VAX/VMS PL/M-86 V1.0 COMPILATION OF MODULE VOUT
OBJECT MODULE PLACED IN VOUT.OBJ
COMPILER INVOKED BY: PLM86 VOUT.PLM OPTIMIZE(3) DEBUG XREF


```
                $title('VOUT.RSP - virtual console disk write')
                $set(debug=0)
                $compact
  1             vout:
                do;

                /* Disk output process.  Reads Virtual OUTput Queue (VOUTQ) associated
                with a virtual console in buffered background mode.  Output is spooled
                to the file VOUTx.$$$.  When console is in foreground purge mode, spooled
                output is read from this file and dumped on the screen.  There is one
                copy of the VOUT process per virtual console.  Each VOUT RSP has
                its own data area, but the code is reentrant for all the VOUT RSPs.
                */

                /* VAX commands used to generate VOUT.RSP

                asm86 rvout.a86
                plm86 vout.plm optimize(3) debug 'p1' 'p2' 'p3'
                link86 rvout.obj, pxios.obj, vout.obj to vout.lnk
                loc86 vout.lnk od(sm(code,dats,data,const,stack)) -
                  ad(sm(code(0))) ss(stack(0))
                h86 vout.dat
                refmt vout.mp2 vout.2
                ren vout.2 vout.mp2

                the hex is uploaded to a micro to make a binary file using the command:

                gencmd vout data[bxxx]

                xxx is taken from the VOUT.MP2 file generated on the VAX by LOC86.
                xxx is the next paragraph after the CODE segment.
                */


                $include (:f1:copyrt.lit)

        =       /*
        =         Copyright (C) 1983
        =         Digital Research
        =         P.O. Box 579
        =         Pacific Grove, CA 93950
        =       */

                $include (:f1:comlit.lit)

  2   1 =       declare
        =               lit             literally       'literally',
        =               dcl             lit             'declare',
        =               true            lit             '0ffh',
        =               false           lit             '0',
        =               no              lit             'not',
```

```
        =           boolean             lit             'byte',
        =           forever             lit             'while true',
        =           cr                  lit             '13',
        =           lf                  lit             '10',
        =           tab                 lit             '9';

            $include (:f1:qd.lit)

        =   /* Queue Descriptor */

 3  1   =   dcl qnamsiz lit '8';

 4  1   =   dcl qd$structure lit 'structure(
        =     link  word,
        =     net byte,
        =     org byte,
        =     flags word,
        =     name(qnamsiz) byte,
        =     msglen word,
        =     nmsgs word,
        =     dq word,
        =     nq word,
        =     msgcnt word,
        =     msgout word,
        =     buffer word)';

        =   /* queue flag values */

 5  1   =   dcl qf$mx       lit '001h'; /* Mutual Exclusion */
 6  1   =   dcl qf$keep     lit '002h'; /* NO DELETE        */
 7  1   =   dcl qf$hide     lit '004h'; /* Not User writable    */
 8  1   =   dcl qf$rsp      lit '008h'; /* rsp queue        */
 9  1   =   dcl qf$table    lit '010h'; /* from qd table    */
10  1   =   dcl qf$rpl      lit '020h'; /* rpl queue        */
11  1   =   dcl qf$dev      lit '040h'; /* device queue     */

        =   /* Queue Parameter Block */

12  1   =   dcl qpb$structure lit 'structure(
        =     flgs    byte,
        =     net     byte,
        =     qaddr   word,
        =     nmsgs   word,
        =     buffptr word,
        =     name (qnamsiz) byte )';

            $include (:f1:mfunc.lit)

        =   /* Concurrent CP/M function numbers */

13  1   =   dcl           m$prtbuf            lit       '9',
        =                 m$select            lit       '14',
        =                 m$openf             lit       '15',
        =                 m$closef            lit       '16',
        =                 m$deletef           lit       '19',
        =                 m$readf             lit       '20',
        =                 m$writef            lit       '21',
```

```
        =                   m$makef             lit     '22',
        =                   m$getlogin          lit     '24',
        =                   m$curdsk            lit     '25',
        =                   m$setdma            lit     '26',
        =                   m$setatt            lit     '30',
        =                   m$setusr            lit     '32',
        =                   m$readrf            lit     '33',
        =                   m$writerf           lit     '34',
        =                   m$resetdrv          lit     '37',
        =                   m$errmode           lit     '45',
        =                   m$dirbios           lit     '50',
        =                   m$makeq             lit     '134',
        =                   m$openq             lit     '135',
        =                   m$deleteq           lit     '136',
        =                   m$readq             lit     '137',
        =                   m$creadq            lit     '138',
        =                   m$writeq            lit     '139',
        =                   m$cwriteq           lit     '140',
        =                   m$delay             lit     '141',
        =                   m$dispatch          lit     '142',
        =                   m$setprior          lit     '145',
        =                   m$attach            lit     '146',
        =                   m$detach            lit     '147',
        =                   m$setcns            lit     '148',
        =                   m$parse             lit     '152',
        =                   m$getcns            lit     '153',
        =                   m$sysdat            lit     '154',
        =                   m$getpd             lit     '156',
        =                   m$abort             lit     '157';

        =   /* Internal calls */

14  1   =   dcl             mi$sleep            lit     '0212H',
        =                   mi$wakeup           lit     '0213H';
        =
            $include (:f1:mxfunc.lit)

        =   /* MP/M-86 XIOS function numbers */

15  1   =   dcl     mx$conin            lit '1',
        =           mx$conout           lit '2',
        =           mx$lstout           lit '4',
        =           mx$switch           lit '7',
        =           mx$upstatus         lit '8';

            $include (:f1:fcb.lit)

16  1   =   declare
        =       f$drvusr        lit '0',    /* drive/user byte              */
        =       f$name          lit '1',    /* file name                    */
        =       f$namelen       lit '8',    /* file name length             */
        =       f$type          lit '9',    /* file type field              */
        =       f$typelen       lit '3',    /* type length                  */
        =       f$rw            lit '9',    /* high bit is R/W attribute    */
        =       f$dirsys        lit '10',   /* high bit is dir/sys attribute */
