asb (rm0stall)
1572  0         literal FTL$_BADIFAB = -3;                 ! invalid ifab or irab (rm0fset,rm0conn,rm0rset,rm0prflnm)
1573  0         literal FTL$_GTCHNFAIL = -4;               ! get channel system service failure (rm0prflnm)
1574  0         literal FTL$_BADORGCASE = -5;              ! invalid orgcase value for dispatch (all rms$
1575  0         !   level routines execept open and create)
1576  0         literal FTL$_BADBDB = -6;                  ! block not a bdb (rm0bufmgr)
1577  0         literal FTL$_ASBALLFAIL = -7;              ! couldn't allocate an asb (rm0stall)
1578  0         literal FTL$_BADASTPRM = -8;               ! ast parameter not a valid ifab/irab addr (rm0stall)
1579  0         literal FTL$_CANTDOAST = -9;               ! couldn't redeclare ast (insf. mem.) (rm0stall)
```

```
1580   0          literal FTL$_NOSTRUCT = -10;              ! rab or fab not same on ast (rm0stall)
1581   0          literal FTL$_NOASB = -11;                 ! asb not allocated or stream not busy on ast (rm0stall)
1582   0          literal FTL$_NONXTBDB = -12;              ! no next bdb available (rm1seqxfr)
1583   0          literal FTL$_BADBUFSIZ = -13;             ! disk buffer size not = 512 (rm1conn)
1584   0          literal FTL$_ENQDEQFAIL = -14;            ! enq or deq service failed (rm0reclck)
1585   0          literal FTL$_NOCURBDB = -15;              ! no current bdb before calling rm$release (rm0reclck)
1586   0          literal FTL$_NOPARENT = -16;              ! no parent lock available for global buffer section lock (rm0share)
1587   0          literal FTL$_DEALLERR = -17;              ! ifab deallocation attempted with other block(s)
1588   0            still allocated (rms0close)
1589   0          literal FTL$_IORNDN = -18;                ! i/o rundown inconsistency (either ifab or irab
1590   0            table entries not zeroed) (rms0rndwn)
1591   0          literal FTL$_XFERSIZE = -19;              ! size of requested transfer not equal to
1592   0            or less than the current number of bytes
1593   0            in use for the bdb (rm0cache)
1594   0          literal FTL$_NOTLOCKED = -20;             ! bdb not locked and a keep lock request
1595   0            was made on a release request.
1596   0          literal FTL$_NODIDORFID = -21;            ! neither a fid nor a did was set upon exit from
1597   0            rm$setdid (rms0erase)
1598   0          literal FTL$_RELEASFAIL = -22;            ! release of non-dirty bdb failed (rm0xtnd23,rms0extend)
1599   0          literal FTL$_NOLOCKBDB = -23;             ! no lock bdb found (rm0xtnd23)
1600   0          literal FTL$_NONETWORK = -24;             ! network routine entered but no network support in rms
1601   0          literal FTL$_LOCKFAILED = -25;            ! failed to lock prolog (rm2create)
1602   0          literal FTL$_BADLEVEL = -26;              ! to search by id, structure level must be 0
1603   0          literal FTL$_ASTDECERR = -27;             ! ast declaration for file sharing failed
1604   0          literal FTL$_BADGBLCNT = -28;             ! Zero global buffer count found when not expected (rm1conn)
1605   0          literal FTL$_ACCNTOVFLO = -29;            ! access count overflow (rm0share)
1606   0          literal FTL$_BDBAVAIL = -30;              ! BDB was available and shouldn't have been.
1607   0          literal FTL$_GBLNOLK = -31;               ! Record locking was not set with global buffers.
1608   0          literal FTL$_LCKFND = -32;                ! A lock was found and we don't know what to do.
1609   0          literal FTL$_NOBLB = -33;                 ! No BLB was found and there should have been one.
1610   0          literal FTL$_NOGBPB = -34;                ! No GBPB was found and should have been.
1611   0          literal FTL$_NOLCLBUF = -35;              ! Should have found a local buffer.
1612   0          literal FTL$_NORDNOTSET = -36;            ! NOREAD not set when NOBUFFER was.
1613   0          literal FTL$_NOTGBPB = -37;               ! Found an illegit BDB.
1614   0          literal FTL$_NOSFSB = -38;                ! No SFSB when allocating BLB.
1615   0          literal FTL$_LOCKHELD = -39;              ! Attempted to return a BLB with lock_id neq 0
1616   0          literal FTL$_RLSDRT = -40;                ! Dirty buffer found in releaseall.
1617   0          literal FTL$_BADBLB = -41;                ! Bad BLB found in blocking AST routine.
1618   0          literal FTL$_BADOWNER = -42;              ! Owner field in BLB is bad in blocking AST routine.
1619   0          literal FTL$_GETLKIFAIL = -43;            ! $GETLKIW failed in last chance (rms0lstch).
1620   0          literal FTL$_BADEBKHBK = -44;             ! tried to store an invalid EBK/HBK (rm0share).
1621   0
1622   0          !*** MODULE $BUGDEF ***
1623   0          !
1624   0          !   the following internal codes are for non-fatal bug check reporting.
1625   0          !   these codes are positive byte values.  they trigger a reporting action
1626   0          !   and return to the caller with r0 set to rms$_bug+<8*the bug code>,
1627   0          !   which is an externally documented rms error code.
1628   0          !
1629   0          literal BUG$_BADDFLTDIR = 1;              ! DEFAULT DIRECTORY STRING INVALID (RM0XPFN)
1630   0
1631   0          !*** MODULE $IDXDEF ***
1632   0          !
1633   0          !           IDX field definitions
1634   0          !
1635   0          !           index descriptor definition
1636   0          !
```

```
1637  0   !           An index descriptor block exists for each key of reference in use.
1638  0   !           they are not necessarily contiguous in memory.
1639  0   !
1640  0   literal IDX$C_BID = 15;                     ! id for index descriptor block
1641  0   literal IDX$M_DUPKEYS = 1;
1642  0   literal IDX$M_CHGKEYS = 2;
1643  0   literal IDX$M_NULKEYS = 4;
1644  0   literal IDX$M_IDX_COMPR = 8;
1645  0   literal IDX$M_INITIDX = 16;
1646  0   literal IDX$M_KEY_COMPR = 64;
1647  0   literal IDX$M_REC_COMPR = 128;
1648  0   literal IDX$C_STRING = 0;                    ! string data type
1649  0   literal IDX$C_SGNWORD = 1;                   ! signed binary word
1650  0   literal IDX$C_UNSGNWORD = 2;                 ! unsigned binary word
1651  0   literal IDX$C_SGNLONG = 3;                   ! signed binary long word
1652  0   literal IDX$C_UNSGNLONG = 4;                 ! unsigned binary long word
1653  0   literal IDX$C_PACKED = 5;                    ! packed decimal
1654  0   literal IDX$C_SGNQUAD = 6;                   ! signed binary quadword
1655  0   literal IDX$C_UNSGNQUAD = 7;                 ! unsigned binary quadword
1656  0   literal IDX$C_V2_BKT = 0;                    ! Prologue two bucket
1657  0   literal IDX$C_CMPIDX = 1;                    ! Prologue 3, index keys are compressed
1658  0   literal IDX$C_NCMPIDX = 2;                   ! Prologue 3, index keys are not compressed
1659  0   literal IDX$C_CMPCMP = 3;                    ! Prologue 3, primary key is compressed, data
1660  0   !    is compressed
1661  0   !     Prologue 3, SIDR key is compressed
1662  0   literal IDX$C_CMPNCMP = 4;                   ! Prologue 3, primary key is compressed,
1663  0   !     data is not compressed
1664  0   literal IDX$C_NCMPCMP = 5;                   ! Prologue 3, primary key is not compressed
1665  0   !     data is compressed
1666  0   literal IDX$C_NCMPNCMP = 6;                  ! Prologue 3, primary key is not compressed
1667  0   !     data is not compressed
1668  0   !     Prologue 3, SIDR key is compressed
1669  0   literal IDX$K_FIXED_BLN = 44;
1670  0   literal IDX$C_FIXED_BLN = 44;
1671  0
1672  0   !   the following is the length of the fixed part of the index descriptor
1673  0   !
1674  0   !
1675  0   !   the following is repeated for each key segment
1676  0   !
1677  0   literal IDX$S_IDXDEF = 48;
1678  0   macro IDX$L_IDXFL = 0,0,32,0 %;              ! forward link to next index descriptor
1679  0   macro IDX$B_BID = 8,0,8,0 %;                 ! block id
1680  0   macro IDX$B_BLN = 9,0,8,0 %;                 ! length of block
1681  0   macro IDX$L_VBN = 10,0,32,0 %;               ! VBN where the descriptor came from
1682  0   macro IDX$W_OFFSET = 14,0,16,0 %;            ! Offset into the block (VBN) of the descriptor
1683  0   macro IDX$B_DESC_NO = 16,0,8,0 %;            ! Descriptor number (index into update buffer)
1684  0   macro IDX$B_IANUM = 18,0,8,0 %;              ! area number for index buckets
1685  0   macro IDX$B_LANUM = 19,0,8,0 %;              ! area number for lower index buckets
1686  0   macro IDX$B_DANUM = 20,0,8,0 %;              ! area number for data buckets
1687  0   macro IDX$B_ROOTLEV = 21,0,8,0 %;            ! level of root
1688  0   macro IDX$B_IDXBKTSZ = 22,0,8,0 %;           ! size of index bucket in vbn's
1689  0   macro IDX$B_DATBKTSZ = 23,0,8,0 %;           ! size of data bucket in vbn's
1690  0   macro IDX$L_ROOTVBN = 24,0,32,0 %;           ! start vbn of root bucket
1691  0   macro IDX$B_FLAGS = 28,0,8,0 %;              ! index/key flags
1692  0   macro IDX$V_DUPKEYS = 28,0,1,0 %;            ! duplicate keys allowed
1693  0   macro IDX$V_CHGKEYS = 28,1,1,0 %;            ! keys can change values
```

```
1694  0        macro IDX$V_NULKEYS = 28,2,1,0 %;          ! null key value allowed
1695  0        macro IDX$V_IDX_COMPR = 28,3,1,0 %;        ! index is compressed
1696  0        macro IDX$V_INITIDX = 28,4,1,0 %;          ! index is not initialized
1697  0        macro IDX$V_KEY_COMPR = 28,6,1,0 %;        ! key has been compressed
1698  0        macro IDX$V_REC_COMPR = 28,7,1,0 %;        ! data record is in compressed form
1699  0        macro IDX$B_DATATYPE = 29,0,8,0 %;         ! data type of key field
1700  0        macro IDX$B_SEGMENTS = 30,0,8,0 %;         ! number of key field segments
1701  0        macro IDX$B_NULLCHAR = 31,0,8,0 %;         ! null character
1702  0        macro IDX$B_KEYSZ = 32,0,8,0 %;            ! total key size
1703  0        macro IDX$B_KEYREF = 33,0,8,0 %;           ! key of reference(0-primary)
1704  0        macro IDX$W_MINRECSZ = 34,0,16,0 %;        ! minimum record size
1705  0        macro IDX$W_IDXFILL = 36,0,16,0 %;         ! index fill
1706  0        macro IDX$W_DATFILL = 38,0,16,0 %;         ! data fill
1707  0        macro IDX$B_IDXBKTYP = 40,0,8,0 %;         ! PLG3 - type of index bucket and SIDR bucket
1708  0        macro IDX$B_DATBKTYP = 41,0,8,0 %;         ! PLG3 - type of primary data bucket
1709  0        macro IDX$W_POSITION = 44,0,16,0 %;        ! key segment position
1710  0        macro IDX$B_SIZE = 46,0,8,0 %;             ! key segment size (plg 3)
1711  0        macro IDX$B_TYPE = 47,0,8,0 %;             ! key segment datatype (plg 3)
1712  0
1713  0        !*** MODULE $UPDDEF ***
1714  0        !
1715  0        !   update buffer flags
1716  0        !
1717  0        literal UPD$M_INS_NEW = 1;
1718  0        literal UPD$M_OLD_DEL = 2;
1719  0        literal UPD$S_UPDDEF = 1;
1720  0        macro UPD$B_FLAGS = 0,0,8,0 %;
1721  0        macro UPD$V_INS_NEW = 0,0,1,0 %;           ! alternate key to be inserted from record buffer
1722  0        macro UPD$V_OLD_DEL = 0,1,1,0 %;           ! delete this key value using old record
1723
1724  0        !*** MODULE $GBHDEF ***
1725  0        !
1726  0        !        GBH field definitions
1727  0        !
1728  0        !        Global Buffer Header (GBH)
1729  0        !
1730  0        !        There is a Global Buffer Header for every file's global buffer section.
1731  0        !
1732  0        !        ***  WARNING - THIS STRUCTURE MUST BE QUADWORD ALIGNED  ***
1733  0        !
1734  0        literal GBH$C_BID = 17;                    ! Block ID code for GBH
1735  0        literal GBH$M_CACHE_IN = 1;
1736  0        literal GBH$M_CACHE_OUT = 2;
1737  0        literal GBH$M_RLS_IN = 4;
1738  0        literal GBH$M_RLS_OUT = 8;
1739  0        literal GBH$M_QIO_START = 16;
1740  0        literal GBH$M_QIO_DONE = 32;
1741  0        literal GBH$M_STALL = 64;
1742  0        literal GBH$M_THREADGO = 128;
1743  0        literal GBH$M_BLB_ENQ = 256;
1744  0        literal GBH$M_BLB_GRANT = 512;
1745  0        literal GBH$M_BLB_DEQ = 1024;
1746  0        literal GBH$M_BLB_BLOCK = 2048;
1747  0        literal GBH$M_F1 = 4096;
1748  0        literal GBH$M_F2 = 8192;
1749  0        literal GBH$M_F3 = 16384;
1750  0        literal GBH$M_F4 = 32768;
```

