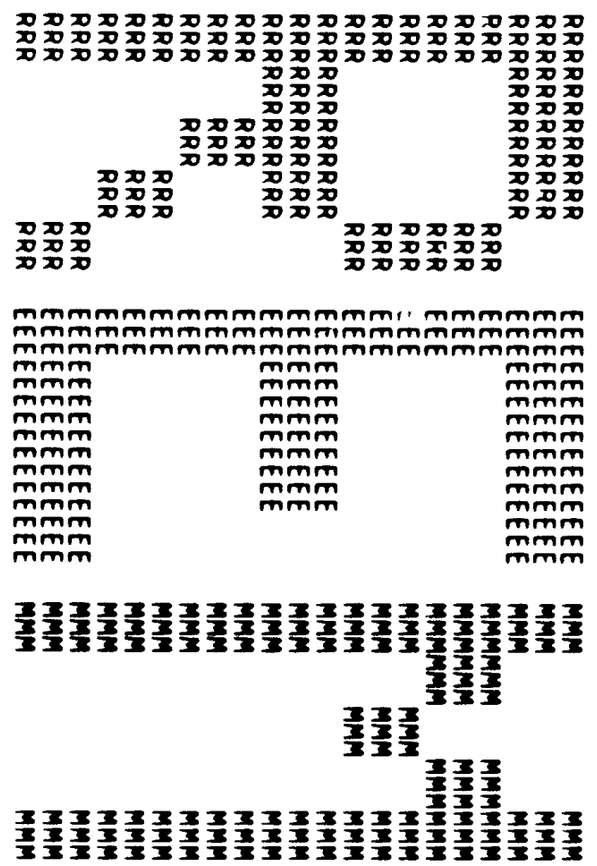

FIG 2

```
RRRRRRRR    EEEEEEEEEE  MM      MM    IIIIII    NN      NN    IIIIII
RRRRRRRR    EEEEEEEEEE  MM      MM    IIIIII    NN      NN    IIIIII
RR      RR  EE          MMMM  MMMM      II      NN      NN      II
RR      RR  EE          MMMM  MMMM      II      NN      NN      II
RR      RR  EE          MM  MM  MM      II      NNNN    NN      II
RR      RR  EE          MM  MM  MM      II      NNNN    NN      II
RRRRRRRR    EEEEEEEE    MM      MM      II      NN  NN  NN      II
RRRRRRRR    EEEEEEEE    MM      MM      II      NN  NN  NN      II
RR  RR      EE          MM      MM      II      NN    NNNN      II
RR  RR      EE          MM      MM      II      NN    NNNN      II
RR    RR    EE          MM      MM      II      NN      NN      II          ....
RR    RR    EE          MM      MM      II      NN      NN      II          ....
RR      RR  EEEEEEEEEE  MM      MM    IIIIII    NN      NN    IIIIII        ....
RR      RR  EEEEEEEEEE  MM      MM    IIIIII    NN      NN    IIIIII        ....

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
```

```
0000     1          .TITLE  REMINI
0000     2          .IDENT  'V04-000'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27
0000    28 ;++
0000    29 ; FACILITY:     REMOTE I/O ACP
0000    30 ;
0000    31 ; ABSTRACT:
0000    32 ;       THIS MODULE PERFORMS INITIALIZATION FOR THE ACP.
0000    33 ;
0000    34 ; ENVIRONMENT:
0000    35 ;       MODE = KERNEL
0000    36 ;--
```

```
0000     38
0000     39 ; AUTHOR:       SCOTT G. DAVIS, CREATION DATE: 06-JUL-79
0000     40 ;
0000     41 ; MODIFIED BY:
0000     42 ;
0000     43 ;       V03-002 CWH52380                CW Hobbs        18-Jan-1983
0000     44 ;               Zero out the newly allocated VCB to prevent access violations
0000     45 ;               on random data.
0000     46 ;
0000     47 ;       V03-001 KDM0002         Kathleen D. Morse       28-Jun-1982
0000     48 ;               Added $DYNDEF.
0000     49 ;
```

```
                                0000     51
                                0000     52 ;
                                0000     53 ; INCLUDE FILES:
                                0000     54 ;
                                0000     55         $AQBDEF
                                0000     56         $DYNDEF
                                0000     57         $PCBDEF
                                0000     58         $REMDEF
                                0000     59         $UCBDEF
                                0000     60         $VCBDEF
                                0000     61
                                0000     62 ;
                                0000     63 ; MACROS:
                                0000     64 ;
                                0000     65
                                0000     66 ;
                                0000     67 ; EQUATED SYMBOLS:
                                0000     68 ;
                                0000     69
                                0000     70 ;
                                0000     71 ; OWN STORAGE:
                                0000     72 ;
                            00000000     73         .PSECT  REM_PURE,NOWRT,NOEXE
                                0000     74
                                0000     75 LKWSET_ADDR:                         ; DESCRIPTOR FOR LOCKING IODONE IN WSET
                    00000000'   0000     76         .LONG   START_LOCK           ; START ADDRESS
                    00000000'   0004     77         .LONG   END_LOCK             ; END ADDRESS
                                0008     78
          54 52 00000010'010E0000'  0008 79 DEV_DESC:       .ASCID  /RT/          ; For device name descriptor
                                0012     80
```

```
                        0012    82
                  00000000      83          .PSECT  REM_INITIALIZE,NOWRT
                        0000    84 ;++
                        0000    85 ;
                        0000    86 ; REM$INITIALIZE - INITIALIZE THE VIRTUAL I/O ACP
                        0000    87 ; FUNCTIONAL DESCRIPTION:
                        0000    88 ;
                        0000    89 ;       This module does the following initialization:
                        0000    90 ;
                        0000    91 ;       1. Allocates an AQB so REMACP can get IRP's from drivers