        =       f$arc           lit '11',   /* high bit is archive attribute */
        =       f$ex            lit '12',   /* extent                       */
```

```
        =       f$s1            lit '13',       /* module byte                  */
        =       f$rc            lit '15',       /* record count                 */
        =       f$diskmap       lit '16',       /* file disk map                */
        =       diskmaplen      lit '16',       /* disk map length              */
        =       f$drvusr2       lit '16',       /* fcb2                         */
        =       f$name2         lit '17',
        =       f$type2         lit '25',
        =       f$cr            lit '32',       /* current record               */
        =       f$rrec          lit '33',       /* random record                */
        =       f$rreco         lit '35';       /*   "      "    overflow        */


 17   1      dcl name$len lit '4';           /* number of letters in RSP name: 'VOUT' */
 18   1      dcl fcblen lit '36';

 19   1      dcl rsplink word external;  /* set to SYSDAT by O.S. initialization */
 20   1      dcl udaseg word external;       /* DS for this process */
 21   1      dcl ncopies byte external;
 22   1      dcl copynum byte at (.ncopies); /* VOUT process copy number, also the */
                                             /* virtual console number for console */
                                             /* output to the XIOS */

        $include (:f1:sd.lit)

        =   /* System Data Page */

 23   1  =     dcl sysdat$pointer pointer;
 24   1  =     dcl sysdat$ptr structure(
        =       offset word,
        =       segment word) at (@sysdat$pointer);
 25   1  =     declare sd based sysdat$pointer structure (
        =           supmod (4) word,
        =       /*  rtmmod (4) word,
        =           memmod (4) word,
        =           ciomod (4) word,
        =           bdosmod (4) word,
        =           xiosmod (4) word,
        =           netmod (4) word,
        =           reservd (4) word */
        =           space(28) word,
        =           mpmseg word,
        =           rspseg word,
        =           endseg word,
        =           module$map byte,
        =           ncns byte,
        =           nlst byte,
        =           nccb byte,
        =           nflags byte,
        =           srchdisk byte,
        =           mmp word,
        =           nslaves byte,
        =           dayfile byte,
        =           tempdisk byte,
        =           tickspersec byte,
        =           lul word,
        =           ccb word,
        =           flags word,
```

```
=           mdul word,
=           mfl word,
=           pul word,
=           qul word,
=           qmau (4) word,
=           rlr word,
=           dlr word,
=           drl word,
=           plr word,
=           slr word,
=           thrdrt word,
=           qlr word,
=           mal word,
=           version word,
=           vernum word,
=           mpmvernum word,
=           tod (2) word,
=           tod_sec byte,
=           ncondev byte,
=           nlstdev byte,
=           nciodev byte,
=           lcb (2) word,
=           lckmax byte,
=           opmax byte,
=           sysltot (2) word,
=           cmod byte );
=
=
26  1  =    declare sd$byte based sysdat$pointer (1) byte;


27  1       dcl ccb$pointer pointer;
28  1       dcl ccb$ptr structure ( offset address, segment address) at
              (@ccb$pointer);
         $include (:f1:vccb.lit)
=            /* Concurrent CP/M Character Control Block Structure */

=        /*              +---------+---------+---------+---------+
=              00        |      attach       |       queue       |
=                        +---------+---------+---------+---------+
=              04        |  flag   | startcol| column  |  nchar  |
=                        +---------+---------+---------+---------+
=              08        |  mimic  | msource |   pc    |   vc    |
=                        +---------+---------+---------+---------+
=              0C        |  btmp   |  rsvd   |       state       |
=                        +---------+---------+---------+---------+
=              10        |     maxbufsiz     |       vinq        |
=                        +---------+---------+---------+---------+
=              14        |       voutq       |       vcmxq       |
=                        +---------+---------+---------+---------+
=              18        | qpbflgs | qpbfill |      qpbqaddr     |
=                        +---------+---------+---------+---------+
=              1C        |      qpbnmsgs     |    qpbbuffptr     |
=                        +---------+---------+---------+---------+
=              20        |       qbuff       |      cosleep      |
=                        +---------+---------+---------+---------+
=              24        |      usleep       |       vsleep      |
```

```
        =                +---------+---------+---------+---------+
        =         28     |           ... reserved ...            |
        =                +---------+---------+---------+---------+


        =   */

29  1   =   dcl ccb$structure lit 'structure (attach address, queue address,
        =    flag byte, startcol byte, column byte, nchar byte, mimic byte, msource byte,
        =    ccb$tail1';
30  1   =   dcl ccb$tail1 lit
        =    'pc byte, vc byte, btmp byte, rsvd byte, state word, maxbufsiz word,
        =    ccb$tail2';
31  1   =   dcl ccb$tail2 lit
        =    'vinq address, voutq address, vcmxq address,
        =    qpbflags byte, qpbresrvd byte, qpbqaddr address,
        =    qpbnmsgs address, qpbbuffptr address, qbuff address, cosleep word,
        =    usleep word, vsleep word, r1 word, r2 word)';

32  1   =    declare                                  /* flag values                    */
        =      cf$listcp          lit        '001h',  /* control P toggle               */
        =      cf$compc           lit        '002h',  /* suppress output                */
        =      cf$switchs         lit        '004h',  /* XIOS supports switch screening */
        =      cf$conout          lit        '008h',  /* XIOS console output ownership  */
        =      cf$vout            lit        '010h';  /* process writing to VOUTQ       */

        =   /* values of state byte */
        =                                        /* conout goes to XIOS  */

        =   /* state word flags */

33  1   =   dcl
        =   csm$buffered           lit        '0001h',
        =   csm$background         lit        '0002h',
        =   csm$purging            lit        '0004h',
        =   csm$noswitch           lit        '0008h',
        =   csm$suspend            lit        '0010h',
        =   csm$abort              lit        '0020h',
        =   csm$filefull           lit        '0040h',
        =   csm$ctrlS              lit        '0080h',
        =   csm$ctrlQ              lit        '0100h',
        =   csm$ctrlP              lit        '0200h';

34  1   =   dcl x$init$offset lit '0Ch',
        =       x$init$pointer pointer,