```
1751  0          literal GBH$K_BLN = 88;                        ! Length of global buffer header structure
1752  0          literal GBH$C_BLN = 88;                        ! Length of global buffer header structure
1753  0          literal GBH$S_GBHDEF = 88;
1754  0          macro GBH$L_GBD_FLNK = 0,0,32,0 %;             ! Self relative queue header for GBD's
1755  0          macro GBH$L_GBD_BLNK = 4,0,32,0 %;
1756  0          macro GBH$B_BID = 8,0,8,0 %;                   ! Block ID
1757  0          macro GBH$B_BLN = 9,0,8,0 %;                   ! Length of GBH in longwords
1758  0          macro GBH$W_TRC_FLGS = 10,0,16,0 %;            ! Trace flags (set to trace given function)
1759  0          macro GBH$V_CACHE_IN = 10,0,1,0 %;             ! Cache inputs
1760  0          macro GBH$V_CACHE_OUT = 10,1,1,0 %;            ! Cache outputs
1761  0          macro GBH$V_RLS_IN = 10,2,1,0 %;               ! Release inputs
1762  0          macro GBH$V_RLS_OUT = 10,3,1,0 %;              ! Release outputs
1763  0          macro GBH$V_QIO_START = 10,4,1,0 %;            ! Qio inputs
1764  0          macro GBH$V_QIO_DONE = 10,5,1,0 %;             ! Qio outputs
1765  0          macro GBH$V_STALL = 10,6,1,0 %;                ! Stall inputs
1766  0          macro GBH$V_THREADGO = 10,7,1,0 %;             ! Stall outputs
1767  0          macro GBH$V_BLB_ENQ = 10,8,1,0 %;              ! Bucket lock ENQ inputs
1768  0          macro GBH$V_BLB_GRANT = 10,9,1,0 %;            ! Bucket lock grant status
1769  0          macro GBH$V_BLB_DEQ = 10,10,1,0 %;             ! Bucket lock DEQ request
1770  0          macro GBH$V_BLB_BLOCK = 10,11,1,0 %;           ! Blocking AST received
1771  0          macro GBH$V_F1 = 10,12,1,0 %;
1772  0          macro GBH$V_F2 = 10,13,1,0 %;
1773  0          macro GBH$V_F3 = 10,14,1,0 %;
1774  0          macro GBH$V_F4 = 10,15,1,0 %;
1775  0          macro GBH$L_HI_VBN = 12,0,32,0 %;              ! Highest possible VBN value (FFFFFFFF).
1776  0          macro GBH$L_GS_SIZE = 16,0,32,0 %;             ! Size of total section in bytes.
1777  0          macro GBH$L_LOCK_ID = 20,0,32,0 %;             ! Lock ID of system file lock.
1778  0          macro GBH$L_GS_LOCK_ID = 24,0,32,0 %;          ! Lock ID of system global section lock.
1779  0          macro GBH$L_USECNT = 28,0,32,0 %;              ! Accessor count for section.
1780  0          macro GBH$L_TRC_FLNK = 32,0,32,0 %;            ! Trace blocks forward link
1781  0          macro GBH$L_TRC_BLNK = 36,0,32,0 %;            ! Trace blocks back link
1782  0          macro GBH$L_GBD_START = 40,0,32,0 %;           ! Offset to first GBD.
1783  0          macro GBH$L_GBD_END = 44,0,32,0 %;             ! Offset to last GBD.
1784  0          macro GBH$L_GBD_NEXT = 48,0,32,0 %;            ! Offset to next cache victim GBD.
1785  0          macro GBH$L_SCAN_NUM = 52,0,32,0 %;            ! Number of GBD's to scan for victim.
1786  0          !
1787  0          !   Global buffer statistics section
1788  0          !
1789  0          macro GBH$L_HIT = 56,0,32,0 %;                 ! Buffer found in global cache
1790  0          macro GBH$L_MISS = 60,0,32,0 %;                ! Buffer not found in global cache
1791  0          macro GBH$L_READ = 64,0,32,0 %;                ! Buffer read from disk into cache
1792  0          macro GBH$L_WRITE = 68,0,32,0 %;               ! Buffer written from cache to disk
1793  0          macro GBH$L_DFW_WRITE = 72,0,32,0 %;           ! Deferred writeback from cache to disk
1794  0          macro GBH$L_CROSS_HIT = 76,0,32,0 %;           ! Cross process hit count.
1795  0          macro GBH$L_OUTBUFQUO = 80,0,32,0 %;           ! Count of times GBLBUFQUO limit was hit.
1796  0          macro GBH$L_FILL_1 = 84,0,32,0 %;              ! Force quadword alignment
1797  0
1798  0          !*** MODULE $TRCDEF ***
1799  0          !
1800  0          !        TRC field definitions
1801  0          !
1802  0          !        Trace block structure (TRC)
1803  0          !
1804  0          !        Tracing saves at specific points in the RMS code for debugging and
1805  0          !        algorithm analysis purposes.
1806  0          !
1807  0          !        ***  WARNING - THIS STRUCTURE MUST BE QUADWORD ALIGNED  ***
```

```
1808  0          !
1809  0          literal TRC$C_BID = 18;                        ! Trace block code
1810  0          literal TRC$K_BLN = 64;                        ! NOTE: should be quadwords multiple to
1811  0          literal TRC$C_BLN = 64;                        ! NOTE: should be quadwords multiple to
1812  0          literal TRC$S_TRCDEF = 64;
1813  0          macro TRC$L_FLNK = 0,0,32,0 %;                 ! Trace block forward link
1814  0          macro TRC$L_BLNK = 4,0,32,0 %;                 ! Trace block back link
1815  0          macro TRC$B_BID = 8,0,8,0 %;                   ! Block ID
1816  0          macro TRC$B_BLN = 9,0,8,0 %;                   ! Length of block in longwords
1817  0          macro TRC$W_FUNCTION = 10,0,16,0 %;            ! Function code (see GBH definitions)
1818  0          macro TRC$L_STRUCTURE = 12,0,32,0 %;           ! Ifab/irab address.
1819  0          macro TRC$W_PID = 16,0,16,0 %;                 ! Process ID
1820  0          macro TRC$W_SEQNUM = 18,0,16,0 %;              ! Sequence number.
1821  0          macro TRC$L_VBN = 20,0,32,0 %;                 ! VBN requested.
1822  0          macro TRC$L_RETURN1 = 24,0,32,0 %;             ! Address of caller.
1823  0          macro TRC$L_RETURN2 = 28,0,32,0 %;             ! Caller's caller.
1824  0          macro TRC$L_ARGS = 32,0,0,0 %;
1825  0          literal TRC$S_ARGS = 32;                       ! Function specific arguments
1826  0          macro TRC$L_ARG_FLG = 32,0,32,0 %;             ! Argument flags (R3).
1827  0          macro TRC$L_BDB_ADDR = 36,0,32,0 %;            ! BDB address.
1828  0          macro TRC$W_BDB_USERS = 40,0,16,0 %;           ! Use count from BDB.
1829  0          macro TRC$W_BDB_BUFF = 42,0,16,0 %;            ! BDB buffer ID.
1830  0          macro TRC$B_BDB_CACHE = 44,0,8,0 %;            ! BDB cache value.
1831  0          macro TRC$B_BDB_FLAGS = 45,0,8,0 %;            ! Status flags from BDB.
1832  0          macro TRC$L_BDB_SEQ = 46,0,32,0 %;             ! Sequence number from BDB.
1833  0          macro TRC$B_BLB_MODE = 50,0,8,0 %;             ! Mode held in BLB.
1834  0          macro TRC$B_BLB_FLAGS = 51,0,8,0 %;            ! Flags from BLB.
1835  0          macro TRC$L_BLB_ADDR = 52,0,32,0 %;            ! Address of BLB.
1836  0          macro TRC$L_BLB_LOCK = 56,0,32,0 %;            ! Lock ID from BLB.
1837  0          macro TRC$L_BLB_SEQ = 60,0,32,0 %;             ! Sequence number from BLB.
1838  0          !   maintain quad alignment on header
1839  0
1840  0          !*** MODULE $GBDDEF ***
1841  0          !
1842  0          !          GBD structure definitions
1843  0          !
1844  0          !          Global Buffer Descriptor (GBD)
1845  0          !
1846  0          !          There is a single GBD for every buffer in a global buffer
1847  0          !          section (used only with shared files).  The GBD's themselves
1848  0          !          are in the section also and linked from a queue header in
1849  0          !          the Global Buffer Header (GBH).
1850  0          !
1851  0          !          ***  WARNING - THIS STRUCTURE MUST BE QUADWORD ALIGNED  ***
1852  0          !
1853  0          literal GBD$C_BID = 19;                        ! Block ID code for GBD
1854  0          literal GBD$M_VALID = 1;
1855  0          literal GBD$K_BLN = 40;                        ! Length of Global Buffer Descriptor structure.
1856  0          literal GBD$C_BLN = 40;                        ! Length of Global Buffer Descriptor structure.
1857  0          literal GBD$S_GBDDEF = 40;
1858  0          macro GBD$L_FLINK = 0,0,32,0 %;                ! Forward link - Note: This is a self relative queue
1859  0          macro GBD$L_BLINK = 4,0,32,0 %;                ! Back link
1860  0          macro GBD$B_BID = 8,0,8,0 %;                   ! Block ID
1861  0          macro GBD$B_BLN = 9,0,8,0 %;                   ! Block length of GBD
1862  0          macro GBD$B_FLAGS = 10,0,8,0 %;                ! Buffer status flags
1863  0          macro GBD$V_VALID = 10,0,1,0 %;                ! Buffer is valid.
1864  0          macro GBD$B_CACHE_VAL = 11,0,8,0 %;            ! Cache value of this bucket
```