                        0000    92 ;       2. "Mounts" all virtual devices - creates VCB's, wires the
                        0000    93 ;          AQB to the VCB, and wires the VCB's to the template UCB
                        0000    94 ;       3. Makes up tables describing the remote devices and the related
                        0000    95 ;          object types (for DECnet).
                        0000    96 ;       Note that there is expected to be a template UCB for each
                        0000    97 ;       virtual/remote device type which will be cloned whenever one is needed.
                        0000    98 ;
                        0000    99 ;--
                        0000   100
                  0000  0000   101 REM$INITIALIZE:: .WORD  0                   ; ACP entry point
                        0002   102          $LKWSET_S       W^LKWSET_ADDR      ; Lock IODONE into working set
                        0011   103          $CMKRNL_S       B^STARTUP          ; GO TO KERNEL MODE FOREVER
                     00 001D   104          HALT                               ; SHOULD NEVER GET HERE!!!!!!!!!
                        001E   105
                        001E   106
                  0000  001E   107 STARTUP:       .WORD   0                   ; ENTRY POINT
              6D  016D'CF  DE   0020   108          MOVAL   W^FATAL,(FP)       ; SET UP FOR EXCEPTIONS
                        0025   109
0000'CF  00000000'GF  3C   0025   110          MOVZWL  G^SYS$GW_RJOBLIM, -    ; Obtain sysgen parameter for max
                        002E   111                  W^REM$GB_MAXLINKS          ; number of links to support
           50  0000'CF  D0   002E   112          MOVL    W^REM$GB_MAXLINKS,R0   ; Check range for 2 to 255
               02   50  D1   0033   113          CMPL    R0,#2                  ;
                    09  19   0036   114          BLSS    10$                    ; Not enough
       000000FE 8F   50  D1   0038   115          CMPL    R0,#254                ; Too large?
                    05  15   003F   116          BLEQ    20$                    ; Ok
                    10  90   0041   117 10$:     MOVB    #REM$C_MAXLINKS,-      ; Use constant
                0000'CF         0043   118                  W^REM$GB_MAXLINKS      ; thats wired in
                0000'CF  96     0046   119 20$:     INCB    W^REM$GB_MAXLINKS      ; Make it the number of terminal links
                        004A   120
                        004A   121 ;
                        004A   122 ; BUILD THE ACP QUEUE BLOCK(AQB)
                        004A   123 ;
               51   1C  9A   004A   124          MOVZBL  #AQB$C_LENGTH,R1       ; Length of AQB
         00000000'GF   16   004D   125          JSB     G^EXE$ALONONPAGED      ; Get a chunk of storage
              0D 50   E8   0053   126          BLBS    R0,30$                 ; If LBS successful allocation
                        0056   127          $EXIT_S #SS$_INSFMEM           ; Exit with status
                        0063   128 30$:
                        0063   129
                        0063   130 ;
                        0063   131 ; FILL IN THE AQB
                        0063   132 ;
            08 A2   1C  B0   0063   133          MOVW    #AQB$C_LENGTH,AQB$W_SIZE(R2) ; Record size of AQB
            0A A2   03  90   0067   134          MOVB    S^#DYN$C_AQB,AQB$B_TYPE(R2) ; Note type of block
               62   52  D0   006B   135          MOVL    R2,AQB$L_ACPQFL(R2)    ; Set queue forward link
            04 A2   52  D0   006E   136          MOVL    R2,AQB$L_ACPQBL(R2)    ; Set queue back link
                 0B A2  94   0072   137          CLRB    AQB$B_MNTCNT(R2)       ; Initialize mount count
                        0075   138 ;
```

```
                                0075   139 ; LINK THE AQB INTO THE AQB LIST
                                0075   140 ;
                                0075   141
                    FF88'   30  0075   142          BSBW    REM$LINK_AQB            ; Link in the AQB
                15 A8    05     90  0078   143          MOVB    #AQB$K_REM,AQB$B_ACPTYPE(R8) ; Mark the ACP type
              0000'CF    58     D0  007C   144          MOVL    P8,W^REM$GL_Q_HEAD      ; Save the AQB header address
                                0081   145 ;
                                0081   146 ; Determine driver information
                                0081   147 ;