        =       x$init$ptr structure (offset word, segment word) at (@x$init$pointer),
        =       x$init based x$init$pointer structure
        =         (tick byte, ticks$sec byte, door byte, resrvd1 (2) byte,
        =         nvcns byte, nccb byte, nlst byte, ccb word, lcb word);


35  1   =   dcl lcb$structure lit 'structure (attach address, queue address,
        =    flag byte, startcol byte, column byte, nchar byte,
        =    mimic byte, msource byte)';
```

```
36  1     dcl ccb based ccb$pointer ccb$structure;

37  1     dcl data$msg lit '0';
38  1     dcl wake$msg lit '0ffh';
39  1     dcl voutq$msg structure (
            dayta byte, type byte);

40  1     mon1: procedure (func,a) external;
41  2       dcl func byte, a address;
42  2     end mon1;

43  1     mon2: procedure (func,a) byte external;
44  2       dcl func byte, a address;
45  2     end mon2;

46  1     mon4: procedure (func,a) pointer external;
47  2       dcl func byte, a address;
48  2     end mon4;

49  1     intsys: procedure (cx, dx, bx) external;     /* internal O.S. functions */
50  2       dcl (cx, dx, bx) word;                     /* see RVOUT module        */
51  2     end intsys;

      /* special disk output assembly module */

52  1     pxios1: procedure (func,p1,p2) external;
53  2       dcl func byte, (p1,p2) address;       /* XIOS interface for process */
54  2     end pxios1;                             /* not in the O.S. */

55  1     dcl ps$ciosleep lit '9';

56  1     sleep: procedure(addr);
57  2       dcl addr word;
58  2       call intsys(mi$sleep, addr, ps$ciosleep);
59  2     end sleep;

60  1     wakeup: procedure(addr);
61  2       dcl addr word;
62  2       call intsys(mi$wakeup, addr, 0);
63  2     end wakeup;


      $if debug=1
                      /* conditionally compiled error print routines */

       print$msg: procedure(endchar, sptr);
        dcl (i, endchar) byte, sptr pointer,
            string based sptr (1) byte;
        i = 0;
        do while string(i) <> endchar;
          call pxios1(mx$conout, string(i), copynum);
          i = i + 1;
        end;
      end print$msg;

      print$hex: procedure (nib);
        dcl nib byte;
```

```
            nib = nib and 0fh;
            if nib < 10 then
              call pxios1(mx$conout, nib + '0', copynum);
            else
              call pxios1(mx$conout, nib + 'A' - 10, copynum);
          end print$hex;

          error: procedure(msgptr);
            dcl msgptr pointer;
            call print$msg(0, @(cr, lf, '**** VOUT ERROR ****  ',0));
            call print$msg(0, msgptr);
            call print$msg(0, @(', CCB.STATE = ', 0));
            call print$hex(shr(ccb.state, 12));
            call print$hex(shr(ccb.state,  8));
            call print$hex(shr(ccb.state,  4));
            call print$hex(ccb.state);
            call print$msg(0, @('H', cr, lf, 0));
          end error;

          $endif

64  1     read$change$mxq: procedure;
65  2       qpb.qaddr = ccb.vcmxq;
66  2       call mon1 (m$readq, .qpb);
67  2     end read$change$mxq;

68  1     write$change$mxq: procedure;
69  2       qpb.qaddr = ccb.vcmxq;
70  2       call mon1 (m$writeq, .qpb);
71  2     end write$change$mxq;

72  1     dcl logeof lit '0ffh';
73  1     dcl dump$op lit '0ffh';

74  1     dcl writing boolean initial (false);
75  1     dcl delete$flag boolean initial (true);      /* delete when convienient     */
76  1     dcl deleted boolean initial (true);          /* has been deleted            */
77  1     dcl file$is$empty boolean initial (true);
78  1     dcl rrr address initial(0);                  /* next random record to read  */
79  1     dcl wrr address initial(0);                  /* next random record to write */

80  1     delete$file: procedure;
81  2       call mon1(m$closef, .fcb);                 /* force allocation vector     */
82  2       call mon1(m$deletef, .fcb);                /* update */
83  2       delete$flag = false;
84  2       deleted = true;
85  2     end delete$file;

86  1     make$file: procedure boolean;
87  2       call setb(0, @fcb(f$ex), fcblen-f$ex);
88  2       fcb(f$drvusr) = sd.tempdisk + 1;    /* try deleting the file in case drive */
89  2       call mon1(m$deletef, .fcb);         /* was read only when delet$file was   */
                                                /* called or tempdisk has changed      */
90  2       if mon2(m$makef, .fcb) = 0ffh then  /* open in locked mode */
91  2         return(false);            /* error - force open attempt next time */
92  2       deleted = false;
93  2       return(true);
```

```
                  /* fcb(f$ex) = fcb(f$ex) or 80h;   /* make system         */
                  /* call mon1(m$setatt, .fcb); */
   94    2        end make$file;

   95    1        reset$file: procedure;
   96    2          delete$flag, file$is$empty = true;
   97    2          writing = false;                  /* force setdma */
   98    2          wrr, rrr = 0;                     /* not necessary ? */
   99    2        end reset$file;

  100    1        dcl bufsiz lit '128';

  101    1        dcl in$buf(bufsiz) byte;          /* buffer to fill on from reading VOUTQ */
  102    1        dcl in$ptr word initial (0ffffh);  /* initially empty buffer */

  103    1        dcl purge$buf (buf$siz) byte;     /* buffer to use when purging */
  104    1        dcl purge$ptr word initial (0ffffh);
  105    1        dcl num$purge$buf$chars word initial (0);

  106    1        write$buf: procedure boolean;
  107    2          if deleted then
  108    2          do;
  109    3            if not make$file then           /* delete and make file */
  110    3              return(false);
  111    3          end;
  112    2          else if rrr = wrr and not file$is$empty then
  113    2            return(false);              /* don't write if we haven't purged it yet */
                    if not writing then       /* we want to be in write mode */
  115    2          do;
  116    3            call mon1(m$setdma,.in$buf);
  117    3            writing = true;
  118    3          end;
  119    2          fcb(f$rrec) = low(wrr);
  120    2          fcb(f$rrec+1) = high(wrr);
  121    2          if mon2(m$writerf, .fcb) <> 0 then
  122    2            return(false);            /* out of disk space or physical error */
  123    2          file$is$empty = false;
  124    2          in$ptr = 0ffffh;
  125    2          wrr = (wrr + 1) mod (ccb.maxbufsiz * 8);   /* next record to write */
  126    2          return(true);
  127    2        end write$buf;

  128    1        read$buf: procedure boolean;
  129    2          dcl ret boolean;
  130    2          if file$is$empty then
  131    2          do;
  132    3            if not deleted then         /* made file but had a write error */
  133    3              call reset$file;
  134    3            return(false);
  135    3          end;
  136    2          if writing then               /* we want to be in read mode */
  137    2          do;
  138    3            call mon1(m$setdma, .purge$buf);
  139    3            writing = false;
  140    3          end;
  141    2          fcb(f$rrec) = low(rrr);
  142    2          fcb(f$rrec+1) = high(rrr);
```