```
1865   0        macro GBD$L_VBN = 12,0,32,0 %;              ! VBN of bucket the buffer describes
1866   0        macro GBD$L_VBNSEQNUM = 16,0,32,0 %;        ! VBN sequence number validity check
1867   0        macro GBD$L_LOCK_ID = 20,0,32,0 %;          ! Lock ID of system lock.
1868   0        macro GBD$W_NUMB = 24,0,16,0 %;             ! Number of bytes in use
1869   0        macro GBD$W_SIZE = 26,0,16,0 %;             ! Size of buffer in bytes
1870   0        macro GBD$L_REL_ADDR = 28,0,32,0 %;         ! Address of buffer relative to GBH
1871   0        macro GBD$W_USECNT = 32,0,16,0 %;           ! Accessor count for bucket
1872   0        macro GBD$B_REHIT_RD = 34,0,8,0 %;          ! Rehit by same process count.
1873   0        macro GBD$B_REHIT_LK = 35,0,8,0 %;          ! Rehit by same locker process.
1874   0
1875   0        !*** MODULE $BLBDEF ***
1876   0
1877   0        !         BLB field definitions
1878   0        !
1879   0        !         Bucket Lock Block (BLB)
1880   0        !
1881   0        !         The BLB contains the argument list for the SYS$ENQ system service
1882   0        !         as well a pointer to the BDB it relates to and other status.
1883   0        !
1884   0        literal BLB$C_BID = 16;                     ! BLB code
1885   0        literal BLB$M_LOCK = 1;
1886   0        literal BLB$M_NOWAIT = 2;
1887   0        literal BLB$M_NOREAD = 4;
1888   0        literal BLB$M_NOBUFFER = 8;
1889   0        literal BLB$M_IOLOCK = 16;
1890   0        literal BLB$M_DFW = 32;
1891   0        literal BLB$M_WRITEBACK = 64;
1892   0        literal BLB$K_BLN = 56;                     ! Length of BLB
1893   0        literal BLB$C_BLN = 56;                     ! Length of BLB
1894   0        literal BLB$S_BLBDEF = 56;
1895   0        macro BLB$L_FLNK = 0,0,32,0 %;              ! Link to next BLB
1896   0        macro BLB$L_BLNK = 4,0,32,0 %;              ! Back link
1897   0        macro BLB$B_BID = 8,0,8,0 %;                ! Block ID
1898   0        macro BLB$B_BLN = 9,0,8,0 %;                ! Block length
1899   0        macro BLB$B_BLBFLGS = 10,0,8,0 %;           ! Control flags for BLB
1900   0        macro BLB$V_LOCK = 10,0,1,0 %;              ! Corresponds to CSH$V_LOCK
1901   0        macro BLB$V_NOWAIT = 10,1,1,0 %;            ! Same as CSH$V_NOWAIT
1902   0        macro BLB$V_NOREAD = 10,2,1,0 %;            ! Same as CSH$V_NOREAD
1903   0        macro BLB$V_NOBUFFER = 10,3,1,0 %;          ! Same as CSH$V_NOBUFFER
1904   0        macro BLB$V_IOLOCK = 10,4,1,0 %;            ! Lock mode for read/write
1905   0        macro BLB$V_DFW = 10,5,1,0 %;               ! This is lock for deferred write buffer
1906   0        macro BLB$V_WRITEBACK = 10,6,1,0 %;         ! The associated buffer must be written back
1907   0        macro BLB$B_MODEHELD = 11,0,8,0 %;          ! Mode of current lock held.
1908   0        macro BLB$L_BDB_ADDR = 12,0,32,0 %;         ! BDB for which this lock is held
1909   0        macro BLB$L_OWNER = 16,0,32,0 %;            ! Address of stream owning this lock
1910   0        macro BLB$L_VBN = 20,0,32,0 %;              ! VBN of bucket lock (resource name)
1911   0        macro BLB$L_RESDSC = 24,0,0,0 %;
1912   0        literal BLB$S_RESDSC = 8;                   ! Resource name descriptor
1913   0        macro BLB$W_LKSTS = 32,0,16,0 %;            ! Lock status word
1914   0        macro BLB$L_LOCK_ID = 36,0,32,0 %;          ! Lock ID
1915   0        macro BLB$L_VALBLK = 40,0,0,0 %;
1916   0        literal BLB$S_VALBLK = 16;                  ! Lock value block
1917   0        macro BLB$L_VALSEQNO = 40,0,32,0 %;         ! Sequence number part of value block
1918   0
1919   0        !*** MODULE $RJBDEF ***
1920   0
1921   0        !         RJB Definitions
```

```
1922  0    !
1923  0    !            RMS Journaling Block (RJB)
1924  0    !
1925  0    !            This block contains the necessary control information to keep
1926  0    !            track of the state of journaling on this file
1927  0    !
1928  0    literal RJB$C_BID = 22;
1929  0    literal RJB$K_BLN = 12;                    ! Length of RJB
1930  0    literal RJB$C_BLN = 12;                    ! Length of RJB
1931  0    literal RJB$M_RU = 1;
1932  0    literal RJB$M_BI = 2;
1933  0    literal RJB$M_AI = 4;
1934  0    literal RJB$M_AT = 8;
1935  0    literal RJB$M_OPEN = 16;
1936  0    literal RJB$S_RJBDEF = 12;
1937  0    macro RJB$Q_CHAN = 0,0,0,0 %;
1938  0    literal RJB$S_CHAN = 8;                    ! Channel Block
1939  0    macro RJB$W_RUCHAN = 0,0,16,0 %;           !  channel for recovery unit journal
1940  0    macro RJB$W_BICHAN = 2,0,16,0 %;           !  channel for before image journal
1941  0    macro RJB$W_AICHAN = 4,0,16,0 %;           !  channel for after image journal
1942  0    macro RJB$W_ATCHAN = 6,0,16,0 %;           !  channel for audit trail journal
1943  0    macro RJB$B_BID = 8,0,8,0 %;               ! Block Id
1944  0    macro RJB$B_BLN = 9,0,8,0 %;               ! Block Length
1945  0    macro RJB$W_FLAGS = 10,0,16,0 %;           ! Flags word
1946  0    macro RJB$V_RU = 10,0,1,0 %;               ! Set to indicate RU channel open
1947  0    macro RJB$V_BI = 10,1,1,0 %;               ! Set to indicate BI channel open
1948  0    macro RJB$V_AI = 10,2,1,0 %;               ! Set to indicate AI channel open
1949  0    macro RJB$V_AT = 10,3,1,0 %;               ! Set to indicate AT channel open
1950  0    macro RJB$V_OPEN = 10,4,1,0 %;             ! Indicates $OPEN mapping entry written
1951  0
1952  0    !*** MODULE $MJBDEF ***
1953  0    !
1954  0    !            MJB field definitions
1955  0    !
1956  0    !              Miscellaneous Journaling Buffer
1957  0    !
1958  0    !            The MJB is used for writing miscellaneous journal entries,
1959  0    !            for example, extend entries or audit-trail entries.
1960  0    !
1961  0    literal MJB$C_BID = 24;
1962  0    literal MJB$M_INIT = 1;
1963  0    literal MJB$M_FORCE = 2;
1964  0    literal MJB$M_FILE = 4;
1965  0    literal MJB$M_SYNCH_SHARE = 8;
1966  0    literal MJB$K_BLN = 32;
1967  0    literal MJB$C_BLN = 32;
1968  0    literal MJB$S_MJBDEF = 32;
1969  0    macro MJB$B_BID = 8,0,8,0 %;               ! block id
1970  0    macro MJB$B_BLN = 9,0,8,0 %;               ! block length in longwords
1971  0    macro MJB$W_FLAGS = 10,0,16,0 %;           ! flags
1972  0    macro MJB$V_INIT = 10,0,1,0 %;             ! set if RJR overhead is initialized
1973  0    macro MJB$V_FORCE = 10,1,1,0 %;            ! set if RJR is to be written thru to journal
1974  0    macro MJB$V_FILE = 10,2,1,0 %;             ! set if file operation to journal
1975  0    macro MJB$V_SYNCH_SHARE = 10,3,1,0 %;      ! set if file lock can't be released during
1976  0    !    STALL
1977  0    !    and not buffered by CJF (input to WRITE_MJB)
1978  0    macro MJB$B_JNL = 12,0,8,0 %;              ! set to CJF$_"jnl type" as input to WRITE_MJB
```

```
1979  0      macro MJB$Q_DESC = 16,0,0,0 %;
1980  0      literal MJB$S_DESC = 8;                  ! RJR descriptor used in $WRITEJNL service
1981  0      macro MJB$W_SIZE = 16,0,16,0 %;          ! size of RJR to write
1982  0      macro MJB$L_POINTER = 20,0,32,0 %;       ! pointer to RJR
1983  0      macro MJB$Q_IOSB = 24,0,0,0 %;
1984  0      literal MJB$S_IOSB = 8;                  ! IOSB to use in $WRITEJNL
1985  0      macro MJB$T_RJR = 32,0,0,0 %;            ! the journal record begins here
1986  0      !
1987  0      ! ****************************************************************
1988  0      ! *                                                              *
1989  0      ! *    Copyright (c) 1982, 1983                                  *
1990  0      ! *    by DIGITAL Equipment Corporation, Maynard, Mass.          *
1991  0      ! *                                                              *
1992  0      ! *    This software is furnished under a license and may be used and  copied  *
1993  0      ! *    only  in  accordance  with  the  terms  of  such  license and with the  *
1994  0      ! *    inclusion of the above copyright notice.  This software or  any  other   *
1995  0      ! *    copies  thereof may not be provided or otherwise made available to any   *
1996  0      ! *    other person.  No title to and ownership of  the  software  is  hereby   *
1997  0      ! *    transferred.                                              *
1998  0      ! *                                                              *
1999  0      ! *    The information in this software is subject to change  without  notice  *
2000  0      ! *    and  should  not  be  construed  as  a commitment by DIGITAL Equipment  *
2001  0      ! *    Corporation.                                              *
2002  0      ! *                                                              *
2003  0      ! *    DIGITAL assumes no responsibility for the use or  reliability  of  its  *
2004  0      ! *    software on equipment which is not supplied by DIGITAL.   *
2005  0      ! *                                                              *
2006  0      ! ****************************************************************
2007  0      ! ***************************************************************************
2008  0      !    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
2009  0      ! ***************************************************************************
2010  0
2011  0
2012  0      !*** MODULE $FWADEF ***
2013  0      ! ++
2014  0      !
2015  0      !  Flags
2016  0      !
2017  0      ! --
2018  0      literal FWA$M_DUPOK = 1;
2019  0      literal FWA$M_NOCOPY = 2;
2020  0      literal FWA$M_SL_PASS = 4;
2021  0      literal FWA$M_RLF_PASS = 8;
2022  0      literal FWA$M_FNA_PASS = 16;
2023  0      literal FWA$M_NAM_DVI = 32;
2024  0      literal FWA$M_EXP_NODE = 64;
2025  0      literal FWA$M_VERSION = 2048;
2026  0      literal FWA$M_TYPE = 4096;
2027  0      literal FWA$M_NAME = 8192;
2028  0      literal FWA$M_DIR = 16384;
2029  0      literal FWA$M_DEVICE = 32768;
2030  0      literal FWA$M_EXP_VER = 65536;
2031  0      literal FWA$M_EXP_TYPE = 131072;
2032  0      literal FWA$M_EXP_NAME = 262144;
2033  0      literal FWA$M_WC_VER = 524288;
2034  0      literal FWA$M_WC_TYPE = 1048576;
2035  0      literal FWA$M_WC_NAME = 2097152;
```

```
2036  0        literal FWA$M_EXP_DIR = 4194304;
2037  0        literal FWA$M_EXP_DEV = 8388608;
2038  0        literal FWA$M_WILDCARD = 16777216;
2039  0        literal FWA$M_NODE = 33554432;
2040  0        literal FWA$M_QUOTED = 67108864;
2041  0        literal FWA$M_GRPMBR = 134217728;
2042  0        literal FWA$M_WILD_DIR = 268435456;
2043  0        literal FWA$M_DIR_LVLS = -536870912;
2044  0        literal FWA$C_ALL = 248;
2045  0
2046  0        !  constant for all flags that vary per parsing pass
2047  0        !
2048  0        literal FWA$C_ALLPASS = 25;
2049  0        !++
2050  0        !
2051  0        !  Misc. Fields
2052  0        !
2053  0        !--
2054  0        literal FWA$C_BID = 40;                    ! bid of fwa
2055  0        literal FWA$C_MAXNODNAM = 6;               ! max node name size
2056  0        literal FWA$C_MAXLNDNAM = 15;              ! max logical node name size
2057  0        literal FWA$C_MAXNODLST = 127;             ! max node spec list size (concatenated node specs)
2058  0        !
2059  0        !        device name descriptor
2060  0        !
2061  0        literal FWA$C_MAXDEVICE = 255;             ! max device name size
2062  0        literal FWA$C_MAXCDIR = 8;                 ! max number of concealed directories
2063  0        literal FWA$C_MAXSUBDIR = 7;               ! max number of sub directories
2064  0        literal FWA$C_MAXDIRLEN = 255;             ! max size of total directory spec
2065  0        !        should be:  top + subdir   39 * 8
2066  0        !                    dots between        7
2067  0        !                    delimiters          2
2068  0        !                                 ----------------
2069  0        !                    total             321
2070  0        !
2071  0        !    The filename, filetype and fileversion descriptors MUST be contiguous
2072  0        !
2073  0        !        file name descriptor
2074  0        !
2075  0        literal FWA$C_MAXNAME = 39;                ! max file name size
2076  0        literal FWA$C_MAXQUOTED = 255;             ! max quoted string size
2077  0        !
2078  0        !        file type descriptor
2079  0        !
2080  0        literal FWA$C_MAXTYPE = 39;                ! max file type size
2081  0        !
2082  0        !        file version number descriptor
2083  0        !
2084  0        literal FWA$C_MAXVER = 6;                  ! maximum version
2085  0        literal FWA$C_MAXRNS = 86;                 ! max resultant name string size
2086  0        literal FWA$C_STATBLK = 10;                ! define length of statistics block
2087  0        literal FWA$K_BLN_FWA = 500;               ! length of fwa
2088  0        literal FWA$C_BLN_FWA = 500;               ! length of fwa
2089  0        literal FWA$C_MAXSUBNOD = 7;               ! max number of secondary (sub) node specs
2090  0        !++
2091  0        !
2092  0        !    buffers for parsed filename elements
```