              51  0008'CF    7E  0081   148          MOVAQ   W^DEV_DESC,R1           ; Set up device name descriptor
                    FF77'   30  0086   149          BSBW    REM$FIND_UCB            ; Find the associated UCB
                  3A 50     E9  0089   150          BLBC    R0,40$ ;GO_AWAY         ; If LBC device not found
              0000'CF    51     D0  008C   151          MOVL    R1,W^REM$GL_TEMPLATE    ; Save the UCB template address
                 55      51     D0  0091   152          MOVL    R1,R5                   ; Set up to allocate VCB
                   009C   30     0094   153          BSBW    REM$ALLOC_VCB           ; Allocate a VCB for this device
                  2C 50   E9     0097   154          BLBC    R0,40$ ;GO_AWAY         ; If LBC error
                                009A   155
                                009A   156 ;
                                009A   157 ; Obtain the space for all the vectors and build the pointers
                                009A   158 ; to them.
                                009A   159 ;
        54   00000000'EF    9A  009A   160          MOVZBL  REM$GB_MAXLINKS, R4     ; The maximum links
             0000'CF    54     7A  00A1   161          EMUL    R4, W^REM$GL_VECSIZE,-  ; Obtain the total size of vectors
        55    000001FF   8F     00A6   162                  #511, R5                ; rounded up by a page
        55    00000200   8F  C6  00AC   163          DIVL2   #512, R5                ; Make number of pages
                     55     D6  00B3   164          INCL    R5                      ; Just to make sure
                                00B5   165          $EXPREG_S -                     ; Obtain the space
                                00B5   166                  PAGCNT = R5,-           ; Number of pages
                                00B5   167                  RETADR = W^REM$GL_UCBVEC ; Return address here
                  67 50     E9  00C6   168 40$:     BLBC    R0, GO_AWAY             ; Not available or something
              53  0000'CF    D0  00C9   169          MOVL    W^REM$GL_UCBVEC, R3     ; Obtain address of space
              52  0000'CF    DE  00CE   170          MOVAL   W^REM$GT_VECTBL, R2     ; Control vector
                00 B2   53  D0  00D3   171 50$:     MOVL    R3, @(R2)               ; Store the address in the pointer
                92   04 A2  C2  00D7   172          SUBL2   4(R2), @(R2)+           ; Back up by the width of entry
                51   54 82  C5  00DB   173          MULL3   (R2)+, R4, R1           ; Make size of this vector
                     53 51  C0  00DF   174          ADDL2   R1, R3                  ; Point beyond this vector
                        62  D5  00E2   175          TSTL    (R2)                    ; End of table
                        ED  12  00E4   176          BNEQ    50$                     ; Nope
              0000'CF   53  02  C3  00E6   177          SUBL3   #2, R3,-                ; Channel was last.  Point this to
                                00EC   178                  W^REM$GL_REJ_CHAN       ; Reject channel address
                                00EC   179
                                00EC   180
                                00EC   181
                                00EC   182 ;
                                00EC   183 ; Allocate space for receive buffers
                                00EC   184 ;
                                00EC   185
                                00EC   186 ;       R4 = maxlinks
                                00EC   187
           52   00000000'GF    3C  00EC   188          MOVZWL  G^IOC$GW_MAXBUF,R2      ; Max I/O size
        55 000001FF 8F  54  52  7A  00F3   189          EMUL    R2,R4,#511,R5           ; Space needed in bytes
           55   00000200 8F  C6  00FC   190          DIVL2   #512,R5                 ; This is the PAGCNT
                     55     D6  0103   191          INCL    R5                      ; Round up, just in case
                53  0000'CF    D0  0105   192          MOVL    W^REM$GL_RBUFVEC,R3     ; This is where to put the count
                        83  D5  010A   193          TSTL    (R3)+                   ; Advance to the first real spot
                                010C   194          $EXPREG_S       PAGCNT= R5-     ; No. of pages
                                010C   195                          RETADR= (R3)    ; Where to return (2 longwords needed)