```
143   2       ret = mon2(m$readrf,.fcb) = 0; /* physical error if false - skips record */
144   2       rrr = (rrr + 1) mod (ccb.maxbufsiz * 8);
145   2       if rrr = wrr then             /* done with file  ? */
146   2         call reset$file;
147   2       return(ret);                  /* return read status */
148   2     end read$buf;

149   1     dcl active$msg boolean initial (false);
150   1     read$voutq: procedure;
151   2       if active$msg then
152   2         return;
153   2       qpb.qaddr = ccb.voutq;
154   2       qpb.buffptr = .voutq$msg;
155   2       call mon1(m$readq, .qpb);
156   2       if voutq$msg.type = data$msg then
157   2         active$msg = true;
158   2     end read$voutq;

159   1     drain$voutq: procedure(char$adr) boolean;
160   2       dcl char$adr address;               /* return false if no chars found in */
161   2       dcl char based char$adr byte;       /* VOUTQ, return true and put char @ */
162   2       dcl (have$a$char, qempty) boolean;  /* char$adr if there is one          */
163   2       qpb.qaddr = ccb.voutq;
164   2       qpb.buffptr = .voutq$msg;
165   2       have$a$char, qempty = false;
166   2       do while not have$a$char and not qempty;
167   3         if mon2(m$creadq, .qpb) = 0 then      /* successful queue read */
168   3           have$a$char = voutq$msg.type = data$msg;  /* and msg is data */
                 else
169   3           qempty = true;
170   3       end;
171   2       char = voutq$msg.dayta;
172   2       if qempty then
173   2         return(false);                /* no chars in queue */
174   2       return(true);                   /* char was a data msg */
175   2     end drain$voutq;

176   1     put$char: procedure boolean;
177   2       active$msg = false;
178   2       if voutq$msg.type <> data$msg then
179   2         return(true);
180   2       voutq$msg.type = wake$msg;    /* probably garbage */
181   2       in$buf(in$ptr := in$ptr + 1) = voutq$msg.dayta;
182   2       if in$ptr = buf$siz - 1 then
183   2         return(write$buf);                      /* don't call again no write  */
184   2       return(true);
185   2     end put$char;

186   1     get$char: procedure (charadr) boolean;
187   2       dcl charadr address, char based charadr byte;
188   2       if purge$ptr + 1 = num$purge$buf$chars then
189   2         if read$buf then
190   2         do;
191   3           num$purge$buf$chars = bufsiz;
192   3           purge$ptr = 0ffffh;
193   3         end;
194   2         else if in$ptr <> 0ffffh then   /* data in buff but not in file */
```

```
195   2          do;
196   3            call move(in$ptr + 1, .in$buf, .purge$buf);
197   3            write$pending = false;
198   3            num$purge$buf$chars = in$ptr + 1;
199   3            in$ptr, purge$ptr = 0ffffh;  /* indicate data in purge$buf */
200   3          end;
201   2          else if active$msg then
202   2          do;
203   3            active$msg = false;
204   3            char = voutq$msg.dayta;
205   3            return(true);
206   3          end;
              
              else
207   2          do;
208   3            if not drain$voutq(char$adr) then        /* get chars from VOUTQ       */
209   3            do;
210   4              do while (ccb.flag and cf$vout) <> 0;  /* user process is NQing wait */
211   5                call mon1(m$delay, 2);               /* for q write to finish      */
212   5              end;
213   4              return(drain$voutq(char$adr));         /* now read message, usr proc */
214   4            end;                                     /* sleeps because of state    */
                   else
215   3              return(true);                          /* got a char from VOUTQ      */
216   3           end;
217   2        purge$ptr = purge$ptr + 1;
218   2        char = purge$buf(purge$ptr);

219   2        return (true);
220   2      end get$char;

221   1      full$disk: procedure;                          /* arrive when we can't write*/
222   2        call read$change$mxq;                        /* to the disk               */
223   2        if (ccb.state and csm$purging) = 0 then      /* wait for PIN to switch us */
                                                           /* to the foreground,        */
224   2          ccb.state = ccb.state or csm$filefull;     /* csm$file$full and csm$pur-*/
                                                           /* ging are mutually exclusive*/
225   2        call write$change$mxq;
226   2      end full$disk;

227   1      dcl write$pending boolean initial (false);
228   1      buffer: procedure;
229   2        if write$pending then
230   2          if write$pending := not write$buf then
231   2          do;
232   3            call full$disk;
233   3            return;
234   3          end;
235   2        do while (ccb.state and not double(csm$ctrlP)) =
              csm$buffered + csm$background;
236   3          call read$voutq;                       /* always do something with the */
237   3          if write$pending := not putchar then   /* character ! */
238   3          do;
239   4            call full$disk;
240   4            return;
241   4          end;
242   3        end;
243   2      end buffer;
```

```
244   1     dcl purgeok$mask lit '(csm$background or csm$abort or csm$ctrlS)';

245   1     purge: procedure;
246   2       dcl (char, count) byte;
247   2       dcl controlP boolean;
248   2       dcl more$in$file boolean;
249   2       more$in$file = true;
250   2       do while (ccb.state and purgeok$mask) = 0 and
                more$in$file;
251   3         call read$change$mxq;
252   3         controlP = (ccb.state and csm$ctrlP) <> 0;
253   3         if (ccb.state and purgeok$mask) = 0 then
254   3         do;
255   4           disable;
256   4           do while (ccb.flag and cf$conout) <> 0;
257   5             call sleep(.ccb.cosleep);
258   5           end;
259   4           ccb.flag = ccb.flag or cf$conout;
260   4           enable;
261   4           count = 0;
262   4           do while more$in$file and count < 40;    /* for performance, purge 40 */
263   5             if (more$in$file := get$char(.char)) and  /* chars before allowing  */
                    (ccb.state and csm$ctrlO) = 0 then     /* state to change, 40 is */
264   5                 do;                                /* is somewhat arbitrary  */
265   6                   call pxios1(mx$conout, char, copynum);
266   6                   if controlP then
267   6                     call pxios1(mx$lstout, char, ccb.mimic);
268   6                 end;
269   5             count = count + 1;
270   5           end;
271   4           ccb.flag = ccb.flag and not cf$conout;
272   4           call write$change$mxq;          /* possibly wake up PIN */
273   4           call wakeup(.ccb.cosleep);      /* or user process      */
274   4         end;
                  else
275   3           call write$change$mxq;
276   3       end;
277   2       if not more$in$file then
278   2       do;
279   3         num$purge$buf$chars = 0;
280   3         purge$ptr, inptr = 0ffffh;
281   3         call read$change$mxq;
282   3         if (ccb.state and csm$purging) <> 0 then