```
2093  0        !
2094  0        !  --
2095  0        literal FWA$C_FIBLEN = 76;              ! fib buffer size
2096  0        literal FWA$C_DIRBUFSIZ = 39;           ! size of each directory buffer
2097  0        !
2098  0        !    rooted directory name buffers
2099  0        !
2100  0        !    NOTE:  These buffers must be contiguous
2101  0        !
2102  0        literal FWA$K_BLN_BUF = 2364;           ! length of fwa and buffers
2103  0        literal FWA$C_BLN_BUF = 2364;           ! length of fwa and buffers
2104  0        literal FWA$K_BLN = 2364;               ! length of fwa and buffers
2105  0        literal FWA$C_BLN = 2364;               ! length of fwa and buffers
2106  0        literal FWA$S_FWADEF = 2364;
2107  0        macro FWA$Q_FLAGS = 0,0,0,0 %;
2108  0        literal FWA$S_FLAGS = 8;                ! various parse status flags
2109  0        macro FWA$B_PASSFLGS = 0,0,8,0 %;       ! flags for pass only
2110  0        macro FWA$B_FLDFLGS = 1,0,8,0 %;        ! flags for fields seen
2111  0        macro FWA$B_WILDFLGS = 2,0,8,0 %;       ! flags for wild cards
2112  0        macro FWA$B_PARSEFLGS = 3,0,8,0 %;      ! flags for parse results
2113  0        macro FWA$B_DIRFLGS = 4,0,8,0 %;        ! flags primarily for directory spec
2114  0        macro FWA$B_DIRWCFLGS = 5,0,8,0 %;      ! directory wild flags
2115  0        macro FWA$B_LNFLGS = 6,0,8,0 %;         ! logical name flag byte
2116  0        macro FWA$B_SLFLGS = 7,0,8,0 %;         ! search list + rooted directory flags
2117  0        !
2118  0        !    flags for pass
2119  0        !
2120  0        macro FWA$V_DUPOK = 0,0,1,0 %;          ! discard duplicate element
2121  0        macro FWA$V_NOCOPY = 0,1,1,0 %;         ! do not copy this field
2122  0        macro FWA$V_SL_PASS = 0,2,1,0 %;        ! search list pass
2123  0        macro FWA$V_RLF_PASS = 0,3,1,0 %;       ! set if applying related file defaults
2124  0        macro FWA$V_FNA_PASS = 0,4,1,0 %;       ! set if primary name string parse pass
2125  0        macro FWA$V_NAM_DVI = 0,5,1,0 %;        ! set if open by name block
2126  0        macro FWA$V_EXP_NODE = 0,6,1,0 %;       ! explicit node has been seen, null or normal
2127  0        !
2128  0        !    flags for fields seen
2129  0        !
2130  0        macro FWA$V_VERSION = 0,11,1,0 %;       ! set if version seen
2131  0        macro FWA$V_TYPE = 0,12,1,0 %;          ! set if type seen
2132  0        macro FWA$V_NAME = 0,13,1,0 %;          ! set if name seen
2133  0        macro FWA$V_DIR = 0,14,1,0 %;           ! set if directory spec seen
2134  0        macro FWA$V_DEVICE = 0,15,1,0 %;        ! set if device seen
2135  0        !
2136  0        !    flags for wild cards
2137  0        !
2138  0        macro FWA$V_EXP_VER = 0,16,1,0 %;       ! set if explicit version
2139  0        macro FWA$V_EXP_TYPE = 0,17,1,0 %;      ! set if explicit type
2140  0        macro FWA$V_EXP_NAME = 0,18,1,0 %;      ! set if explicit name
2141  0        macro FWA$V_WC_VER = 0,19,1,0 %;        ! set if wildcard (*) version
2142  0        macro FWA$V_WC_TYPE = 0,20,1,0 %;       ! " type
2143  0        macro FWA$V_WC_NAME = 0,21,1,0 %;       ! " name
2144  0        macro FWA$V_EXP_DIR = 0,22,1,0 %;       ! set if explicit directory
2145  0        macro FWA$V_EXP_DEV = 0,23,1,0 %;       ! set if explicit device
2146  0        !
2147  0        !    flags for parse results
2148  0        !
2149  0        macro FWA$V_WILDCARD = 0,24,1,0 %;      ! set if any wildcard seen
```

```
2150  0      macro FWA$V_NODE = 0,25,1,0 %;              ! set if node name seen
2151  0      macro FWA$V_QUOTED = 0,26,1,0 %;            ! set is quoted string seen
2152  0      macro FWA$V_GRPMBR = 0,27,1,0 %;            ! set if directory in [grp,mbr] format
2153  0      macro FWA$V_WILD_DIR = 0,28,1,0 %;          ! inclusive or of directory wild cards
2154  0      macro FWA$V_DIR_LVLS = 0,29,3,0 %;
2155  0      literal FWA$S_DIR_LVLS = 3;                 ! ! of directory sublevels (0 = ufd only)
2156  0        ! (valid only if node set and no fldflgs)
2157  0      !
2158  0      !   flags primarily for directory spec
2159  0      !
2160  0      macro FWA$V_DIR1 = 4,0,1,0 %;               ! ufd level directory or group seen
2161  0      macro FWA$V_DIR2 = 4,1,1,0 %;               ! sfd level 1 directory or member seen
2162  0      !
2163  0      !   directory wild flags
2164  0      !
2165  0      macro FWA$V_WILD_UFD = 4,8,1,0 %;           ! the dir1 spec was a wild card
2166  0      macro FWA$V_WILD_SFD1 = 4,9,1,0 %;          ! the dir2 spec was a wild card
2167  0      macro FWA$V_WILD_GRP = 4,8,1,0 %;           ! the grp spec contained a wild card
2168  0      macro FWA$V_WILD_MBR = 4,9,1,0 %;           ! the mbr spec contained a wild card
2169  0      !
2170  0      !   logical name flag and miscellaneous byte
2171  0      !
2172  0      macro FWA$V_LOGNAME = 4,16,1,0 %;           ! a logical name has been seen this pass
2173  0      !   (note: this byte is saved as context
2174  0      !    when processing [.dir-list] format)
2175  0      macro FWA$V_OBJTYPE = 4,17,1,0 %;           ! set if quoted string is of the
2176  0      !    "objecttype=..." form
2177  0      !    (valid only if quoted set)
2178  0      macro FWA$V_NETSTR = 4,18,1,0 %;            ! set if quoted string is of the
2179  0      !    "objecttype=taskname/..." form
2180  0      !    (valid only if quoted and objtype set)
2181  0      macro FWA$V_DEV_UNDER = 4,19,1,0 %;         ! device name was prefixed with an underscore
2182  0      macro FWA$V_FILEFOUND = 4,20,1,0 %;         ! true if at least one file found by search
2183  0      macro FWA$V_REMRESULT = 4,21,1,0 %;         ! use resultant string returned by fal
2184  0      macro FWA$V_SYNTAX_CHK = 4,22,1,0 %;        ! syntax-only checking is requested (NAM$V_SYNCHK set)
2185  0      !
2186  0      !   search list and rooted directory flag byte
2187  0      !
2188  0      macro FWA$V_SLPRESENT = 4,24,1,0 %;         ! search list present
2189  0      macro FWA$V_CONCEAL_DEV = 4,25,1,0 %;       ! concealed device present
2190  0      macro FWA$V_ROOT_DIR = 4,26,1,0 %;          ! root directory present
2191  0      macro FWA$V_DFLT_MFD = 4,27,1,0 %;          ! default MFD string inserted, due to [-]
2192  0      macro FWA$V_EXP_ROOT = 4,28,1,0 %;          ! explicit root directory
2193  0      !
2194  0      !   Value for all filename elements except node
2195  0      !
2196  0      macro FWA$B_BID = 8,0,8,0 %;                ! bid
2197  0      macro FWA$B_BLN = 9,0,8,0 %;                ! bln
2198  0      macro FWA$B_DIRTERM = 10,0,8,0 %;           ! directory spec terminator (']' or '>')
2199  0      macro FWA$B_ROOTERM = 11,0,8,0 %;           ! root directory spec terminator (']' or '>')
2200  0      macro FWA$L_ESCSTRING = 12,0,32,0 %;        ! escape equivalence string
2201  0      macro FWA$B_ESCFLG = 12,0,8,0 %;            ! set to the char <esc> if an escape string
2202  0      !   seen, zero otherwise
2203  0      macro FWA$B_ESCTYP = 13,0,8,0 %;            ! escape 'type' byte
2204  0      macro FWA$W_ESCIFI = 14,0,16,0 %;           ! escape ifi value
2205  0      macro FWA$Q_FIB = 16,0,0,0 %;
2206  0      literal FWA$S_FIB = 8;                      ! fib descriptor
```

