```
RRRRRRRRRRR      MMM          MMM      SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRRRRRRRRRR      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  SSSSSSSSSSS
RRR       RRR    MMM          MMM               SSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
```

FILEID**RMSIDXMAC

UT

```
RRRRRRRR    MM       MM    SSSSSSSS    IIIIII    DDDDDDDD    XX      XX  MM       MM    AAAAAA      CCCCCCCC
RRRRRRRR    MM       MM    SSSSSSSS    IIIIII    DDDDDDDD    XX      XX  MM       MM    AAAAAA      CCCCCCCC
RR      RR  MMMM   MMMM  SS              II      DD      DD  XX      XX  MMMM   MMMM  AA      AA  CC
RR      RR  MMMM   MMMM  SS              II      DD      DD  XX      XX  MMMM   MMMM  AA      AA  CC
RR      RR  MM MM MM MM  SS              II      DD      DD    XX  XX    MM MM MM MM  AA      AA  CC
RR      RR  MM MM MM MM  SS              II      DD      DD    XX  XX    MM MM MM MM  AA      AA  CC
RRRPRRRR    MM       MM    SSSSSS        II      DD      DD      XX      MM       MM  AA      AA  CC
RRRRRRRR    MM       MM    SSSSSS        II      DD      DD      XX      MM       MM  AA      AA  CC
RR  RR      MM       MM          SS      II      DD      DD    XX  XX    MM       MM  AAAAAAAAAA  CC
RR  RR      MM       MM          SS      II      DD      DD    XX  XX    MM       MM  AAAAAAAAAA  CC
RR    RR    MM       MM          SS      II      DD      DD  XX      XX  MM       MM  AA      AA  CC                ....
RR    RR    MM       MM          SS      II      DD      DD  XX      XX  MM       MM  AA      AA  CC                ....
RR      RR  MM       MM  SSSSSSSS      IIIIII    DDDDDDDD    XX      XX  MM       MM  AA      AA    CCCCCCCC        ....
RR      RR  MM       MM  SSSSSSSS      IIIIII    DDDDDDDD    XX      XX  MM       MM  AA      AA    CCCCCCCC        ....

RRRRRRRR      333333      222222
RR      RR  33      33  22      22
RR      RR  33      33  22      22
RR      RR          33          22
RR      RR          33          22
RRRRRRRR        33          22
RRRRRRRR        33          22
RR  RR              33      22
RR  RR              33      22
RR    RR    33      33    22
RR    RR    33      33    22
RR      RR    333333    2222222222
RR      RR    333333    2222222222
```

```
        $BEGIN RMSIDXMAC,V04-000

        MACRO DEFINITIONS FOR RMS-32 INDEX FILE ORGANIZATION
```



```
;++
;
; FACILITY:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
;
; ABSTRACT:
;
;
; ENVIRONMENT:
;
;               VAX/VMS OPERATING SYSTEM
;
;--
;
; AUTHOR:  D. H. Gillespie     CREATION DATE:  17-MAR-1978
;          and W. Koenig
;
;
; MODIFIED BY:
;
;       V03-002 MCN0003         Maria del C. Nasr       15-Mar-1982
;               Use new general linkage for RMSBUG3.  Take out definition
;               for R_REC_SIZE, and R_VBN.
;
;       V03-001 MCN0002         Maria del C. Nasr       25-Mar-1982
;               Add macro definition to calculate key buffer address.
;
;       V02-003 CDS0001         C Saether                9-Dec-1981
;               Comment out references to CSH$M_READAHEAD flag in the
;               $CSHFLAGS macro.
```

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

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

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

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

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

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

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

```
                %;

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

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

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

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

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

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

        COMMON_IOREG =
                R_BDB, R_BKT_ADDR %,

        COMMON_RABREG =
                R_RAB, R_IRAB, R_IFAB, R_IMPURE%,

        COMMON_FAB_STR =
                R_FAB_STR,
                R_IFAB_STR,
                R_IFAB_FILE_STR,
                R_IMPURE_STR %,

        COMMON_IO_STR =
                R_BDB_STR,
                R_BKT_ADDR_STR %,

        COMMON_RAB_STR =
                R_RAB_STR,
                R_IRAB_STR,
                R_IFAB_STR,
                R_IMPURE_STR%;


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

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


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

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

     RMSSUC (CODE) = (
        %IF %LENGTH EQL 0 %THEN
                1
        %ELSE
                %NAME('RMS$_',CODE) AND %X'FFFF'
        %FI)
        %;
!
! macro to make constants that are calculated have the <0,16> attribute
!
!       macros to make the code a little nicer
!
MACRO
!
! this macro allows you to make a call to another routine
!  (and do whatever you want in a block before the call),
! and if an error resulted, do whatever you want and
!   then return with the status.
!
   RETURN_ON_ERROR (CALL) [] =
       (LOCAL STATUS;
        IF NOT (STATUS = (CALL)) THEN
                                        (%REMAINING;
                                         RETURN .STATUS)) %,
!
! this macro is the same as the one above, except that it
!  it returns w/ status whether or not there was an error
!  and the caller can supply an 'else' block
!   note:  the 'call' part and 'else' part must be separated by a comma
!    not a semicolon and the 'else' part must be terminated by a semicolon
!
   RETURN_ELSE (CALL) [] =
       (LOCAL STATUS;
        IF NOT (STATUS = (CALL)) THEN RETURN .STATUS
        ELSE
                %REMAINING
                RETURN .STATUS)
   %;
```

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

```
    END
        %;
```

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

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

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

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

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

RMSCALLS
MAR

RMSIDXLNK
R32

RMS
LST

RMS32MAC
MAR

RMSMSGMAC
MAR

UTLDEF
R32

RMSIDXMAC
R32

UTLDEFUND
R32

RMSINTDEF
LST

RMSIDXDEF
R32

NT@MACROS
MAR