283   3         do;
284   4           ccb.state = ccb.state and not double(csm$purging);
285   4           call pxios1(mx$upstatus, 0, 0);
286   4         end;
287   3         call write$change$mxq;
288   3       end;
289   2       call wakeup(.ccb.usleep);  /* wake up user process */
290   2     end purge;

291   1     abort: procedure;
292   2       dcl junk word;
293   2       do while drain$voutq(.junk);       /* drain input queue        */
294   3       end;                               /* may wake up user process */
295   2       call read$change$mxq;
```

```
296   2       ccb.state = ccb.state and not double(csm$abort);
297   2       call write$change$mxq;
298   2       call reset$file;
299   2       write$pending = false;
300   2       purge$ptr, inptr = 0ffffh;
301   2       num$purge$buf$chars = 0;
302   2       call wakeup(.ccb.usleep);  /* wake up user process */
303   2     end abort;

304   1     initq: procedure(qdaddr) address;
305   2       dcl qdaddr address;
306   2       dcl ret boolean;
307   2       dcl iqd based qdaddr qd$structure;
308   2       call move(qnamsiz, .iqd.name, .qpb.name);
309   2       ret = mon2(m$makeq, qdaddr);                  /* 0ffh return = error */
310   2       ret = ret or mon2(m$openq, .qpb);             /* ret = 0 if no error */

        $if debug = 1
          if ret then        /* if debugging print error */
            call error(@('Queue initialization error',0));
        $endif

311   2       return(qpb.qaddr);
312   2     end initq;

313   1     dcl pd$pointer pointer;     /* in RSP assembly interface */
314   1     dcl pd based pd$pointer (1) byte;
315   1     dcl pd$name lit '8';

316   1     dcl voutq$buf (32) byte;
317   1     dcl voutq qd$structure initial
            (0,0,0, qf$hide + qf$keep,        'VOUTQ   ',2,16,0,0,0,0,.voutq$buf);

318   1     dcl vinq$buf (64) byte;          /* 64 bytes type ahead */
319   1     dcl vinq qd$structure initial
            (0,0,0, qf$keep + qf$hide,        'VINQ    ',1,64,0,0,0,0,.vinq$buf);

320   1     dcl vcmxq qd$structure initial
            (0,0,0,qf$keep + qf$mx + qf$hide, 'VCMXQ   ',0,1,0,0,0,0,0,0);

321   1     dcl qpb qpb$structure;

322   1     dcl dummy (1) byte data ('Z');  /* make constant segment non-zero to */
                                            /* hex generation */

323   1     dcl fcb(36) byte initial (0,'          ', '$$$');

        /* initialization */

324   1     plmstart: procedure public;
325   2       dcl save$state word;
326   2       call mon1(m$errmode, 0ffh);           /* don't display errors */
327   2       ccb$ptr.segment, sysdat$ptr.segment = rsplink;
328   2       sysdat$ptr.offset = 0;
329   2       ccb$ptr.offset = sd.ccb * copynum * size(ccb);

330   2       pd$pointer = mon4(m$getpd,0);
```

```
 331   2        call movb(.pd(pd$name), .fcb(f$name), qnamsiz);

 332   2        call move(4, .fcb(f$name + name$len), .vinq.name(4));
 333   2        ccb.vinq = initq(.vinq);
 334   2        call move(3, .fcb(f$name + name$len), .voutq.name(5));
 335   2        ccb.voutq = initq(.voutq);
 336   2        call move(3, .fcb(f$name + name$len), .vcmxq.name(5));
 337   2        ccb.vcmxq = initq(.vcmxq);

 338   2        call mon1(m$setcns, copynum);          /* copynum is virtual console # */

 339   2        fcb(f$drvusr) = sd.tempdisk + 1;
 340   2        call write$change$mxq;                 /* write initial MX message */
 341   2        call mon1(m$setprior, 200);

 342   2        do forever;
 343   3          if delete$flag then
 344   3            call delete$file;
 345   3          if (ccb.state and not double(csm$ctrlP + csm$ctrlO)) =
                     csm$buffered + csm$background then        /* if ctrlO,background and */
 346   3            call buffer;                             /* buffered, then sleep    */
 347   3          else if ( (ccb.state and not double(csm$ctrlO + csm$ctrlP))
                     and csm$purging) <> 0 then
 348   3            call purge;
 349   3          else if (ccb.state and csm$abort) <> 0 then
 350   3            call abort;
                   if delete$flag then
 352   3            call delete$file;
                   else
 353   3            call read$voutq;
 354   3        end;

 355   2      end plmstart;
 356   1      end vout;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
43	0000H	2	A. . . . . WORD PARAMETER 44
40	0000H	2	A. . . . . WORD PARAMETER 41
46	0000H	2	A. . . . . WORD PARAMETER 47
291	0500H	69	ABORT. . . . . PROCEDURE STACK=0010H 350
149	0189H	1	ACTIVEMSG. . . . . BYTE INITIAL 151 157 177 201 203
60	0004H	2	ADDR . . . . . WORD PARAMETER AUTOMATIC 61 62
56	0004H	2	ADDR . . . . . WORD PARAMETER AUTOMATIC 57 58
36	0000H	2	ATTACH . . . . . WORD MEMBER(CCB)
2			BOOLEAN. . . . . LITERALLY 74 75 76 77 86 106 128 129 149 159 162 176 186 227 247 248 306
36	000CH	1	BTMP . . . . . BYTE MEMBER(CCB)
228	037DH	53	BUFFER . . . . . PROCEDURE STACK=0014H 346
307	001AH	2	BUFFER . . . . . WORD MEMBER(IQD)
320	001AH	2	BUFFER . . . . . WORD MEMBER(VCMXQ)
317	001AH	2	BUFFER . . . . . WORD MEMBER(VOUTQ)
319	001AH	2	BUFFER . . . . . WORD MEMBER(VINQ)
321	0006H	2	BUFFPTR. . . . . WORD MEMBER(QPB) 154 164
100			BUFSIZ . . . . . LITERALLY 101 103 182 191
49	0000H	2	BX . . . . . WORD PARAMETER 50
36	0000H	44	CCB. . . . . STRUCTURE BASED(CCBPOINTER) 65 69 125 144 153 163 210 223 224 235 250 252 253 256 257 259 263 267 271 273 282 284 289 296 302 329 333 335 337 345 347 349
25	0054H	2	CCB. . . . . WORD MEMBER(SD) 329
34	0008H	2	CCB. . . . . WORD MEMBER(XINIT)
27	0004H	4	CCBPOINTER . . . . POINTER 28 36 65 69 125 144 153 163 210 223 224 235 250 252 253 256 257 259 263 267 271 273 282 284 289 296 302 333 335 337 345 347 349
28	0004H	4	CCBPTR . . . . . STRUCTURE AT 327 329
29			CCBSTRUCTURE . . . LITERALLY 36
30			CCBTAIL1 . . . . . LITERALLY 36
31			CCBTAIL2 . . . . . LITERALLY 36
32			CFCOMPC. . . . . LITERALLY
32			CFCONOUT . . . . . LITERALLY 256 259 271
32			CFLISTCP . . . . . LITERALLY
32			CFSWITCHS. . . . . LITERALLY
32			CFVOUT . . . . . LITERALLY 210
161	0000H	1	CHAR . . . . . BYTE BASED(CHARADR) 171
187	0000H	1	CHAR . . . . . BYTE BASED(CHARADR) 204 218
246	018DH	1	CHAR . . . . . BYTE 263 265 267
186	0004H	2	CHARADR. . . . . WORD PARAMETER AUTOMATIC 187 204 208 213 218
159	0004H	2	CHARADR. . . . . WORD PARAMETER AUTOMATIC 160 161 171
25	0090H	1	CMOD . . . . . BYTE MEMBER(SD)
36	0006H	1	COLUMN . . . . . BYTE MEMBER(CCB)
247	018FH	1	CONTROLP . . . . . BYTE 252 266
22	0000H	1	COPYNUM. . . . . BYTE EXTERNAL(2) AT 265 329 338
36	0022H	2	COSLEEP. . . . . WORD MEMBER(CCB) 257 273
246	018EH	1	COUNT. . . . . BYTE 261 262 269
2			CR . . . . . LITERALLY
33			CSMABORT . . . . . LITERALLY 250 253 296 349

```
  33                CSMBACKGROUND. . .  LITERALLY        235  250  253  345
  33                CSMBUFFERED. . . .  LITERALLY        235  345
  33                CSMCTRLO . . . . .  LITERALLY        263  345  347
  33                CSMCTRLP . . . . .  LITERALLY        235  252  345  347
  33                CSMCTRLS . . . . .  LITERALLY        250  253
  33                CSMFILEFULL. . . .  LITERALLY        224
  33                CSMNOSWITCH. . . .  LITERALLY