```

```
                12 50   E9  011B  196          BLBC    R0,GO_AWAY              ; If LBC error - evaporate
                   04   11  011E  197          BRB     70$                     ; Go into loop to set up address vector
                        0120  198 60$:
63   52   83    C1  0120  199          ADDL3   (R3)+,R2,(R3)           ; Compute next address
                        0124  200 70$:
                F9 54   F5  0124  201          SOBGTR  R4,60$                  ; Loop
                        0127  202 ;
                        0127  203 ; Initialize the transport mechanism, e.g., DECnet
                        0127  204 ;
                 FED6'  30  0127  205          BSBW    REM$XPORT_START         ; Do whatever is necessary
                03 50   E9  012A  206          BLBC    R0,GO_AWAY              ; If LBC couldn't get going
                        012D  207 ;
                        012D  208 ; NOW TRY TO PERFORM SOME OPERATION
                        012D  209 ;
                 FED0'  31  012D  210          BRW     REM$MAIN                ; TRY TO DEQUEUE A REQUEST - WILL HIBERNATE
                        0130  211 GO_AWAY:
                 FECD'  31  0130  212          BRW     REM$GO_AWAY             ; Clean up everything and say goodbye
```

```
                                0133   214 ;++
                                0133   215 ;
                                0133   216 ; REM$ALLOC_VCB - This routine allocates a VCB and hooks everything together
                                0133   217 ;
                                0133   218 ; INPUTS:
                                0133   219 ;
                                0133   220 ;       R5 - UCB address
                                0133   221 ;
                                0133   222 ; OUTPUTS:
                                0133   223 ;
                                0133   224 ;       R0 - LBC => error; LBS => OK
                                0133   225 ;
                                0133   226 ; SIDE EFFECTS
                                0133   227 ;
                                0133   228 ;       VCB is hooked to UCB
                                0133   229 ;       AQB is hooked to VCB
                                0133   230 ;       AQB mount count is set to 1
                                0133   231 ;
                                0133   232 ;--
                                0133   233
                                0133   234 REM$ALLOC_VCB::
                                0133   235 ;
                                0133   236 ; Now allocate a VCB and mark the device mounted
                                0133   237 ;
          51    EC 8F    9A   0133   238         MOVZBL  #VCB$C_LENGTH,R1        ; Get length of block
     00000000'GF    16   0137   239         JSB     G^EXE$ALONONPAGED       ; Allocate the storage
           2C 50    E9   013D   240         BLBC    R0,10$                  ; If LBC couldn't allocate
              3F    BB   0140   241         PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Protect registers from the movc5
62 00EC 8F    00   6E    00   2C   0142   242         MOVC5   #0, (SP), #0, #VCB$C_LENGTH, (R2) ; Zero the newly allocated VCB
              3F    BA   014A   243         POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore the registers
        0A A2    11   90   014C   244         MOVB    S^#DYN$C_VCB,VCB$B_TYPE(R2) ; Set structure type
     08 A2    EC 8F    9B   0150   245         MOVZBW  #VCB$C_LENGTH,VCB$W_SIZE(R2) ; Set size of VCB
        0C A2    01   B0   0155   246         MOVW    #1,VCB$W_TRANS(R2)      ; Set the traditional ACP idle count
           4C A2    B4   0159   247         CLRW    VCB$W_MCOUNT(R2)        ; No terminals mounted yet.
50    0000'CF    D0   015C   248         MOVL    W^REM$GL_Q_HEAD,R0      ; Get AQB address
           0B A0    96   0161   249         INCB    AQB$B_MNTCNT(R0)        ; Bump the mount count
        10 A2    80   9E   0164   250         MOVAB   (R0)+,VCB$L_AQB(R2)     ; Link AQB to VCB and set success
        34 A5    52   D0   0168   251         MOVL    R2,UCB$L_VCB(R5)        ; Link VCB to UCB
                      016C   252 10$:
                 05   016C   253         RSB                             ; Done
```

```
                016D    255
                016D    256 ;
                016D    257 ; COME HERE IF THERE ANY EXCEPTIONS
