C 13

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

FILEID**RMSMSCMAC

```
RRRRRRRR    MM      MM    SSSSSSSS  MM      MM    SSSSSSSS    CCCCCCCC  MM      MM    AAAAAA      CCCCCCCC
RRRRRRRR    MM      MM    SSSSSSSS  MM      MM    SSSSSSSS    CCCCCCCC  MM      MM    AAAAAA      CCCCCCCC
RR      RR  MMMM  MMMM  SS          MMMM  MMMM  SS          CC          MMMM  MMMM  AA      AA  CC
RR      RR  MMMM  MMMM  SS          MMMM  MMMM  SS          CC          MMMM  MMMM  AA      AA  CC
RR      RR  MM  MM  MM  SS          MM  MM  MM  SS          CC          MM  MM  MM  AA      AA  CC
RR      RR  MM  MM  MM  SS          MM  MM  MM  SS          CC          MM  MM  MM  AA      AA  CC
RRRRRRRR    MM      MM    SSSSSS    MM      MM    SSSSSS    CC          MM      MM  AA      AA  CC
RRRRRRRR    MM      MM    SSSSSS    MM      MM    SSSSSS    CC          MM      MM  AA      AA  CC
RR  RR      MM      MM          SS  MM      MM          SS  CC          MM      MM  AAAAAAAAAA  CC
RR  RR      MM      MM          SS  MM      MM          SS  CC          MM      MM  AAAAAAAAAA  CC
RR    RR    MM      MM          SS  MM      MM          SS  CC          MM      MM  AA      AA  CC          ....
RR    RR    MM      MM          SS  MM      MM          SS  CC          MM      MM  AA      AA  CC          ....
RR      RR  MM      MM  SSSSSSSS    MM      MM  SSSSSSSS      CCCCCCCC  MM      MM  AA      AA    CCCCCCCC  ....
RR      RR  MM      MM  SSSSSSSS    MM      MM  SSSSSSSS      CCCCCCCC  MM      MM  AA      AA    CCCCCCCC  ....

MM      MM    AAAAAA    RRRRRRRR
MM      MM    AAAAAA    RRRRRRRR
MMMM  MMMM  AA      AA  RR      RR
MMMM  MMMM  AA      AA  RR      RR
MM  MM  MM  AA      AA  RR      RR
MM  MM  MM  AA      AA  RR      RR
MM      MM  AA      AA  RRRRRRRR
MM      MM  AA      AA  RRRRRRRR
MM      MM  AAAAAAAAAA  RR  RR
MM      MM  AAAAAAAAAA  RR  RR
MM      MM  AA      AA  RR    RR
MM      MM  AA      AA  RR    RR
MM      MM  AA      AA  RR      RR
MM      MM  AA      AA  RR      RR
```

```
;++
;
; Facility:     RMS-32
;
; Abstract:     RMS internal macros
;
; Modified By:
;
;       V03-007 DAS0001         David Solomon           13-Feb-1984
;               Remove temporary (and now obsolete) RMS Recovery macros.
;
;       V03-006 SHZ0001         Stephen H. Zalewski     21-Apr-1983
;               Remove obsolete macros.
;
;       V03-005 KBT0448         Keith B. Thompson       5-Dec-1982
;               Remove the $cachec macro
;
;       V03-004 JWH0163         Jeffrey W. Horn         21-Dec-1982
;               Make references to the tracepoint page be offset from
;               a SHELL global variable (PIO$A_TRACE) rather than the
;               top of the process IO impure area.
;
;       V03-003 KPL0001         Peter Lieberwirth       26-Oct-1982
;               Add some macros for RMS recovery.  (Some of these are
;               only temporary.)
;
;       V03-002 RAS0089         Ron Schaefer     7-Jun-1982
;               Delete the incorrect $QUAD_ALIGN macro.
;
;       V03-001 RAS0078         Ron Schaefer    17-Mar-1982
;               Fix the $BEGIN macro for V3.
;
;
```

```
        .NLIST
;
;  macro to generate a global symbol and entr· mask
;

.MACRO  $ENTRY P1,MASK
        $DPSECT
        .IF     NB <MASK>
P1::
        .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
        .IFF
        P1==.-2
        .ENDC
.ENDM   $ENTRY

;
;
;  a test system macro to:
;
;       1. generate a global symbol if $$rmstest bit 0 on
;       2. increment a global location if $$rmstest bit 1 on
;

.MACRO  $TSTPT P1
        .IF     NE $$RMSTEST&1
$$TPS'P1::
        .ENDC
        .IF     NE $$RMSTEST&2
        INCL    @#PIO$A_TRACE+TPT$L_'P1
        .ENDC
.ENDM   $TSTPT

;
;  macro to set a single bit
;

.MACRO  SSB POS,BAS,?L
        BBSS    POS,BAS,L
L:
.ENDM   SSB

;
;  macro to clear a single bit
;

.MACRO  CSB POS,BAS,?L
        BBCC    POS,BAS,L
L:
.ENDM   CSB


;
;  module begin macro
;

.MACRO  $BEGIN MODNAM,IDNUM,PSNAME,TITL,PSATR
```

```
        .TITLE  MODNAM TITL             ;
        .IDENT  /V04-'IDNUM/
.MACRO  $DPSECT
        .PSECT  PSNAME,GBL,PIC,NOWRT,PSATR
.ENDM
        $DPSECT
        $$.PSECT_EP=0
        $$RMSTEST=26                    ; get test point increment, rmsbug,