  33                CSMPURGING . . . .  LITERALLY        223  282  284  347
  33                CSMSUSPEND . . . .  LITERALLY
  49  0000H      2  CX . . . . . . . .  WORD PARAMETER         50
  37                DATAMSG. . . . . .  LITERALLY        156  168  178
  25  004FH      1  DAYFILE. . . . . .  BYTE MEMBER(SD)
  39  0000H      1  DAYTA. . . . . .  BYTE MEMBER(VOUTQMSG)              171  181  204
   2                DCL. . . . . . . .  LITERALLY
  76  0086H      1  DELETED. . . . . .  BYTE INITIAL           84   92  107  132
  80  005EH     35  DELETEFILE . . . .  PROCEDURE STACK=0008H              344  352
  75  0085H      1  DELETEFLAG . . . .  BYTE INITIAL           83   96  343  351
  16                DISKMAPLEN . . . .  LITERALLY
  25  006AH      2  DLR. . . . . . . .  WORD MEMBER(SD)
  34  0002H      1  DOOR . . . . . . .  BYTE MEMBER(XINIT)
                    DOUBLE . . . . . .  BUILTIN          235  284  296  345  347
 320  0012H      2  DQ . . . . . . . .  WORD MEMBER(VCMXQ)
 317  0012H      2  DQ . . . . . . . .  WORD MEMBER(VOUTQ)
 319  0012H      2  DQ . . . . . . . .  WORD MEMBER(VINQ)
 307  0012H      2  DQ . . . . . . . .  WORD MEMBER(IQD)
 159  0219H    108  DRAINVOUTQ . . . .  PROCEDURE BYTE STACK=000AH              208  213  293
  25  006CH      2  DRL. . . . . . . .  WORD MEMBER(SD)
 322  0000H      1  DUMMY. . . . . . .  BYTE ARRAY(1) DATA
  73                DUMPOP . . . . . .  LITERALLY
  49  0000H      2  DX . . . . . . . .  WORD PARAMETER         50
  25  0044H      2  ENDSEG . . . . . .  WORD MEMBER(SD)
   2                FALSE. . . . . . .  LITERALLY         74   83   91   92   97  110  113  122  123  134  139
                                          149  165  173  177  197  203  227  299
  16                FARC . . . . . . .  LITERALLY
 323  01F2H     36  FCB. . . . . . . .  BYTE ARRAY(36) INITIAL              81   82   87   88   89   90  119  120
                                          121  141  142  143  331  332  334  336  339
  18                FCBLEN . . . . . .  LITERALLY         87
  16                FCR. . . . . . . .  LITERALLY
  16                FDIRSYS. . . . . .  LITERALLY
  16                FDISKMAP . . . . .  LITERALLY
  16                FDRVUSR. . . . . .  LITERALLY         88  339
  16                FDRVUSR2 . . . . .  LITERALLY
  16                FEX. . . . . . . .  LITERALLY         87
  77  0087H      1  FILEISEMPTY. . . .  BYTE INITIAL           96  112  123  130
  36  0004H      1  FLAG . . . . . . .  BYTE MEMBER(CCB)             210  256  259  271
 317  0004H      2  FLAGS. . . . . . .  WORD MEMBER(VOUTQ)
 307  0004H      2  FLAGS. . . . . . .  WORD MEMBER(IQD)
 320  0004H      2  FLAGS. . . . . . .  WORD MEMBER(VCMXQ)
 319  0004H      2  FLAGS. . . . . . .  WORD MEMBER(VINQ)
  25  0056H      2  FLAGS. . . . . . .  WORD MEMBER(SD)
 321  0000H      1  FLGS . . . . . . .  BYTE MEMBER(QPB)
  16                FNAME. . . . . . .  LITERALLY        331  332  334  336
  16                FNAME2 . . . . . .  LITERALLY
  16                FNAMELEN . . . . .  LITERALLY
   2                FOREVER. . . . . .  LITERALLY        342
  16                FRC. . . . . . . .  LITERALLY
  16                FRREC. . . . . . .  LITERALLY        119  120  141  142
```

```
 16                 FRRECO . . . . . .  LITERALLY
 16                 FRW. . . . . .  LITERALLY
 16                 FS1. . . . . .  LITERALLY
 16                 FTYPE. . . . . .  LITERALLY
 16                 FTYPE2 . . . . . .  LITERALLY
 16                 FTYPELEN . . . . .  LITERALLY
221  035CH    33   FULLDISK . . . . .  PROCEDURE STACK=000CH          232  239
 40  0000H     1   FUNC . . . . . .  BYTE PARAMETER        41
 52  0000H     1   FUNC . . . . . .  BYTE PARAMETER        53
 46  0000H     1   FUNC . . . . . .  BYTE PARAMETER        47
 43  0000H     1   FUNC . . . . . .  BYTE PARAMETER        44
186  0288H   164   GETCHAR. . . . . .  PROCEDURE BYTE STACK=0010H          263
162  018AH     1   HAVEACHAR. . . . .  BYTE       165  166  168
                   HIGH . . . . . .  BUILTIN        120  142
101  0088H   128   INBUF. . . . . .  BYTE ARRAY(128)      116  181  196
304  0545H    53   INITQ. . . . . .  PROCEDURE WORD STACK=000AH          333  335  337
102  0012H     2   INPTR. . . . . .  WORD INITIAL         124  181  182  194  196  198  199  280  300
 49  0000H         INTSYS . . . . . .  PROCEDURE EXTERNAL(6) STACK=0000H       58   62
307  0000H    28   IQD. . . . . .  STRUCTURE BASED(QDADDR)        308
292  0018H     2   JUNK . . . . . .  WORD       233
 25  0086H     4   LCB. . . . . .  WORD ARRAY(2) MEMBER(SD)
 34  000AH     2   LCB. . . . . .  WORD MEMBER(XINIT)
 35                 LCBSTRUCTURE . . .  LITERALLY
 25  008AH     1   LCKMAX . . . . . .  BYTE MEMBER(SD)
  2                 LF . . . . . .  LITERALLY
317  0000H     2   LINK . . . . . .  WORD MEMBER(VOUTQ)
307  0000H     2   LINK . . . . . .  WORD MEMBER(IQD)
319  0000H     2   LINK . . . . . .  WORD MEMBER(VINQ)
320  0000H     2   LINK . . . . . .  WORD MEMBER(VCMXQ)
  2                 LIT. . . . . .  LITERALLY        2    3    4    5    6    7    8    9   10   11   12
                                      13   14   15   16   17   18   29   30   31   32   33   34   35   37
                                      38   55   72   73  100  244  315
 72                 LOGEOF . . . . . .  LITERALLY
                    LOW. . . . . .  BUILTIN        119  141
 25  0052H     2   LUL. . . . . .  WORD MEMBER(SD)
 13                 MABORT . . . . . .  LITERALLY
 86  0081H    67   MAKEFILE . . . . .  PROCEDURE BYTE STACK=0008H          109
 25  0076H     2   MAL. . . . . .  WORD MEMBER(SD)
 13                 MATTACH. . . . . .  LITERALLY
 36  0010H     2   MAXBUFSIZ. . . . .  WORD MEMBER(CCB)          125  144
 13                 MCLOSEF. . . . . .  LITERALLY       81
 13                 MCREADQ. . . . . .  LITERALLY      167
 13                 MCURDSK. . . . . .  LITERALLY
 13                 MCWRITEQ . . . . .  LITERALLY
 13                 MDELAY . . . . . .  LITERALLY      211
 13                 MDELETEF . . . . .  LITERALLY       82   89
 13                 MDELETEQ . . . . .  LITERALLY
 13                 MDETACH. . . . . .  LITERALLY
 13                 MDIRBIOS . . . . .  LITERALLY
 13                 MDISPATCH. . . . .  LITERALLY
 25  0058H     2   MDUL . . . . . .  WORD MEMBER(SD)
 13                 MERRMODE . . . . .  LITERALLY      326
 25  005AH     2   MFL. . . . . .  WORD MEMBER(SD)
 13                 MGETCNS. . . . . .  LITERALLY
 13                 MGETLOGIN. . . . .  LITERALLY
 13                 MGETPD . . . . . .  LITERALLY      330
 36  0008H     1   MIMIC. . . . . .  BYTE MEMBER(CCB)          267
```

```
  14                  MISLEEP. . . . . LITERALLY        58
  14                  MIWAKEUP . . . . LITERALLY        62
  13                  MMAKEF . . . . . LITERALLY        90
  13                  MMAKEQ . . . . . LITERALLY       309
  25  004CH      2    MMP. . . . . . . WORD MEMBER(SD)
  25  0046H      1    MODULEMAP. . . . BYTE MEMBER(SD)
  40  0000H           MON1 . . . . . . PROCEDURE EXTERNAL(3) STACK=0000H        66   70   81   82   89  116
