```
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**RMSOWAIT

```
RRRRRRRR    MM      MM    SSSSSSSS    000000    WW      WW    AAAAAA      IIIIII    TTTTTTTTTT
RRRRRRRR    MM      MM    SSSSSSSS    000000    WW      WW    AAAAAA      IIIIII    TTTTTTTTTT
RR      RR  MMMM  MMMM  SS          00      00  WW      WW  AA      AA      II          TT
RR      RR  MMMM  MMMM  SS          00      00  WW      WW  AA      AA      II          TT
RR      RR  MM  MM  MM  SS          00    0000  WW      WW  AA      AA      II          TT
RR      RR  MM  MM  MM  SS          00    0000  WW      WW  AA      AA      II          TT
RRRRRRRR    MM      MM    SSSSSS    00  00  00  WW      WW  AA      AA      II          TT
RRRRRRRR    MM      MM    SSSSSS    00  00  00  WW      WW  AA      AA      II          TT
RR  RR      MM      MM          SS  0000    00  WW  WW  WW  AAAAAAAAAA      II          TT
RR  RR      MM      MM          SS  0000    00  WW  WW  WW  AAAAAAAAAA      II          TT
RR    RR    MM      MM          SS  00      00  WWWW  WWWW  AA      AA      II          TT         ....
RR    RR    MM      MM          SS  00      00  WWWW  WWWW  AA      AA      II          TT         ....
RR      RR  MM      MM  SSSSSSSS      000000    WW      WW  AA      AA    IIIIII        TT         ....
RR      RR  MM      MM  SSSSSSSS      000000    WW      WW  AA      AA    IIIIII        TT         ....


LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          $BEGIN  RMSOWAIT,000,RM$RMS,<WAIT FOR AYNC OPERATION COMPLETION>
0000     2
0000     3  ;****************************************************************************
0000     4  ;*                                                                          *
0000     5  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     6  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     7  ;*  ALL RIGHTS RESERVED.                                                    *
0000     8  ;*                                                                          *
0000     9  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    10  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    11  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    12  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    13  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    14  ;*  TRANSFERRED.                                                            *
0000    15  ;*                                                                          *
0000    16  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    17  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    18  ;*  CORPORATION.                                                            *
0000    19  ;*                                                                          *
0000    20  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    21  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    22  ;*                                                                          *
0000    23  ;*                                                                          *
0000    24  ;****************************************************************************
0000    25  ;
0000    26  ;++
0000    27  ; FACILITY: RMS32
0000    28  ;
0000    29  ; ABSTRACT:
0000    30  ;               Routine to perform the $WAIT function to allow
0000    31  ;               for synchronization with the completion of asynchronous
0000    32  ;               requests.
0000    33  ;
0000    34  ;
0000    35  ; ENVIRONMENT:
0000    36  ;
0000    37  ; AUTHOR: L F Laverdure,          Creation Date: 3-Feb-1977
0000    38  ;
0000    39  ; Modified By:
0000    40  ;
0000    41  ;       V03-002 RAS0141         Ron Schaefer            6-Apr-1983
0000    42  ;               Change the check for same RAB to be more fool-proof.
0000    43  ;
0000    44  ;       V03-001 KBT0197         Keith B. Thompson       23-Aug-1982
0000    45  ;               Reorganize psects
0000    46  ;
0000    47  ;       V02-006 REFORMAT        Frederick E. Deen, Jr.      29-Jul-1980
0000    48  ;               This code was reformatted to adhere to RMS standards
0000    49  ;
0000    50  ;--
0000    51  ;
0000    52
```

```
0000     54          .SBTTL  DECLARATIONS
0000     55
0000     56 ;
0000     57 ; INCLUDE FILES:
0000     58 ;
0000     59
0000     60 ;
0000     61 ; MACROS:
0000     62 ;
0000     63
0000     64          $IRBDEF
0000     65          $IFBDEF
0000     66          $RABDEF
0000     67          $IMPDEF
0000     68          $RMSDEF
0000     69
0000     70 ;
0000     71 ; EQUATED SYMBOLS:
0000     72 ;
0000     73
0000     74 ;
0000     75 ; OWN STORAGE:
0000     76 ;
0000     77
```