```
2207   0       macro FWA$L_DEVBUFSIZ = 24,0,32,0 %;          ! device buffer size
2208   0       macro FWA$L_DEV_CLASS = 28,0,32,0 %;          ! device class
2209   0       macro FWA$L_RECSIZ = 32,0,32,0 %;             ! blocked record size
2210   0       macro FWA$L_UNIT = 36,0,32,0 %;               ! device unit number
2211   0       macro FWA$L_UIC = 40,0,32,0 %;                ! file owner uic
2212   0       macro FWA$W_PRO = 44,0,16,0 %;                ! file protection word
2213   0       macro FWA$B_DIRLEN = 46,0,8,0 %;              ! overall directory spec length
2214   0       macro FWA$B_SUBNODCNT = 47,0,8,0 %;           ! number of secondary (sub) node specs found
2215   0       macro FWA$L_DIRBDB = 48,0,32,0 %;             ! address of directory file bdb
2216   0       macro FWA$L_LOOKUP = 52,0,32,0 %;             ! address of new directory cache node
2217   0       macro FWA$L_DEVNODADR = 56,0,32,0 %;          ! address of device directory cache node
2218   0       macro FWA$Q_DIR = 60,0,0,0 %;
2219   0       literal FWA$S_DIR = 8;                        ! directory name scratch buffer
2220   0       macro FWA$L_UCHAR = 68,0,32,0 %;              ! user characteristics longword
2221   0       macro FWA$W_UCHAR = 68,0,16,0 %;
2222   0       macro FWA$L_FWA_PTR = 72,0,32,0 %;            ! pointer to second fwa if any ($RENAME)
2223   0       macro FWA$L_SWB_PTR = 76,0,32,0 %;            ! pointer to swb
2224   0       macro FWA$L_BUF_PTR = 80,0,32,0 %;            ! address of temporary buffer
2225   0       macro FWA$L_IMPURE_AREA = 84,0,32,0 %;        ! saved R11 (rm$xpfn only)
2226   0       macro FWA$L_ATR_WORK = 88,0,32,0 %;           ! pointer to work area for ACP attributes
2227   0       !   (zero if one not currently allocated)
2228   0       ! ++
2229   0       !
2230   0       !  Logical name and search list fields
2231   0       !
2232   0       ! --
2233   0       !
2234   0       !  Item list block for logical name services
2235   0       !
2236   0       macro FWA$T_ITMLST = 92,0,0,0 %;
2237   0       literal FWA$S_ITMLST = 64;                    ! logical name item list
2238   0       macro FWA$T_ITM_INDEX = 92,0,0,0 %;
2239   0       literal FWA$S_ITM_INDEX = 12;                 ! index
2240   0       macro FWA$T_ITM_ATTR = 104,0,0,0 %;
2241   0       literal FWA$S_ITM_ATTR = 12;                  ! attributes
2242   0       macro FWA$T_ITM_STRING = 116,0,0,0 %;
2243   0       literal FWA$S_ITM_STRING = 12;                ! string
2244   0       macro FWA$T_ITM_MAX_INDEX = 128,0,0,0 %;
2245   0       literal FWA$S_ITM_MAX_INDEX = 12;             ! max index
2246   0       macro FWA$L_ITM_END = 140,0,32,0 %;           ! terminating longword
2247   0       !
2248   0       !  Logical name translation fields
2249   0       !
2250   0       macro FWA$B_BUFFLG = 156,0,8,0 %;             ! flag for which translation buffer is in use
2251   0       !   (0 = buf2 in use, -1 = buf1 in use)
2252   0       macro FWA$B_XLTMODE = 157,0,8,0 %;            ! mode of translation on input to $TRNLNM
2253   0       !   mode of equivalence string on output from $TRNLNM
2254   0       macro FWA$W_XLTSIZ = 158,0,16,0 %;            ! length of equivalence string
2255   0       macro FWA$L_XLTBUFF1 = 160,0,32,0 %;          ! primary translation buffer descriptor
2256   0       macro FWA$L_XLTBUFF2 = 164,0,32,0 %;          ! secondary translation buffer descriptor
2257   0       !
2258   0       !  SLBH and SLB pointers
2259   0       !
2260   0       macro FWA$L_SLBH_PTR = 168,0,32,0 %;          ! current SLB list
2261   0       macro FWA$L_SLB_PTR = 172,0,32,0 %;           ! current SLB list
2262   0       macro FWA$L_SLBH_FLINK = 176,0,32,0 %;        ! SLBH que fwd link
2263   0       macro FWA$L_SLBH_BLINK = 180,0,32,0 %;        ! SLBH que back link
```

```
2264  0       !
2265  0       !   Fake SLB -    NOTE:          This MUST be the size of SLB$C_BLN
2266  0       !                                The field FWA$B_LEVEL must be at the same offset
2267  0       !                                as SLB$Q_LEVEL would be.  (It sounds like a real
2268  0       !                                hack but it works very nicely)
2269  0       !
2270  0       macro FWA$T_SLB = 184,0,0,0 %;
2271  0       literal FWA$S_SLB = 24;                          !  space for SLB$C_BLN
2272  0       macro FWA$B_LEVEL = 195,0,8,0 %;                 !  recursion level
2273  0       !
2274  0       !   Logical name descriptor
2275  0       !
2276  0       macro FWA$Q_LOGNAM = 208,0,0,0 %;
2277  0       literal FWA$S_LOGNAM = 8;                        !  logical name descriptor
2278  0       ! ++
2279  0       !
2280  0       !   descriptors for parsed filename elements
2281  0       !
2282  0       !
2283  0       !       The descriptors are defined as:
2284  0       !
2285  0       !       ----------------------------------------------
2286  0       !       !       flags             !         length   !
2287  0       !       !---------------------------------------------!
2288  0       !       !                     address                 !
2289  0       !       ----------------------------------------------
2290  0       !
2291  0       !
2292  0       !       The flags are defined by FSCB$V_flag in $FSCBDEF
2293  0       !
2294  0       ! --
2295  0       macro FWA$Q_NODE = 216,0,0,0 %;
2296  0       literal FWA$S_NODE = 8;                          !  node name (actually node spec list) descriptor
2297  0       !   (the associated buffer is fwa$t_nodebuf)
2298  0       macro FWA$Q_DEVICE = 224,0,0,0 %;
2299  0       literal FWA$S_DEVICE = 8;                        !  device name descriptor
2300  0       macro FWA$Q_CONCEAL_DEV = 232,0,0,0 %;
2301  0       literal FWA$S_CONCEAL_DEV = 8;                   !  concealed device descriptor
2302  0       !
2303  0       !       directory name descriptors    NOTE:  The two sets of directory
2304  0       !                                            descriptors must be contigous
2305  0       !                                            or RM$SETDID will break
2306  0       !
2307  0       macro FWA$Q_CDIR1 = 240,0,0,0 %;
2308  0       literal FWA$S_CDIR1 = 8;                         !  concealed top directory descriptors
2309  0       macro FWA$Q_CDIR2 = 248,0,0,0 %;
2310  0       literal FWA$S_CDIR2 = 8;                         !  concealed subdirectory 1
2311  0       macro FWA$Q_CDIR3 = 256,0,0,0 %;
2312  0       literal FWA$S_CDIR3 = 8;                         !  "  " 2
2313  0       macro FWA$Q_CDIR4 = 264,0,0,0 %;
2314  0       literal FWA$S_CDIR4 = 8;                         !  "  " 3
2315  0       macro FWA$Q_CDIR5 = 272,0,0,0 %;
2316  0       literal FWA$S_CDIR5 = 8;                         !  "  " 4
2317  0       macro FWA$Q_CDIR6 = 280,0,0,0 %;
2318  0       literal FWA$S_CDIR6 = 8;                         !  "  " 5
2319  0       macro FWA$Q_CDIR7 = 288,0,0,0 %;
2320  0       literal FWA$S_CDIR7 = 8;                         !  "  " 6
```

```
2321  0         macro FWA$Q_CDIR8 = 296,0,0,0 %;
2322  0         literal FWA$S_CDIR8 = 8;                    !  " "  7
2323  0         macro FWA$Q_DIR1 = 304,0,0,0 %;
2324  0         literal FWA$S_DIR1 = 8;                     !  top level directory descriptors
2325  0         macro FWA$Q_DIR2 = 312,0,0,0 %;
2326  0         literal FWA$S_DIR2 = 8;                     !  subdirectory 1
2327  0         macro FWA$Q_DIR3 = 320,0,0,0 %;
2328  0         literal FWA$S_DIR3 = 8;                     !  " 2
2329  0         macro FWA$Q_DIR4 = 328,0,0,0 %;
2330  0         literal FWA$S_DIR4 = 8;                     !  " 3
2331  0         macro FWA$Q_DIR5 = 336,0,0,0 %;
2332  0         literal FWA$S_DIR5 = 8;                     !  " 4
2333  0         macro FWA$Q_DIR6 = 344,0,0,0 %;
2334  0         literal FWA$S_DIR6 = 8;                     !  " 5
2335  0         macro FWA$Q_DIR7 = 352,0,0,0 %;
2336  0         literal FWA$S_DIR7 = 8;                     !  " 6
2337  0         macro FWA$Q_DIR8 = 360,0,0,0 %;
2338  0         literal FWA$S_DIR8 = 8;                     !  " 7
2339  0         macro FWA$Q_NAME = 368,0,0,0 %;
2340  0         literal FWA$S_NAME = 8;                     !  file name descriptor
2341  0         macro FWA$Q_QUOTED = 368,0,0,0 %;
2342  0         literal FWA$S_QUOTED = 8;                   !  quoted string descriptor
2343  0         macro FWA$Q_TYPE = 376,0,0,0 %;
2344  0         literal FWA$S_TYPE = 8;                     !  file type descriptor
2345  0         macro FWA$Q_VERSION = 384,0,0,0 %;
2346  0         literal FWA$S_VERSION = 8;                  !  file version descriptor
2347  0         macro FWA$Q_RNS = 392,0,0,0 %;
2348  0         literal FWA$S_RNS = 8;                      !  resultant name string descriptor
2349  0         macro FWA$Q_SHRFIL = 400,0,0,0 %;
2350  0         literal FWA$S_SHRFIL = 8;                   !  shared file device descriptor (readable form)
2351  0         macro FWA$Q_SHRFIL_LCK = 408,0,0,0 %;
2352  0         literal FWA$S_SHRFIL_LCK = 8;               !  shared file device descriptor (unreadable form - used for lock name)
2353  0         macro FWA$Q_AS_SHRFIL = 416,0,0,0 %;
2354  0         literal FWA$S_AS_SHRFIL = 8;                !  secondary device descriptor (readable form)
2355  0         macro FWA$T_STATBLK = 424,0,0,0 %;
2356  0         literal FWA$S_STATBLK = 10;                 !
2357  0         macro FWA$L_SBN = 424,0,32,0 %;             !  starting lbn if contiguous
2358  0         macro FWA$L_HBK = 428,0,32,0 %;             !  high vbn
2359  0         !
2360  0         !   node descriptors
2361  0         !
2362  0         macro FWA$Q_NODE1 = 436,0,0,0 %;
2363  0         literal FWA$S_NODE1 = 8;                    !  primary node spec descriptor
2364  0         !  (the associated buffer is fwa$t_nodebuf)
2365  0         macro FWA$Q_NODE2 = 444,0,0,0 %;
2366  0         literal FWA$S_NODE2 = 8;                    !  secondary (sub) node spec descriptors (1-7)
2367  0         macro FWA$Q_NODE3 = 452,0,0,0 %;
2368  0         literal FWA$S_NODE3 = 8;                    !  note: bytes 2-3 of each of these descriptors
2369  0         macro FWA$Q_NODE4 = 460,0,0,0 %;
2370  0         literal FWA$S_NODE4 = 8;                    !  contains the flags word that is output
2371  0         macro FWA$Q_NODE5 = 468,0,0,0 %;
2372  0         literal FWA$S_NODE5 = 8;                    !  from nxtfld subroutine in rm0xpfn
2373  0         macro FWA$Q_NODE6 = 476,0,0,0 %;
2374  0         literal FWA$S_NODE6 = 8;                    !  note: fwa$q_node1 thru 'fwa$q_node8'
2375  0         macro FWA$Q_NODE7 = 484,0,0,0 %;
2376  0         literal FWA$S_NODE7 = 8;                    !  describe the same string as does
2377  0         macro FWA$Q_NODE8 = 492,0,0,0 %;
```