                       138  155  211  326  338  341
  43  0000H           MON2 . . . . . . PROCEDURE BYTE EXTERNAL(4) STACK=0000H        90  121  143  167  309
                       310
  46  0000H           MON4 . . . . . . PROCEDURE POINTER EXTERNAL(5) STACK=0000H         330
  13                  MOPENF . . . . . LITERALLY
  13                  MOPENQ . . . . . LITERALLY       310
 248  0190H      1    MOREINFILE . . . BYTE         249  250  262  263  277
                     MOV8 . . . . . . BUILTIN          331
                     MOVE . . . . . . BUILTIN          196  308  332  334  336
  13                  MPARSE . . . . . LITERALLY
  25  0040H      2    MPMSEG . . . . . WORD MEMBER(SD)
  25  007CH      2    MPMVERNUM. . . . WORD MEMBER(SD)
  13                  MPRTBUF. . . . . LITERALLY
  13                  MREADF . . . . . LITERALLY
  13                  MREADQ . . . . . LITERALLY        66  155
  13                  MREADRF. . . . . LITERALLY       143
  13                  MRESETDRV. . . . LITERALLY
  13                  MSELECT. . . . . LITERALLY
  13                  MSETATT. . . . . LITERALLY
  13                  MSETCNS. . . . . LITERALLY       338
  13                  MSETDMA. . . . . LITERALLY       116  138
  13                  MSETPRIOR. . . . LITERALLY       341
  13                  MSETUSR. . . . . LITERALLY
 317  0016H      2    MSGCNT . . . . . WORD MEMBER(VOUTQ)
 307  0016H      2    MSGCNT . . . . . WORD MEMBER(IQD)
 320  0016H      2    MSGCNT . . . . . WORD MEMBER(VCMXQ)
 319  0016H      2    MSGCNT . . . . . WORD MEMBER(VINQ)
 320  000EH      2    MSGLEN . . . . . WORD MEMBER(VCMXQ)
 317  000EH      2    MSGLEN . . . . . WORD MEMBER(VOUTQ)
 319  000EH      2    MSGLEN . . . . . WORD MEMBER(VINQ)
 307  000EH      2    MSGLEN . . . . . WORD MEMBER(IQD)
 320  0018H      2    MSGOUT . . . . . WORD MEMBER(VCMXQ)
 319  0018H      2    MSGOUT . . . . . WORD MEMBER(VINQ)
 317  0018H      2    MSGOUT . . . . . WORD MEMBER(VOUTQ)
 307  0018H      2    MSGOUT . . . . . WORD MEMBER(IQD)
  36  0009H      1    MSOURCE. . . . . BYTE MEMBER(CCB)
  13                  MSYSDAT. . . . . LITERALLY
  13                  MWRITEF. . . . . LITERALLY
  13                  MWRITEQ. . . . . LITERALLY        70
  13                  MWRITERF . . . . LITERALLY       121
  15                  MXCONIN. . . . . LITERALLY
  15                  MXCONOUT . . . . LITERALLY       265
  15                  MXLSTOUT . . . . LITERALLY       267
  15                  MXSWITCH . . . . LITERALLY
  15                  MXUPSTATUS . . . LITERALLY       285
 321  0008H      8    NAME . . . . . . BYTE ARRAY(8) MEMBER(QPB)      308
 320  0006H      8    NAME . . . . . . BYTE ARRAY(8) MEMBER(VCMXQ)          336
 319  0006H      8    NAME . . . . . . BYTE ARRAY(8) MEMBER(VINQ)           332
 317  0006H      8    NAME . . . . . . BYTE ARRAY(8) MEMBER(VOUTQ)          334
 307  0006H      8    NAME . . . . . . BYTE ARRAY(8) MEMBER(IQD)      308
```

```
 17                 NAMELEN. . . . . LITERALLY       332  334  336
 34  0006H      1   NCCB . . . . . . BYTE MEMBER(XINIT)
 25  0049H      1   NCCB . . . . . . BYTE MEMBER(SD)
 36  0007H      1   NCHAR. . . . . . BYTE MEMBER(CCB)
 25  0085H      1   NCIODEV. . . . . BYTE MEMBER(SD)
 25  0047H      1   NCNS . . . . . . BYTE MEMBER(SD)
 25  0083H      1   NCONDEV. . . . . BYTE MEMBER(SD)
 21  0000H      1   NCOPIES. . . . . BYTE EXTERNAL(2)            22
319  0002H      1   NET. . . . . . . BYTE MEMBER(VINQ)
320  0002H      1   NET. . . . . . . BYTE MEMBER(VCMXQ)
317  0002H      1   NET. . . . . . . BYTE MEMBER(VOUTQ)
307  0002H      1   NET. . . . . . . BYTE MEMBER(IQD)
321  0001H      1   NET. . . . . . . BYTE MEMBER(QPB)
 25  004AH      1   NFLAGS . . . . . BYTE MEMBER(SD)
 34  0007H      1   NLST . . . . . . BYTE MEMBER(XINIT)
 25  0048H      1   NLST . . . . . . BYTE MEMBER(SD)
 25  0084H      1   NLSTDEV. . . . . BYTE MEMBER(SD)
321  0004H      2   NMSGS. . . . . . WORD MEMBER(QPB)
320  0010H      2   NMSGS. . . . . . WORD MEMBER(VCMXQ)
319  0010H      2   NMSGS. . . . . . WORD MEMBER(VINQ)
317  0010H      2   NMSGS. . . . . . WORD MEMBER(VOUTQ)
307  0010H      2   NMSGS. . . . . . WORD MEMBER(IQD)
  2                 NO . . . . . . . LITERALLY
320  0014H      2   NQ . . . . . . . WORD MEMBER(VCMXQ)
319  0014H      2   NQ . . . . . . . WORD MEMBER(VINQ)
317  0014H      2   NQ . . . . . . . WORD MEMBER(VOUTQ)
307  0014H      2   NQ . . . . . . . WORD MEMBER(IQD)
 25  004EH      1   NSLAVES. . . . . BYTE MEMBER(SD)
105  0016H      2   NUMPURGEBUFCHARS . WORD INITIAL         188  191  198  279  301
 34  0005H      1   NVCNS. . . . . . BYTE MEMBER(XINIT)
 28  0000H      2   OFFSET . . . . . WORD MEMBER(CCBPTR)       329
 34  0000H      2   OFFSET . . . . . WORD MEMBER(XINITPTR)
 24  0000H      2   OFFSET . . . . . WORD MEMBER(SYSDATPTR)         328
 25  0088H      1   OPMAX. . . . . . BYTE MEMBER(SD)
320  0003H      1   ORG. . . . . . . BYTE MEMBER(VCMXQ)
319  0003H      1   ORG. . . . . . . BYTE MEMBER(VINQ)
317  0003H      1   ORG. . . . . . . BYTE MEMBER(VOUTQ)
307  0003H      1   ORG. . . . . . . BYTE MEMBER(IQD)
 52  0000H      2   P1 . . . . . . . WORD PARAMETER         53
 52  0000H      2   P2 . . . . . . . WORD PARAMETER         53
 36  000AH      1   PC . . . . . . . BYTE MEMBER(CCB)
314  0000H      1   PD . . . . . . . BYTE BASED(PDPOINTER) ARRAY(1)      331
315                 PDNAME . . . . . LITERALLY       331
313  001AH      4   PDPOINTER. . . . . POINTER         314  330  331
324  057AH    308   PLMSTART . . . . . PROCEDURE PUBLIC STACK=0018H
 25  006EH      2   PLR. . . . . . . WORD MEMBER(SD)
 55                 PSCIOSLEEP . . . . LITERALLY        58
 25  005CH      2   PUL. . . . . . . WORD MEMBER(SD)
245  0382H    334   PURGE. . . . . . PROCEDURE STACK=0014H         348
103  0108H    128   PURGEBUF . . . . . BYTE ARRAY(128)      138  196  218
244                 PURGEOKMASK. . . . LITERALLY       250  253
104  0014H      2   PURGEPTR . . . . . WORD INITIAL         188  192  199  217  218  280  300
176  0285H     51   PUTCHAR. . . . . . PROCEDURE BYTE STACK=0010H          237
 52  0000H          PXIOS1 . . . . . . PROCEDURE EXTERNAL(7) STACK=0000H       265  267  285
321  0002H      2   QADDR. . . . . . . WORD MEMBER(QPB)           65   69  153  163  311
 36  0020H      2   QBUFF. . . . . . . WORD MEMBER(CCB)
304  0004H      2   QDADDR . . . . . . WORD PARAMETER AUTOMATIC       305  307  308  309
```

```
   4                 QDSTRUCTURE. . . .  LITERALLY       307  317  319  320
 162  018BH      1   QEMPTY . . . . . .  BYTE       165  166  169  172
  11                 QFDEV. . . . . .  LITERALLY
   7                 QFHIDE . . . . . .  LITERALLY       317  319  320
   6                 QFKEEP . . . . . .  LITERALLY       317  319  320
   5                 QFMX . . . . . .  LITERALLY       320
  10                 QFRPL. . . . . .  LITERALLY
   8                 QFRSP. . . . . .  LITERALLY
   9                 QFTABLE. . . . . .  LITERALLY
  25  0074H      2   QLR. . . . . .  WORD MEMBER(SD)
  25  0060H      8   QMAU . . . . . .  WORD ARRAY(4) MEMBER(SD)
   3                 QNAMSIZ. . . . . .  LITERALLY       307  308  317  319  320  321  331