```
0000     79          .SBTTL  RMS$WAIT - $WAIT ROUTINE
0000     80
0000     81 ;++
0000     82 ; RMS$WAIT - Allow synchronization with the completion of asynchronous requests
0000     83 ;
0000     84 ;  This module performs the following functions:
0000     85 ;
0000     86 ;       1. Checks for valid IRAB (via ISI) and if none, assumes DISCONNECT
0000     87 ;          was done,  therefore leaves STS alone unless it is zero, in
0000     88 ;          which case it is set to RMS$_SUC.
0000     89 ;       2. Checks that the RAB is the same one for last operation,
0000     90 ;          and if not merely declares success if not busy, else if busy sets
0000     91 ;          flag in R4 to indicate d fferent RAB and causes WAIT for same
0000     92 ;          event flag.
0000     93 ;       3. If same RAB, tests STS fo  non-zero, indicating completion.
0000     94 ;          if zero, checks IRAB busy  it and causes WAIT if busy, else returns
0000     95 ;          success.
0000     96 ;       4. If STALL required, exits RMS with STATUS = RMS$_STALL
0000     97 ;
0000     98 ; CALLING SEQUENCE:
0000     99 ;
0000    100 ;       Entered from EXEC as a result of user's calling SYS$WAIT
0000    101 ;       (e.g., by using the $WAIT MACRO)
0000    102 ;
0000    103 ; INPUT PARAMETERS:
0000    104 ;
0000    105 ;       AP      User's argument list addr
0000    106 ;
0000    107 ; IMPLICIT INPUTS:
0000    108 ;
0000    109 ;       The contents of the RAB (STS) and RELATED IRAB.
0000    110 ;
0000    111 ; OUTPU  PARAMETERS:
0000    112 ;
0000    113 ;       R1      destroyed
0000    114 ;       R0      status code
0000    115 ;
0000    116 ; IMPLICIT OUTPUTS:
0000    117 ;
0000    118 ;       The IRAB is updated to reflect the STALL.
0000    119 ;
0000    120 ;       A COMPLETION AST is queued if specified in the original user arglist.
0000    121 ;
0000    122 ; COMPLETION CODES:
0000    123 ;
0000    124 ;       Standard RMS (see functional spec for list) as per the
0000    125 ;       operation waited upon.
0000    126 ;
0000    127 ; SIDE EFFECTS:
0000    128 ;
0000    129 ;       None
0000    130 ;
0000    131 ;--
0000    132
```

```
                        0000   134          $ENTRY  RMS$WAIT
                        0000   135          $TSTPT  WAIT
            FFF7'   30  0006   136          BSBW    RM$RABCHK                       ; valid RAB? (exits if bad)
                        0009   137                                                  ;  R11=IMAGE IMPURE AREA addr
                        0009   138                                                  ;  R9=ISI, R8=RAB addr, R7=MODE
                        0009   139 ;
                        0009   140 ; Get the IRAB addr, check for validity
                        0009   141 ;
                        0009   142
      50    07    D0    0009   143          MOVL    #IMP$L_IRABTBL/4,R0             ; IRAB table offset/4
          FFF1'   30    000C   144          BSBW    RM$GTIADR                       ; get IRAB addr
             45   13    000F   145          BEQL    DISCONNECT                      ; branch if bad (assume DISCONNECT)
           00000008    0011   146          .IF     NE $$RMSTEST&$$RMS_TBUGCHK
0A    08 A9    91    0011   147          CMPB    IRB$B_BID(R9),#IRB$C_BID
             52   12    0015   148          BNEQ    ERRBUG                          ; branch if not a valid IRAB
                        0017   149          .ENDC
                        0017   150
                        0017   151 ;
                        0017   152 ; Also check the stored arg list RAB for samen ss
                        0017   153 ;
                        0017   154
             54   D4    0017   155          CLRL    R4                              ; clear flag assuming same RAB
58    24 A9    D1    0019   156          CMPL    IRB$L_LAST_RAB(R9),R8           ; same RAB?
             2C   12    001D   157          BNEQ    DIFRAB                          ; branch if not
                        001F   158
                        001F   159 ;
                        001F   160 ; Same RAB.  If STS has a non-zero value implying operation done, just exit.
                        001F   161 ;
                        001F   162
50    08 A8    D0    001F   163          MOVL    RAB$L_STS(R8),R0                ; pick up status
             41   12    0023   164          BNEQ    EXIT                            ; branch if status indicates done
3A 69    20   E1    0025   165          BBC     #IRB$V_BUSY,(R9),SUCXIT         ; all set if not busy
                        0029   166          ASSUME  IRB$V_ASYNC EQ IFB$V_ASYNC
18 69    23   E1    0029   167 CHKASY: BBC     #IRB$V_ASYNC,(R9),SYNC_STALL    ; branch if operation synch.
                        002D   168          SSB     #IRB$V_ASYNCWAIT,(R9)           ; indicate set event flag req'd
      53    1E   9A    0031   169          MOVZBL  #IMP$C_ASYEFN,R3                ; set EFN to WAIT on
                        0034   170          $CLREF_S        R3                      ; clear it
                        003D   171 SETSTALL:
                        003D   172          RMSSUC  STALL,AP                        ; get set to STALL
          FFBB'   31    0042   173          BRW     RM$ENBAST                       ; go re-enable ASTS and exit
                        0045   174
                        0045   175 ;
                        0045   176 ;  Wait upon a synchronous request (must be from AST level)
                        0045   177 ;
                        0045   178
                        0045   179 SYNC_STALL:
                        0045   180          ASSUME  IFB$B_EFN EQ IRB$B_EFN
53    0B A9    9A    0045   181          MOVZBL  IRB$B_EFN(R9),R3                ; set EFN to WAIT on
             F2   11    0049   182          BRB     SETSTALL
```