```
2378  0          literal FWA$S_NODE8 = 8;                    ! fwa$q_node
2379  0          macro FWA$T_FIBBUF = 500,0,0,0 %;
2380  0          literal FWA$S_FIBBUF = 76;                  ! fib buffer
2381  0          macro FWA$T_RNM_FID = 576,0,0,0 %;
2382  0          literal FWA$S_RNM_FID = 6;                  ! saved fid for rename directory check
2383  0
2384  0      !   directory name buffers
2385  0      !
2386  0      !   NOTE:  These buffers must be contiguous
2387  0      !
2388  0          macro FWA$T_DIR1BUF = 582,0,0,0 %;
2389  0          literal FWA$S_DIR1BUF = 39;                 ! ufd level (or group)
2390  0          macro FWA$T_DIR2BUF = 621,0,0,0 %;
2391  0          literal FWA$S_DIR2BUF = 39;                 ! 1st sfd level (or member)
2392  0          macro FWA$T_DIR3BUF = 660,0,0,0 %;
2393  0          literal FWA$S_DIR3BUF = 39;                 ! subdirectory 2
2394  0          macro FWA$T_DIR4BUF = 699,0,0,0 %;
2395  0          literal FWA$S_DIR4BUF = 39;                 ! subdirectory 3
2396  0          macro FWA$T_DIR5BUF = 738,0,0,0 %;
2397  0          literal FWA$S_DIR5BUF = 39;                 ! subdirectory 4
2398  0          macro FWA$T_DIR6BUF = 777,0,0,0 %;
2399  0          literal FWA$S_DIR6BUF = 39;                 ! subdirectory 5
2400  0          macro FWA$T_DIR7BUF = 816,0,0,0 %;
2401  0          literal FWA$S_DIR7BUF = 39;                 ! subdirectory 6
2402  0          macro FWA$T_DIR8BUF = 855,0,0,0 %;
2403  0          literal FWA$S_DIR8BUF = 39;                 ! subdirectory 7
2404  0          macro FWA$T_CDIR1BUF = 894,0,0,0 %;
2405  0          literal FWA$S_CDIR1BUF = 39;                ! ufd level (or group)
2406  0          macro FWA$T_CDIR2BUF = 933,0,0,0 %;
2407  0          literal FWA$S_CDIR2BUF = 39;                ! 1st sfd level (or member)
2408  0          macro FWA$T_CDIR3BUF = 972,0,0,0 %;
2409  0          literal FWA$S_CDIR3BUF = 39;                ! subdirectory 2
2410  0          macro FWA$T_CDIR4BUF = 1011,0,0,0 %;
2411  0          literal FWA$S_CDIR4BUF = 39;                ! subdirectory 3
2412  0          macro FWA$T_CDIR5BUF = 1050,0,0,0 %;
2413  0          literal FWA$S_CDIR5BUF = 39;                ! subdirectory 4
2414  0          macro FWA$T_CDIR6BUF = 1089,0,0,0 %;
2415  0          literal FWA$S_CDIR6BUF = 39;                ! subdirectory 5
2416  0          macro FWA$T_CDIR7BUF = 1128,0,0,0 %;
2417  0          literal FWA$S_CDIR7BUF = 39;                ! subdirectory 6
2418  0          macro FWA$T_CDIR8BUF = 1167,0,0,0 %;
2419  0          literal FWA$S_CDIR8BUF = 39;                ! subdirectory 7
2420  0      !
2421  0      !   NOTES: 1. The following buffers must be contiguous as eventually the
2422  0      !             type and version are appended to the name string
2423  0      !
2424  0      !          2. The name buffer and the type buffer must be 1 byte larger then
2425  0      !             the max name and type size (resp) because xpfn writes the
2426  0      !             name and type terminators in the buffer at the end of the string
2427  0      !
2428  0          macro FWA$T_NAMEBUF = 1206,0,0,0 %;
2429  0          literal FWA$S_NAMEBUF = 256;                ! file name/quoted string buffer
2430  0          macro FWA$T_TYPEBUF = 1462,0,0,0 %;
2431  0          literal FWA$S_TYPEBUF = 40;                 ! file type buffer
2432  0          macro FWA$T_VERBUF = 1502,0,0,0 %;
2433  0          literal FWA$S_VERBUF = 6;                   ! file version buffer
2434  0          macro FWA$L_UCBSTS = 1508,0,32,0 %;         ! ucb$l_sts field for prim device
```

```
2435   0         macro FWA$B_UNDER_DEV = 1512,0,8,0 %;        ! character "_" stored here
2436   0         macro FWA$T_DEVICEBUF = 1513,0,0,0 %;
2437   0         literal FWA$S_DEVICEBUF = 255;               ! device name buffer
2438   0         macro FWA$T_CDEVICEBUF = 1768,0,0,0 %;
2439   0         literal FWA$S_CDEVICEBUF = 256;              ! concealed device name buffer
2440   0         macro FWA$B_UNDER_NOD = 2024,0,8,0 %;        ! character "_" stored here
2441   0         macro FWA$T_NODEBUF = 2025,0,0,0 %;
2442   0         literal FWA$S_NODEBUF = 127;                 ! node name buffer
2443   0         macro FWA$T_WILD = 2152,0,0,0 %;
2444   0         literal FWA$S_WILD = 48;                     ! scratch field used by RMOWILD
2445   0              size =          count        1
2446   0                              name        39
2447   0                              .dir;*        6
2448   0                              spare         2
2449   0                                     -------------
2450   0                                          48
2451   0         macro FWA$T_SHRFILBUF = 2200,0,0,0 %;
2452   0         literal FWA$S_SHRFILBUF = 16;               ! shared file device id buffer (readable form)
2453   0         macro FWA$T_SHRFIL_LCKNAM = 2216,0,0,0 %;
2454   0         literal FWA$S_SHRFIL_LCKNAM = 16;           ! shared file device id buffer (unreadable form - used for lock name)
2455   0         macro FWA$T_AS_SHRFILBUF = 2232,0,0,0 %;
2456   0         literal FWA$S_AS_SHRFILBUF = 16;            ! secondary device id buffer (readable form)
2457   0         macro FWA$Q_BIJNL = 2248,0,0,0 %;
2458   0         literal FWA$S_BIJNL = 8;                    ! descriptor of BI journal name
2459   0         macro FWA$Q_AIJNL = 2256,0,0,0 %;
2460   0         literal FWA$S_AIJNL = 8;                    ! descriptor of AI journal name
2461   0         macro FWA$Q_ATJNL = 2264,0,0,0 %;
2462   0         literal FWA$S_ATJNL = 8;                    ! descriptor of AT journal name
2463   0         macro FWA$T_BIACE = 2272,0,0,0 %;
2464   0         literal FWA$S_BIACE = 20;                   ! BI journal name ACE
2465   0         macro FWA$T_BIJNLN = 2276,0,0,0 %;
2466   0         literal FWA$S_BIJNLN = 16;
2467   0         macro FWA$T_AIACE = 2292,0,0,0 %;
2468   0         literal FWA$S_AIACE = 20;                   ! AI journal name ACE
2469   0         macro FWA$T_AIJNLN = 2296,0,0,0 %;
2470   0         literal FWA$S_AIJNLN = 16;
2471   0         macro FWA$T_ATACE = 2312,0,0,0 %;
2472   0         literal FWA$S_ATACE = 20;                   ! AT journal name ACE
2473   0         macro FWA$T_ATJNLN = 2316,0,0,0 %;
2474   0         literal FWA$S_ATJNLN = 16;
2475   0         macro FWA$T_IDACE = 2332,0,0,0 %;
2476   0         literal FWA$S_IDACE = 32;                   ! Journal ID ACE
2477   0         macro FWA$T_JNLID = 2336,0,0,0 %;
2478   0         literal FWA$S_JNLID = 28;                   ! complete journal ID
2479   0         macro FWA$T_VOLNAM = 2336,0,0,0 %;
2480   0         literal FWA$S_VOLNAM = 12;                  ! volume lable of media file resides on
2481   0         macro FWA$T_FID = 2348,0,0,0 %;
2482   0         literal FWA$S_FID = 6;                      ! file-id
2483   0         macro FWA$Q_ID_DATE = 2356,0,0,0 %;
2484   0         literal FWA$S_ID_DATE = 8;                  ! id time stamp
2485   0
2486   0         !*** MODULE $SLBHDEF ***
2487   0         !
2488   0         !        SLBH          - Search List Header Block
2489   0         !
2490   0         literal SLBH$C_BID = 43;                    ! ID
2491   0         literal SLBH$K_BLN = 20;                    ! length of SLBH
```

```
2492  0        literal SLBH$C_BLN = 20;                          ! Length of SLBH
2493  0        literal SLBH$S_SLBHDEF = 20;
2494  0        macro SLBH$L_FLINK = 0,0,32,0 %;                  ! forward link
2495  0        macro SLBH$L_BLINK = 4,0,32,0 %;                  ! backward link
2496  0        macro SLBH$B_BID = 8,0,8,0 %;                     ! block ID
2497  0        macro SLBH$B_BLN = 9,0,8,0 %;                     ! length
2498  0        macro SLBH$B_PASSFLGS = 10,0,8,0 %;               ! flags for FWA$B_PASSFLGS
2499  0        macro SLBH$B_STR_LEN = 11,0,8,0 %;                ! string length
2500  0        macro SLBH$L_SLB_QUE = 12,0,32,0 %;               ! ptr to SLB queue
2501  0        macro SLBH$L_NAM_FNB = 16,0,32,0 %;               ! saved FNB from RLF file
2502  0        macro SLBH$T_STRING = 20,0,0,0 %;                 ! start of string
2503  0
2504  0        !*** MODULE $SLBDEF ***
2505  0        !
2506  0        !       SLB          - Search List Block
2507  0        !
2508  0        literal SLB$C_BID = 41;                           ! ID
2509  0        literal SLB$M_REALSLB = 1;
2510  0        literal SLB$K_BLN = 24;                           ! length of SLB
2511  0        literal SLB$C_BLN = 24;                           ! length of SLB
2512  0        literal SLB$S_SLBDEF = 24;
2513  0        macro SLB$L_FLINK = 0,0,32,0 %;                   ! forward link
2514  0        macro SLB$L_BLINK = 4,0,32,0 %;                   ! backward link
2515  0        macro SLB$B_BID = 8,0,8,0 %;                      ! block ID
2516  0        macro SLB$B_BLN = 9,0,8,0 %;                      ! length
2517  0        macro SLB$B_FLAGS = 10,0,8,0 %;                   ! flags
2518  0        macro SLB$V_REALSLB = 10,0,1,0 %;                 ! 'Real' SLB as opposed to the fake FWA one
2519  0        macro SLB$B_LEVEL = 11,0,8,0 %;                   ! recursion level
2520  0        macro SLB$L_INDEX = 12,0,32,0 %;                  ! translation index
2521  0        macro SLB$L_MAX_INDEX = 16,0,32,0 %;              ! max translation index
2522  0        macro SLB$L_ATTR = 20,0,32,0 %;                   ! attributes flags
2523  0
2524  0        !*** MODULE $FSCBDEF ***
2525  0        !
2526  0        !       FSCB - FileScan control block
2527  0        !
2528  0        !       This block is passed to PARSE_STRING from XPFN and RMS$FILESCAN
2529  0        !
2530  0        !
2531  0        !       The descriptors are defined as:
2532  0        !
2533  0        !       -------------------------------------------------
2534  0        !       !      flags          !          length         !
2535  0        !       !------------------------------------------------!
2536  0        !       !                   address                      !
2537  0        !       -------------------------------------------------
2538  0        !
2539  0        !
2540  0        !
2541  0        !       descriptor flags
2542  0        !
2543  0        !       These flags are used through out the RMS file name parsing routines.
2544  0        !       The flags can be found in all of the field descriptors.
2545  0        !
2546  0        !       NOTE: The flag ELIPS must be the first bit in the second word.
2547  0        !             It is referenced this way in RM0WILD and other places
2548  0        !
```