                                        ; and optional debug code
        $$RMS_UMODE=4
        $$RMS_TBUGCHK=8
        $$RMS_PBUGCHK=16
        .DSABL  DBG
.ENDM   $BEGIN

;
;  the $begin macro defines the following switches for rms assembly
;       (bits in $$rmstest):
;
;       bit 0                   include tstpt global symbols
;       bit 1                   include tstpt counter increment
;       bit 2   ($$rms_umode)   assemble code for user mode
;       bit 3   ($$rms_tbugchk) include optional debugging code
;       bit 4   ($$rms_pbugchk) include "semi-permanent" debugging code
;


;
;  hard error macros
;

.MACRO  RMSPBUG CODE
        .IF     NE $$RMSTEST&$$RMS_PBUGCHK
        $FTLDEF
        $BUGDEF
        JSB     RM$BUG
        .BYTE   CODE
        .ENDC
.ENDM   RMSPBUG
.MACRO  RMSTBUG CODE
        .IF     NE $$RMSTEST&$$RMS_TBUGCHK
        $FTLDEF
        JSB     RM$BUG
        .BYTE   CODE
        .ENDC
.ENDM   RMSTBUG
```

```
;
;  macro to store rms error code
;

.MACRO  RMSERR NAM,ADDR=R0
        RMSSTS  NAM,ADDR
.ENDM   RMSERR

;

.MACRO  RMSSUC NAM=SUC,ADDR=R0
        .IF     IDN <NAM><SUC>
        MOVL    #1,ADDR
        .IFF
        RMSSTS  NAM,ADDR
        .ENDC
.ENDM   RMSSUC

;

.MACRO  RMSSTS NAM,ADDR
        .IF     DF RMS$ 'NAM
        .IF     EQ <RMS$ 'NAM&^XFF00>
        MOVZBL  #<RMS$_'NAM&^XFF>,ADDR
        .IFF
        MOVZWL  #<RMS$_'NAM&^XFFFF>,ADDR
        .ENDC
        .MEXIT
        .IFF
        MOVZWL  #<RMS$_'NAM&^XFFFF>,ADDR
        .ENDC
.ENDM   RMSSTS

;

.MACRO  RMSERR_WORD NAM
        .WORD   RMS$_'NAM&^XFFFF
.ENDM   RMSERR_WORD
```

```
.MACRO  $CACHE VBN=R1,SIZE=R2,FLAGS=R3,ERR=
        .IF     IDN <VBN><#0>
        CLRL    R1
        .IFF
        .IIF    DIF <VBN><R1>, MOVL VBN,R1
        .ENDC
        .IF     IDN <SIZE><#0>
        CLRL    R2
        .IFF
        .IIF    DIF <SIZE><R2>, MOVZWL SIZE,R2
        .ENDC
        .IF     DIF <FLAGS><R3>
        .IIF    IDN <FLAGS><#0>, CLRL R3
        .IIF    DIF <FLAGS><#0>, $CSHFLAGS <FLAGS>
        .ENDC
        BSBW    RM$CACHE
        .IIF    NB <ERR>, BLBC R0,ERR
.ENDM   $CACHE

.MACRO  $CSHFLAGS FLAGS
        $CSHDEF
        $$.TMP=0
        .IRP    X,<FLAGS>
        $$.TMP=$$.TMP!CSH$M_'X
        .ENDM
        .IF     EQ $$.TMP
        CLRL    R3
        .IFF
        .IIF    NE $$.TMP&CSH$M_NOBUFFER, $$.TMP=$$.TMP!CSH$M_NOREAD
                                        ; nobuffer implies noread
        MOVL    #$$.TMP,R3
        .ENDC
.ENDM   $CSHFLAGS
```

```
;
;  macro for specifying optional functions to be performed by
;  rm$rset.  an in-line byte is generated specifying the options.
;  a zero byte indicates no options desired.
;
;  for non-zero values,
;
;       anyfac (bit 5) set specifies that bits 0 - 2 are not to be used to
;               check a fac bit.
;
;       cflg (bit 4) set specifies that irb$v_find_last is
;               to be cleared
;
;       bio (bit 3) set specifies that the file must be accessed
;               for block i/o.; clear specifies that the file must
;               not be accessed for block i/o
;
;       fac (bits 0 - 2) specifies the bit in fac which must be
;               set for this function to be valid (e.g., fab$v_get
;               for $get or $find)
;
;       bits 6 and 7 are spares and must be zero.
;
;

.MACRO  $RABSET FAC=0,BIO=0,CFLG=0,ANYFAC=0
        BSBW    RM$RSET
        .BYTE   <ANYFAC @ 5>!<CFLG @ 4>!<BIO @ 3>!FAC
.ENDM   $RABSET
```

```
;
;  the following macros are used to facilitate the gathering of
;  relatively low usage code paths into separate psects in order
;  to achieve higher normal-usage code locality.  the advantage of
;  this technique as opposed to using separate modules is that
;  having all the relevant code in a single module improves
;  readablity.
;

.MACRO  $NEWPSECT PSNAME,PSATR
        .SAVE
        .PSECT  PSNAME,GBL,PIC,NOWRT,PSATR
.ENDM   $NEWPSECT

;
;  the $psect_restore macro merely returns to the previous psect.
;

.MACRO  $PSECT_RESTORE PSNAME
        .RESTORE
.ENDM   $PSECT_RESTORE

        .LIST
```

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RMSCALLS
MAR

RMSIDXLNK
R32

RMS
LST

RMS32MAC
MAR

RMSMSCMAC
MAR

UTLDEF
R32

RMSIDXMAC
R32

UTLDEFUND
R32

RMSINTDEF
LST

RMSIDXDEF
R32

NT0MACROS
MAR