                016D    258 ;
0000            016D    259 FATAL:  .WORD   0
                016F    260         BUG_CHECK       FATALEXCPT,FATAL
                0173    261
                0173    262
                0173    263         .END    REM$INITIALIZE
```

```
$$T1                            = 00000000
AQB$B_ACPTYPE                   = 00000015
AQB$B_MNTCNT                    = 0000000B
AQB$B_TYPE                      = 0000000A
AQB$C_LENGTH                    = 0000001C
AQB$K_REM                       = 00000005
AQB$L_ACPQBL                    = 00000004
AQB$L_ACPQFL                    = 00000000
AQB$W_SIZE                      = 00000008
BUG$_FATALEXCPT                   ********    X   03
DEV_DESC                          00000008 R      02
DYN$C_AQB                       = 00000003
DYN$C_VCB                       = 00000011
END_LOCK                          ********    X   02
EXE$ALONONPAGED                   ********    X   03
FATAL                             0000016D R      03
GO_AWAY                           00000130 R      03
IOC$GW_MAXBUF                     ********    X   03
LKWSET_ADDR                       00000000 R      02
REM$ALLOC_VCB                     00000133 RG     03
REM$C_CURECO                    = 00000001
REM$C_CURVRS                    = 00000001
REM$C_LNK_READ                  = 00000002
REM$C_MAXDEVS                   = 0000000A
REM$C_MAXLINKS                  = 00000010
REM$C_MAXUNITS                  = 00000010
REM$C_MBX_READ                  = 00000001
REM$C_ST_ATTRIB                 = 00000002
REM$C_ST_CONFIG                 = 00000001
REM$FIND_UCB                      ********    X   03
REM$GB_MAXLINKS                   ********    X   03
REM$GL_Q_HEAD                     ********    X   03
REM$GL_RBUFVEC                    ********    X   03
REM$GL_REJ_CHAN                   ********    X   03
REM$GL_TEMPLATE                   ********    X   03
REM$GL_UCBVEC                     ********    X   03
REM$GL_VECSIZE                    ********    X   03
REM$GO_AWAY                       ********    X   03
REM$GT_VECTBL                     ********    X   03
REM$INITIALIZE                    00000000 RG     03
REM$LINK_AQB                      ********    X   03
REM$MAIN                          ********    X   03
REM$XPORT_START                   ********    X   03
SS$_INSFMEM                       ********    X   03
STARTUP                           0000001E R      03
START_LOCK                        ********    X   02
SYS$CMKRNL                        ********   GX   03
SYS$EXIT                          ********   GX   03
SYS$EXPREG                        ********   GX   03
SYS$GW_RJOBLIM                    ********    X   03
SYS$LKWSET                        ********   GX   03
UCB$L_VCB                       = 00000034
VCB$B_TYPE                      = 0000000A
VCB$C_LENGTH                    = 000000EC
VCB$L_AQB                       = 00000010
VCB$W_MCOUNT                    = 0000004C
VCB$W_SIZE                      = 00000008
VCB$W_TRANS                     = 0000000C
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
REM_PURE	00000012 (18.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC BYTE
REM_INITIALIZE	00000173 (371.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	43	00:00:00.07	00:00:00.52
Command processing	173	00:00:00.72	00:00:02.38
Pass 1	280	00:00:07.98	00:00:17.73
Symbol table sort	0	00:00:01.26	00:00:02.37
Pass 2	66	00:00:01.52	00:00:03.79
Symbol table output	8	00:00:00.08	00:00:00.08
Psect synopsis output	5	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	578	00:00:11.66	00:00:26.91

The working set limit was 1200 pages.
44377 bytes (87 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 877 non-local and 8 local symbols.
263 source lines were read in Pass 1, producing 16 object records in Pass 2.
20 pages of virtual memory were used to define 19 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[REM.OBJ]REM.MLB;1	1
_255DUA28:[SYS.OBJ]LIB.MLB;1	6
_255DUA28:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	16

977 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:REMINI/OBJ=OBJ$:REMINI MSRC$:REMINI/UPDATE=(ENH$:REMINI)+EXECML$/LIB+LIB$:REM/LIB
```

REMLOCKDB
LIS

REMQIO
LIS

RMS

REMINI
LIS

RMS
MAP

REMPOST
LIS

REMGETIRP
LIS

REM

REMMACROS
MAR

REMACP
MAP

REMXPORTD
LIS

STOPREM
LIS

DAPDEF
MDL

REMACPDAT
LIS

REMPROTCL
LIS

REMSUBR
LIS

TRANSFER
LIS

STOPREM
MAP