```
                          004B   184
                          004B   185 ;++
                          004B   186 ;
                          004B   187 ;  RAB for $WAIT and original operation are different.
                          004B   188 ;  All set if IRAB not busy, else clear STS and cause WAIT.
                          004B   189 ;
                          004B   190 ;--
                          004B   191
    14 69    20    E1     004B   192 DIFRAB: BBC     #IRB$V_BUSY,(R9),SUCXIT         ; branch if not busy
             54    D6     004F   193         INCL    R4                              ; set flag for different RABS
          08 A8    D4     0051   194         CLRL    RAB$L_STS(R8)                   ; zero STS for WAIT
             D3    11     0054   195         BRB     CHKASY                          ; go check SYNC/ASYNC
                          0056   196                                                 ;  operation & STALL
                          0056   197
                          0056   198 ;++
                          0056   199 ;
                          0056   200 ;  Bad ISI value.  Assume DISCONNECT has occurred.
                          0056   201 ;  Leave STS unchanged unless it is zero, in which case declare success.
                          0056   202 ;
                          0056   203 ;--
                          0056   204
                          0056   205 DISCONNECT:
      50    08 A8    D0     0056   206         MOVL    RAB$L_STS(R8),R0                ; get status
             07    13     005A   207         BEQL    SUCXIT                          ; branch if zero
  8001 8F    50    B1     005C   208         CMPW    R0,#RMS$_STALL&^XFFFF           ; did user set a weird
                          0061   209                                                 ;  value in here?
             03    12     0061   210         BNEQ    EXIT                            ; branch if not (o.k.)
                          0063   211 SUCXIT: RMSSUC                                  ; show normal success
          FF97'    31     0066   212 EXIT:   BRW     RMSEX_NOSTR                     ; no need to WAIT (already done)
                          0069   213
                          0069   214 ;++
                          0069   215 ;
                          0069   216 ;  Internal RMS problem  -  IRAB table pointed to an INVALID IRAB
                          0069   217 ;
                          0069   218 ;--
                          0069   219
                          0069   220 ERRBUG: RMSTBUG FTL$_BADIFAB
                          0070   221
                          0070   222
                          0070   223         .END
```

```
$$.PSECT_EP                     = 00000000
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
CHKAST                            00000029 R      01
DIFRAB                            0000004B R      01
DISCONNECT                        00000056 R      01
ERRBUG                            00000069 R      01
EXIT                              00000066 R      01
FTL$_BADIFAB                    = FFFFFFFD
IFB$B_EFN                       = 0000000B
IFB$V_ASYNC                     = 00000023
IMP$C_ASYEFN                    = 0000001E
IMP$L_IRABTBL                   = 0000001C
IRB$B_BID                       = 00000008
IRB$B_EFN                       = 0000000B
IRB$C_BID                       = 0000000A
IRB$L_LAST_RAB                  = 00000024
IRB$V_ASYNC                     = 00000023
IRB$V_ASYNCWAIT                 = 00000024
IRB$V_BUSY                      = 00000020
PIO$A_TRACE                       ********    X   01
RAB$L_STS                       = 00000008
RM$BUG                            ********    X   01
RM$ENBAST                         ********    X   01
RM$EX_NOSTR                       ********    X   01
RM$GTIADR                         ********    X   01
RM$RABCHK                         ********    X   01
RMS$WAIT                        = FFFFFFFE RG   01
RMS$_STALL                      = 00018001
SETSTALL                          0000003D R      01
SUCXIT                            00000063 R      01
SYNC_STALL                        00000045 R      01
SYS$CLREF                         ********  GX    01
TPT$L_WAIT                        ********    X   01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RM$RMS	00000070 (112.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
ABS	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.08	00:00:01.08
Command processing	116	00:00:00.78	00:00:04.89
Pass 1	261	00:00:07.42	00:00:20.83
Symbol table sort	0	00:00:01.00	00:00:01.94

Pass 2	53	00:00:01.32	00:00:03.76
Symbol table output	5	00:00:00.06	00:00:00.06
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	467	00:00:10.68	00:00:32.58

The working set limit was 1200 pages.
39866 bytes (78 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 794 non-local and 1 local symbols.
223 source lines were read in Pass 1, producing 13 object records in Pass 2.
23 pages of virtual memory were used to define 22 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	12
_255DUA28:[SYS.OBJ]LIB.MLB;1	1
_255DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	18

926 GETS were required to define 18 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:RMSOWAIT/OBJ=OBJ$:RMSOWAIT MSRC$:RMSOWAIT/UPDATE=(ENH$:RMSOWAIT)+EXECML$/LIB+LIB$:RMS/LIB

RMS0TRUNC
LIS

STAPRFLNM
LIS

RPGCVTPTO
LIS

RPGHANDLE
LIS

RPGMOVE1
LIS

RPGLIB
REQ

RMSGBL
LIS

RPGRTL

RPGDSPLY
LIS

RPGRTL
MAP

RPGPROLOG
REQ

RPGEXTIND
LIS

RPGLIB
LIS

RMS0SRCH
LIS

RMS0WAIT
LIS

RPGBTZ
LIS

RPGMOVE2
LIS

RPGDIVIDE
LIS

RPGIOEXCE
LIS

RPGDEF
REQ

RMS0UPDAT
LIS

RPGERROR
LIS