```
2549  0        literal FSCB$M_ELIPS = 65536;
2550  0        literal FSCB$M_WILD = 131072;
2551  0        literal FSCB$M_ACS = 262144;
2552  0        literal FSCB$M_QUOTED = 524288;
2553  0        literal FSCB$M_NULL = 1048576;
2554  0        literal FSCB$M_PWD = 2097152;
2555  0        literal FSCB$M_GRPMBR = 4194304;
2556  0        literal FSCB$M_MINUS = 8388608;
2557  0        literal FSCB$M_CONCEAL = 16777216;
2558  0        literal FSCB$M_MFD = 33554432;
2559  0        literal FSCB$M_ROOTED = 67108864;
2560  0        literal FSCB$S_FSCBDEF = 4;
2561  0        macro FSCB$V_ELIPS = 0,16,1,0 %;       ! elipssis was detected in directory (dir)
2562  0        macro FSCB$V_WILD = 0,17,1,0 %;        ! a wild card was detected (dir,name,type,ver)
2563  0        macro FSCB$V_ACS = 0,18,1,0 %;         ! access control string in node name (node)
2564  0        macro FSCB$V_QUOTED = 0,19,1,0 %;      ! quoted file spec (name)
2565  0        macro FSCB$V_NULL = 0,20,1,0 %;        ! field was null (terminator only) (all)
2566  0        macro FSCB$V_PWD = 0,21,1,0 %;         ! password masked out (set in xpfn) (node)
2567  0        macro FSCB$V_GRPMBR = 0,22,1,0 %;      ! group,member format directory (dir)
2568  0        macro FSCB$V_MINUS = 0,23,1,0 %;       ! minus directory field (dir)
2569  0        macro FSCB$V_CONCEAL = 0,24,1,0 %;     ! name was concealed (dev)
2570  0        macro FSCB$V_MFD = 0,25,1,0 %;         ! MFD directory (set in xpfn) (dir)
2571  0        macro FSCB$V_ROOTED = 0,26,1,0 %;      ! directory was a root directory (dir)
2572  0        !
2573  0        !          FSCB
2574  0        !
2575  0        literal FSCB$M_NODE = 1;
2576  0        literal FSCB$M_DEVICE = 2;
2577  0        literal FSCB$M_ROOT = 4;
2578  C        literal FSCB$M_DIRECTORY = 8;
2579  0        literal FSCB$M_NAME = 16;
2580  0        literal FSCB$M_TYPE = 32;
2581  0        literal FSCB$M_VERSION = 64;
2582  0        literal FSCB$C_MAXNODE = 8;            ! max number of node descriptors
2583  0        literal FSCB$C_MAXROOT = 8;            ! max number of root descriptors
2584  0        literal FSCB$K_BLN = 260;
2585  0        literal FSCB$C_BLN = 260;
2586  0        literal FSCB$C_MAXDIR = 8;             ! max number of directory descriptors
2587  0        literal FSCB$S_FSCBDEF1 = 260;
2588  0        macro FSCB$B_FLDFLAGS = 0,0,8,0 %;     ! field flags
2589  0        macro FSCB$V_NODE = 0,0,1,0 %;
2590  0        macro FSCB$V_DEVICE = 0,1,1,0 %;
2591  0        macro FSCB$V_ROOT = 0,2,1,0 %;
2592  0        macro FSCB$V_DIRECTORY = 0,3,1,0 %;
2593  0        macro FSCB$V_NAME = 0,4,1,0 %;
2594  0        macro FSCB$V_TYPE = 0,5,1,0 %;
2595  0        macro FSCB$V_VERSION = 0,6,1,0 %;
2596  0        macro FSCB$B_NODES = 1,0,8,0 %;        ! number of nodes in spec
2597  0        macro FSCB$B_ROOTS = 2,0,8,0 %;        ! number of root directories in spec
2598  0        macro FSCB$B_DIRS = 3,0,8,0 %;         ! number of directories in spec
2599  0        macro FSCB$Q_FILESPEC = 4,0,0,0 %;
2600  0        literal FSCB$S_FILESPEC = 8;           ! full file spec
2601  0        macro FSCB$Q_NODE = 12,0,0,0 %;
2602  0        literal FSCB$S_NODE = 8;               ! full node list spec
2603  0        macro FSCB$Q_DEVICE = 20,0,0,0 %;
2604  0        literal FSCB$S_DEVICE = 8;             ! device spec
2605  0        macro FSCB$Q_ROOT = 28,0,0,0 %;
```

```
2606  0        literal FSCB$S_ROOT = 8;                            !  full root directory list spec
2607  0        macro FSCB$Q_DIRECTORY = 36,0,0,0 %;                !  full directory list spec
2608  0        literal FSCB$S_DIRECTORY = 8;
2609  0        macro FSCB$Q_NAME = 44,0,0,0 %;                     !  file name
2610  0        literal FSCB$S_NAME = 8;
2611  0        macro FSCB$Q_TYPE = 52,0,0,0 %;                     !  file type
2612  0        literal FSCB$S_TYPE = 8;
2613  0        macro FSCB$Q_VERSION = 60,0,0,0 %;                  !  file version
2614  0        literal FSCB$S_VERSION = 8;
2615  0        macro FSCB$Q_NODE1 = 68,0,0,0 %;                    !  the NODEn descriptors must be contiguous
2616  0        literal FSCB$S_NODE1 = 8;
2617  0        macro FSCB$Q_NODE2 = 76,0,0,0 %;
2618  0        literal FSCB$S_NODE2 = 8;
2619  0        macro FSCB$Q_NODE3 = 84,0,0,0 %;
2620  0        literal FSCB$S_NODE3 = 8;
2621  0        macro FSCB$Q_NODE4 = 92,0,0,0 %;
2622  0        literal FSCB$S_NODE4 = 8;
2623  0        macro FSCB$Q_NODE5 = 100,0,0,0 %;
2624  0        literal FSCB$S_NODE5 = 8;
2625  0        macro FSCB$Q_NODE6 = 108,0,0,0 %;
2626  0        literal FSCB$S_NODE6 = 8;
2627  0        macro FSCB$Q_NODE7 = 116,0,0,0 %;
2628  0        literal FSCB$S_NODE7 = 8;
2629  0        macro FSCB$Q_NODE8 = 124,0,0,0 %;
2630  0        literal FSCB$S_NODE8 = 8;
2631  0        macro FSCB$Q_ROOT1 = 132,0,0,0 %;
2632  0        literal FSCB$S_ROOT1 = 8;                           !  the ROOTn descriptors must be contiguous
2633  0        macro FSCB$Q_ROOT2 = 140,0,0,0 %;
2634  0        literal FSCB$S_ROOT2 = 8;
2635  0        macro FSCB$Q_ROOT3 = 148,0,0,0 %;
2636  0        literal FSCB$S_ROOT3 = 8;
2637  0        macro FSCB$Q_ROOT4 = 156,0,0,0 %;
2638  0        literal FSCB$S_ROOT4 = 8;
2639  0        macro FSCB$Q_ROOT5 = 164,0,0,0 %;
2640  0        literal FSCB$S_ROOT5 = 8;
2641  0        macro FSCB$Q_ROOT6 = 172,0,0,0 %;
2642  0        literal FSCB$S_ROOT6 = 8;
2643  0        macro FSCB$Q_ROOT7 = 180,0,0,0 %;
2644  0        literal FSCB$S_ROOT7 = 8;
2645  0        macro FSCB$Q_ROOT8 = 188,0,0,0 %;
2646  0        literal FSCB$S_ROOT8 = 8;
2647  0        macro FSCB$Q_DIRECTORY1 = 196,0,0,0 %;
2648  0        literal FSCB$S_DIRECTORY1 = 8;                      !  the DIRECTORYn descriptors must be contiguous
2649  0        macro FSCB$Q_DIRECTORY2 = 204,0,0,0 %;
2650  0        literal FSCB$S_DIRECTORY2 = 8;
2651  0        macro FSCB$Q_DIRECTORY3 = 212,0,0,0 %;
2652  0        literal FSCB$S_DIRECTORY3 = 8;
2653  0        macro FSCB$Q_DIRECTORY4 = 220,0,0,0 %;
2654  0        literal FSCB$S_DIRECTORY4 = 8;
2655  0        macro FSCB$Q_DIRECTORY5 = 228,0,0,0 %;
2656  0        literal FSCB$S_DIRECTORY5 = 8;
2657  0        macro FSCB$Q_DIRECTORY6 = 236,0,0,0 %;
2658  0        literal FSCB$S_DIRECTORY6 = 8;
2659  0        macro FSCB$Q_DIRECTORY7 = 244,0,0,0 %;
2660  0        literal FSCB$S_DIRECTORY7 = 8;
2661  0        macro FSCB$Q_DIRECTORY8 = 252,0,0,0 %;
2662  0        literal FSCB$S_DIRECTORY8 = 8;
```

```
2663   0
2664   0          !*** MODULE $SWBDEF ***
2665   0
2666   0          !  Directory string work buffer for wild card directory processing
2667   0
2668   0          literal SWB$M_ELLIPSIS = 1;
2669   0          literal SWB$M_BOUNDED = 2;
2670   0          literal SWB$M_WILD = 4;
2671   0          literal SWB$M_DELIMITER = 8;
2672   0          literal SWB$M_TRAVERSE = 16;
2673   0          literal SWB$M_FIRST = 32;
2674   0          literal SWB$M_ELLIPSIS_EXISTS = 64;
2675   0          literal SWB$M_VALID_DID = 128;
2676   0          literal SWB$C_BID = 42;                            !  ID
2677   0          literal SWB$K_BLN = 328;
2678   0          literal SWB$C_BLN = 328;
2679   0          !    wild dir spec
2680   0          literal SWB$S_SWBDEF = 328;
2681   0          macro SWB$B_FLAGS = 0,0,8,0 %;                     !  flags (must be first)
2682   0          macro SWB$V_ELLIPSIS = 0,0,1,0 %;                  !  ellipsis
2683   0          macro SWB$V_BOUNDED = 0,1,1,0 %;                   !  ellipsis bounded
2684   0          macro SWB$V_WILD = 0,2,1,0 %;                      !  wild name
2685   0          macro SWB$V_DELIMITER = 0,3,1,0 %;                 !  following delimiter
2686   0          macro SWB$V_TRAVERSE = 0,4,1,0 %;                  !  should skip subtree
2687   0          macro SWB$V_FIRST = 0,5,1,0 %;                     !  first time through
2688   0          macro SWB$V_ELLIPSIS_EXISTS = 0,6,1,0 %; !  dir spec contains ...
2689   0          macro SWB$V_VALID_DID = 0,7,1,0 %;                 !  FIB DID is valid
2690   0          macro SWB$B_PATLEN = 1,0,8,0 %;                    !  length of current token
2691   0          macro SWB$B_PPOS = 2,0,8,0 %;                      !  position in pattern
2692   0          macro SWB$B_TOKENS_LEFT = 3,0,8,0 %;               !  number of non ... tokens left
2693   0          macro SWB$B_MINIMUM = 4,0,8,0 %;                   !  minimum level for success
2694   0          macro SWB$B_MAXIMUM = 5,0,8,0 %;                   !  maximum level for success
2695   0          macro SWB$B_FIRST_E = 6,0,8,0 %;                   !  token ! of first ellipsis
2696   0          macro SWB$B_DIRWCFLGS = 7,0,8,0 %;                 !  FWA$B_DIRWCFLGS on entry
2697   0          macro SWB$B_BID = 8,0,8,0 %;                       !  block ID
2698   0          macro SWB$B_BLN = 9,0,8,0 %;                       !  length
2699   0          macro SWB$Q_PATTERN = 12,0,0,0 %;
2700   0          literal SWB$S_PATTERN = 8;                         !  descriptor of pattern
2701   0          macro SWB$L_SCRATCH_PAT = 20,0,32,0 %; !  scratch copy of first longword
2702   0          macro SWB$T_SCRATCH_BUF = 24,0,0,0 %;
2703   0          literal SWB$S_SCRATCH_BUF = 48;                    !  scratch temp buffer (same size as FWA$T_WILD)
2704   0          macro SWB$T_PATTERN_BUF = 72,0,0,0 %;
2705   0          literal SWB$S_PATTERN_BUF = 256;                   !  should be: FWA$C_MAXDIRLEN-2.
2706   0
2707   0          !  *********************************************************************
2708   0          !  *                                                                   *
2709   0          !  *  Copyright (c) 1982, 1983                                         *
2710   0          !  *  by DIGITAL Equipment Corporation, Maynard, Mass.                 *
2711   0          !  *                                                                   *
2712   0          !  *  This software is furnished under a license and may be used and  copied  *
2713   0          !  *  only  in  accordance  with  the  terms  of  such  license and with the  *
2714   0          !  *  inclusion of the above copyright notice.  This software or  any  other  *
2715   0          !  *  copies  thereof may not be provided or otherwise made available to any  *
2716   0          !  *  other person.  No title to and ownership of  the  software  is  hereby  *
2717   0          !  *  transferred.                                                     *
2718   0          !  *                                                                   *
2719   0          !  *  The information in this software is subject to change  without  notice  *
```

```
2720  0      *  and  should  not  be  construed  as  a commitment by DIGITAL Equipment  *
2721  0      *  Corporation.                                                            *
2722  0      *                                                                          *
2723  0      *  DIGITAL assumes no responsibility for the use or  reliability  of  its  *
2724  0      *  software on equipment which is not supplied by DIGITAL.                  *
2725  0      *                                                                          *
2726  0      !**************************************************************************
2727  0      !**************************************************************************
2728  0      !  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.
2729  0      !**************************************************************************
2730  0
2731  0
2732  0      !*** MODULE $SFSBDEF ***
2733  0      literal SFSB$C_BID = 16;                      ! sfsb code
2734  0      literal SFSB$C_FIX_LEN = 10;                  ! 10 bytes of fixed size data
2735  0      literal SFSB$K_BLN = 68;                      ! length of sfsb
2736  0      literal SFSB$C_BLN = 68;                      ! length of sfsb
2737  0      !
2738  0      !        keep the next two fields in same order as they are in FAB
2739  0      !
2740  0      literal SFSB$S_SFSBDEF = 68;
2741  0      macro SFSB$Q_FILENAME = 0,0,0,0 %;
2742  0      literal SFSB$S_FILENAME = 8;                  ! descriptor of shared file resource name.
2743  0      !    resource name is NODE, DEVICE, FILE_ID
2744  0      !    points to RESNAM, below
2745  0      macro SFSB$W_NAME_LEN = 0,0,16,0 %;           ! subfield to address descriptor length field
2746  0      macro SFSB$L_ADDRESS = 4,0,32,0 %;            ! subfield to address descriptor address field
2747  0      macro SFSB$B_BID = 8,0,8,0 %;                 ! block id
2748  0      macro SFSB$B_BLN = 9,0,8,0 %;                 ! block length in longwords
2749  0      macro SFSB$B_CURMODE = 10,0,8,0 %;            ! Mode of the current lock
2750  0      macro SFSB$B_PREMODE = 11,0,8,0 %;            ! Mode of the previous lock
2751  0      macro SFSB$T_RESNAM = 12,0,0,0 %;
2752  0      literal SFSB$S_RESNAM = 32;                   ! 32 bytes for name of shared resource
2753  0      macro SFSB$L_FAC_CODE = 12,0,32,0 %;          ! RMS facility code (RMS$)
2754  0      macro SFSB$W_FID_NUM = 16,0,16,0 %;           ! file id word one
2755  0      macro SFSB$W_FID_SEQ = 18,0,16,0 %;           ! file id word two
2756  0      macro SFSB$W_FID_RVN = 20,0,16,0 %;           ! file id word three
2757  0      macro SFSB$T_DEV_NAM = 22,0,0,0 %;
2758  0      literal SFSB$S_DEV_NAM = 22;                  ! 22 bytes remain to hold device id (node$device_name)
2759  0      macro SFSB$L_LKSB = 44,0,32,0 %;              ! lock status block
2760  0      macro SFSB$W_STATUS = 44,0,16,0 %;            ! VMS status code
2761  0      macro SFSB$S_BITS = 46,0,16,0 %;              ! various status bits
2762  0      macro SFSB$L_LOCK_ID = 48,0,32,0 %;           ! second longword of LKSB is the lock id
2763  0      macro SFSB$L_LVB = 52,0,0,0 %;
2764  0      literal SFSB$S_LVB = 16;                      ! lock value block
2765  0      macro SFSB$B_FAC = 52,0,8,0 %;                ! fac bits from FAB
2766  0      macro SFSB$B_SHR = 53,0,8,0 %;                ! sharing bits (from FAB SHR field)
2767  0      macro SFSB$L_HBK = 60,0,32,0 %;               ! high block
2768  0      macro SFSB$L_EBK = 64,0,32,0 %;               ! end of file
2769  0
2770  0      !*** MODULE $GBSBDEF ***
2771  0      !
2772  0      !        GBSB field definitions - global buffer synchronization block
2773  0      !
2774  0      !        The GBSB contains the information necessary to determine if a
2775  0      !        global section is already open for a file on a given node, and
2776  0      !        is used for synchronizing access to the global section.
```
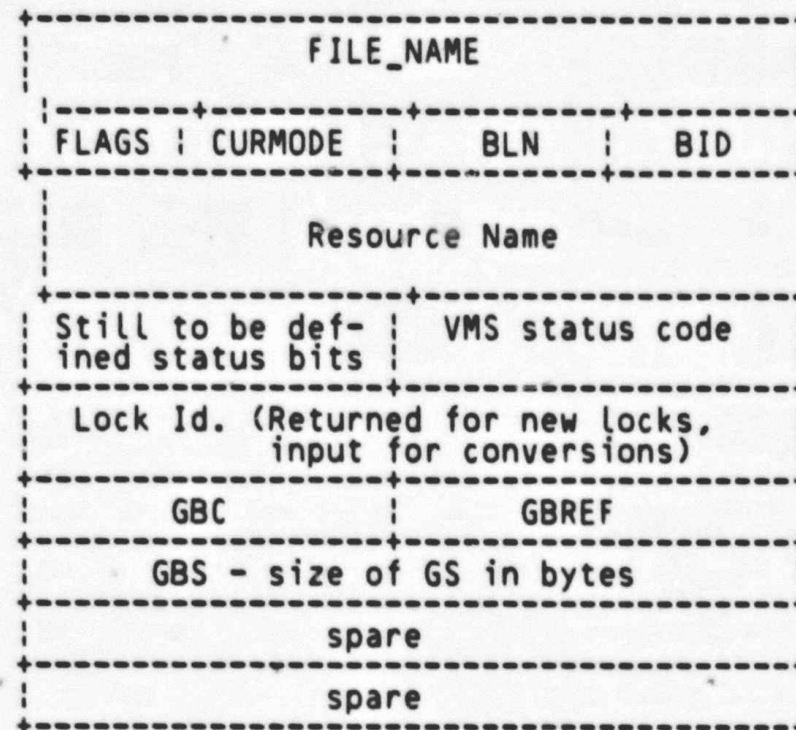
```
2777  0  !
2778  0  !           +-------------------------------------------+
2779  0  !  gbsb:    |                FILE_NAME                  |
2780  0  !           |                                           |
2781  0  !           |-------+---------+------------+------------|
2782  0  !           | FLAGS | CURMODE |    BLN     |    BID     |
2783  0  !           +-------+---------+------------+------------+
2784  0  !           :                                           :
2785  0  !           :              Resource Name                :
2786  0  !           :                                           :
2787  0  !           +-------------------------------------------+
2788  0  !  lksb:    |  Still to be def-  |   VMS status code    |
2789  0  !           |  ined status bits  |                      |
2790  0  !           |-----------------------------------------+
2791  0  !           |  Lock Id. (Returned for new locks,        |
2792  0  !           |           input for conversions)          |
2793  0  !           +-------------------------------------------+
2794  0  !  lkvb:    |       GBC          |       GBREF          :
2795  0  !           |-----------------------------------------+
2796  0  !           |         GBS - size of GS in bytes         |
2797  0  !           +-------------------------------------------+
2798  0  !           :                 spare                     :
2799  0  !           +-------------------------------------------+
2800  0  !           :                 spare                     :
2801  0  !           +-------------------------------------------+
2802  0  !
2803  0  !
2804  0      literal GBSB$C_BID = 9;                      ! gbsb code
2805  0      literal GBSB$M_NOTACCESSED = 1;
2806  0      literal GBSB$K_BLN = 68;                     ! length of gbsb
2807  0      literal GBSB$C_BLN = 68;                     ! length of gbsb
2808  0      literal GBSB$S_GBSBDEF = 68;
2809  0      macro GBSB$Q_FILENAME = 0,0,0,0 %;
2810  0      literal GBSB$S_FILENAME = 8;                 ! descriptor of shared file resource name.
2811  0      !   resource name is NODE, DEVICE, FILE_ID
2812  0      !   points to RESNAM, below
2813  0      macro GBSB$W_NAME_LEN = 0,0,16,0 %;          ! subfield to address descriptor length field
2814  0      macro GBSB$L_ADDRESS = 4,0,32,0 %;           ! subfield to address descriptor address field
2815  0      macro GBSB$B_BID = 8,0,8,0 %;                ! block id
2816  0      macro GBSB$B_BLN = 9,0,8,0 %;                ! block length in longwords
2817  0      macro GBSB$B_CURMODE = 10,0,8,0 %;           ! Mode of the current lock
2818  0      macro GBSB$B_FLAGS = 11,0,8,0 %;             ! spare
2819  0      macro GBSB$V_NOTACCESSED = 11,0,1,0 %;       ! Process has already decremeted access count for GBS.
2820  0      macro GBSB$T_RESNAM = 12,0,0,0 %;
2821  0      literal GBSB$S_RESNAM = 32;                  ! 32 bytes for name of shared resource
2822  0      macro GBSB$L_LKSB = 44,0,32,0 %;             ! lock status block
2823  0      macro GBSB$W_STATUS = 44,0,16,0 %;           ! VMS status code
2824  0      macro GBSB$W_S_BITS = 46,0,16,0 %;           ! various status bits
2825  0      macro GBSB$L_LOCK_ID = 48,0,32,0 %;          ! second longword of LKSB is the lock id
2826  0      macro GBSB$W_GBC = 52,0,16,0 %;              ! Number of global buffers in section.
2827  0      macro GBSB$W_GBREF = 54,0,16,0 %;            ! Number of accessors to global section.
2828  0      macro GBSB$L_GS_SIZE = 56,0,32,0 %;          ! Size of global section in bytes.
2829  0
2830  0      ! Version:        'V04-000'
2831  0
2832  0      !*********************************************************************
2833  0      !*                                                                   *
```
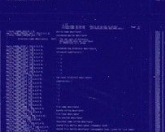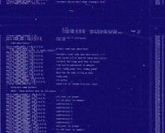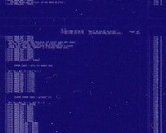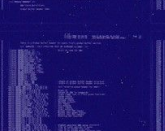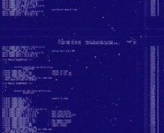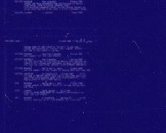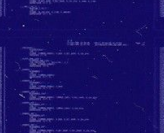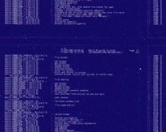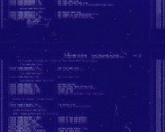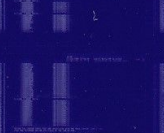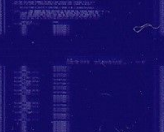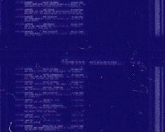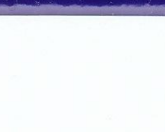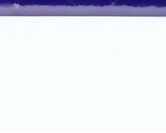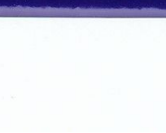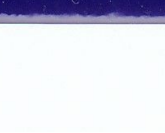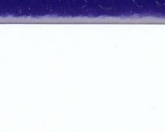
```
2834  0      !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                         *
2835  0      !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.          *
2836  0      !*  ALL RIGHTS RESERVED.                                            *
2837  0      !*                                                                  *
2838  0      !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
2839  0      !*  ONLY IN ACCORDANCE WITH THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
2840  0      !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
2841  0      !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
2842  0      !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
2843  0      !*  TRANSFERRED.                                                     *
2844  0      !*                                                                  *
2845  0      !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
2846  0      !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
2847  0      !*  CORPORATION.                                                     *
2848  0      !*                                                                  *
2849  0      !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
2850  0      !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.         *
2851  0      !*                                                                  *
2852  0      !*                                                                  *
2853  0      !****************************************************************************
2854  0
2855  0      !++
2856  0      ! UTLDEF_UNDECLARE.R32 - undeclare macros defined in UTLDEF.R32.
2857  0      !--
2858  0
2859  0      UNDECLARE %QUOTE $BYTEOFFSET;
2860  0      UNDECLARE %QUOTE $BITPOSITION;
2861  0      UNDECLARE %QUOTE $FIELDWIDTH;
2862  0      UNDECLARE %QUOTE $EXTENSION;
2863  0      UNDECLARE %QUOTE $FIELDMASK;
2864  0      UNDECLARE %QUOTE $EQULST;
2865  0      UNDECLARE %QUOTE GET2ND_;
2866  0      UNDECLARE %QUOTE NUL2ND_;
2867  0      UNDECLARE %QUOTE GET1ST_;
```

                              COMMAND QUALIFIERS

        BLISS/LIB=LIB$:RMSINTDEF/LIS=LIS$:RMSINTDEF.LST LIB$:RMSINTDEF

; Run Time:         00:17.4
; Elapsed Time:     00:19.9
; Lines/CPU Min:    9908
; Lexemes/CPU-Min: 62899
; Memory Used:  292 pages
; Library Precompilation Complete

RMSCALLS
MAR

RMSIDXLNK
R32

RMS
LST

RMS32MAC
MAR

RMSMSCMAC
MAR

UTLDEF
R32

RMSIDXMAC
R32

UTLDEFUND
R32

RMSINTDEF
LST

RMSIDXDEF
R32

NT0MACROS
MAR

NT0ACCESS
LIS

NT0CLOSE
LIS

NT0BLDXAB
LIS

NT0CONN
LIS

NT0CREATE
LIS

NT0DAPIO
LIS

NT0DAPCRC
LIS

NT0AECFIL
LIS

NT0BLKIO
LIS