 321  0072H     16   QPB. . . . . .  STRUCTURE        65   66   69   70  153  154  155  163  164  167  308
                                        310  311
  36  001EH      2   QPBBUFFPTR . . . .  WORD MEMBER(CCB)
  36  0018H      1   QPBFLAGS . . . . .  BYTE MEMBER(CCB)
  36  001CH      2   QPBNMSGS . . . . .  WORD MEMBER(CCB)
  36  001AH      2   QPBQADDR . . . . .  WORD MEMBER(CCB)
  36  0019H      1   QPBRESRVD. . . . .  BYTE MEMBER(CCB)
  12                 QPBSTRUCTURE . . .  LITERALLY       321
  36  0002H      2   QUEUE. . . . . .  WORD MEMBER(CCB)
  25  005EH      2   QUL. . . . . .  WORD MEMBER(SD)
  36  0028H      2   R1 . . . . . .  WORD MEMBER(CCB)
  36  002AH      2   R2 . . . . . .  WORD MEMBER(CCB)
 128  0167H    125   READBUF. . . . . .  PROCEDURE BYTE STACK=0008H        139
  64  002AH     26   READCHANGEMXQ. . .  PROCEDURE STACK=0008H        222  251  281  295
 150  01E4H     53   READVOUTQ. . . . .  PROCEDURE STACK=0008H        236  353
  95  00C4H     28   RESETFILE. . . . .  PROCEDURE STACK=0002H        133  146  298
  34  0003H      2   RESRVD1. . . . . .  BYTE ARRAY(2) MEMBER(XINIT)
 306  0191H      1   RET. . . . . .  BYTE       309  310
 129  0188H      1   RET. . . . . .  BYTE       143  147
  25  0068H      2   RLR. . . . . .  WORD MEMBER(SD)
  78  000EH      2   RRR. . . . . .  WORD INITIAL          98  112  141  142  144  145
  19  0000H      2   RSPLINK. . . . . .  WORD EXTERNAL(0)          327
  25  0042H      2   RSPSEG . . . . . .  WORD MEMBER(SD)
  36  000DH      1   RSVD . . . . . .  BYTE MEMBER(CCB)
 325  0082H      2   SAVESTATE. . . . .  WORD
  25  0000H    145   SD . . . . . .  STRUCTURE BASED(SYSDATPOINTER)        88  329  339
  26  0000H      1   SDBYTE . . . . . .  BYTE BASED(SYSDATPOINTER) ARRAY(1)
  34  0002H      2   SEGMENT. . . . . .  WORD MEMBER(XINITPTR)
  28  0002H      2   SEGMENT. . . . . .  WORD MEMBER(CCBPTR)       327
  24  0002H      2   SEGMENT. . . . . .  WORD MEMBER(SYSDATPTR)         327
                     SETB . . . . . .  BUILTIN          87
                     SIZE . . . . . .  BUILTIN         329
  56  0000H     21   SLEEP. . . . . .  PROCEDURE STACK=000CH          257
  25  0070H      2   SLR. . . . . .  WORD MEMBER(SD)
  25  0008H     56   SPACE. . . . . .  WORD ARRAY(28) MEMBER(SD)
  25  004BH      1   SRCHDISK . . . . .  BYTE MEMBER(SD)
  36  0005H      1   STARTCOL . . . . .  BYTE MEMBER(CCB)
  36  000EH      2   STATE. . . . . .  WORD MEMBER(CCB)          223  224  235  250  252  253  263  282  284
                                        296  345  347  349
  25  0000H      8   SUPMOD . . . . . .  WORD ARRAY(4) MEMBER(SD)
  23  0000H      4   SYSDATPOINTER. . .  POINTER          24   25   26   88  329  339
  24  0000H      4   SYSDATPTR. . . . .  STRUCTURE AT         327  328
  25  008CH      4   SYSLTOT. . . . . .  WORD ARRAY(2) MEMBER(SD)
   2                 TAB. . . . . .  LITERALLY
  25  0050H      1   TEMPDISK . . . . .  BYTE MEMBER(SD)       88  339
```

```
  25  0072H     2  THRORT . . . . . .  WORD MEMBER(SD)
  34  0000H     1  TICK . . . . . .    BYTE MEMBER(XINIT)
  25  0051H     1  TICKSPERSEC. . . .  BYTE MEMBER(SD)
  34  0001H     1  TICKSSEC . . . . .  BYTE MEMBER(XINIT)
  25  007EH     4  TOD. . . . . .      WORD ARRAY(2) MEMBER(SD)
  25  0082H     1  TOD_SEC. . . . . .  BYTE MEMBER(SD)
   2             TRUE . . . . . .     LITERALLY        75   76   77   84   93   96  117  126  157  169  174
                                        179  184  205  215  219  249  342
  39  0001H     1  TYPE . . . . . .    BYTE MEMBER(VOUTQMSG)          156  168  178  180
  20  0000H     2  UDASEG . . . . . .  WORD EXTERNAL(1)
  36  0024H     2  USLEEP . . . . . .  WORD MEMBER(CCB)          289  302
  36  0008H     1  VC . . . . . .      BYTE MEMBER(CCB)
 320  0056H    28  VCMXQ. . . . . .    STRUCTURE INITIAL       336  337
  36  0016H     2  VCMXQ. . . . . .    WORD MEMBER(CCB)         65   69  337
  25  007AH     2  VERNUM . . . . . .  WORD MEMBER(SD)
  25  0078H     2  VERSION. . . . . .  WORD MEMBER(SD)
  36  0012H     2  VINQ . . . . . .    WORD MEMBER(CCB)         333
 319  003AH    28  VINQ . . . . . .    STRUCTURE INITIAL        332  333
 318  0182H    64  VINQBUF. . . . . .  BYTE ARRAY(64)      319
   1  0000H        VOUT . . . . . .    PROCEDURE STACK=0000H
 317  001EH    28  VOUTQ. . . . . .    STRUCTURE INITIAL        334  335
  36  0014H     2  VOUTQ. . . . . .    WORD MEMBER(CCB)         153  163  335
 316  0192H    32  VOUTQBUF . . . . .  BYTE ARRAY(32)       317
  39  000CH     2  VOUTQMSG . . . . .  STRUCTURE       154  156  164  168  171  178  180  181  204
  36  0026H     2  VSLEEP . . . . . .  WORD MEMBER(CCB)
  38             WAKEMSG. . . . . .  LITERALLY       180
  60  0015H    21  WAKEUP . . . . . .  PROCEDURE STACK=000CH          273  289  302
 106  00E0H   135  WRITEBUF . . . . .  PROCEDURE BYTE STACK=000CH          183  230
  68  0044H    26  WRITECHANGEMXQ . .  PROCEDURE STACK=0008H          225  272  275  287  297  340
 227  018CH     1  WRITEPENDING . . .  BYTE INITIAL          197  229  230  237  299
  74  0084H     1  WRITING. . . . . .  BYTE INITIAL           97  114  117  136  139
  79  0010H     2  WRR. . . . . .      WORD INITIAL           98  112  119  120  125  145
  34  0000H    12  XINIT. . . . . .    STRUCTURE BASED(XINITPOINTER)
  34             XINITOFFSET. . . .  LITERALLY
  34  0008H     4  XINITPOINTER . . .  POINTER          34
  34  0008H     4  XINITPTR . . . . .  STRUCTURE AT
```


MODULE INFORMATION:

```
  CODE AREA SIZE     = 06AEH   1710D
  CONSTANT AREA SIZE = 0001H      1D
  VARIABLE AREA SIZE = 0216H    534D
  MAXIMUM STACK SIZE = 0018H     24D
  771 LINES READ
  0 PROGRAM WARNINGS
  0 PROGRAM ERRORS
```

END OF PL/M-86 COMPILATION

```
VAX/VMS 8086 LOCATER, V1.0VX

INPUT FILE: VOUT.LNK
OUTPUT FILE: VOUT.DAT
CONTROLS SPECIFIED IN INVOCATION COMMAND:
  DO(SM(CODE,DATA,DATA,CONST,STACK))AD(SM(CODE(0),DATA(10000H)))
  SS(STACK(0))TO VOUT.DAT
WARNING 26:  DECREASING SIZE OF SEGMENT
  SEGMENT:  STACK
WARNING 36:  SEGMENTS OVERLAP
  SEGMENT:  CODE
  SEGMENT:  (NO NAME)
  LOW OVERLAP ADDRESS :  00200H
  HIGH OVERLAP ADDRESS:  00216H


SYMBOL TABLE OF MODULE RVOUT

BASE     OFFSET TYPE SYMBOL              BASE     OFFSET TYPE SYMBOL

1000H    0000H  PUB  RSPLINK            1000H    0010H  PUB  PD
1000H    0020H  PUB  UDASEG             1000H    0088H  PUB  U_RETSEG
1000H    0086H  PUB  U_WRKSEG           1000H    00A0H  PUB  U_INSYS
1000H    0004H  PUB  NCOPIES            0000H    05FCH  PUB  PLMSTART
0000H    0060H  PUB  PXIOS1             0000H    0060H  PUB  PXIOS2
0000H    0060H  PUB  PXIOS3             0000H    0060H  PUB  PXIOS4
0000H    0025H  PUB  XDOS               0000H    0025H  PUB  MON1
0000H    0025H  PUB  MON2               0000H    0025H  PUB  MON3
0000H    0025H  PUB  MON4               0000H    0034H  PUB  INTSYS
```

MODULE = VOUT

BASE	OFFSET	TYPE	SYMBOL	BASE	OFFSET	TYPE	SYMBOL
1000H	0360H	SYM	MEMORY	1000H	0004H	SYM	COPYNUM
1000H	0140H	SYM	SYSDATPOINTER	1000H	0140H	SYM	SYSDATPTR
1000H	0140H	BAS	SD	1000H	0140H	BAS	SDBYTE
1000H	0144H	SYM	CCBPOINTER	1000H	0144H	SYM	CCBPTR
1000H	0148H	SYM	XINITPOINTER	1000H	0148H	SYM	XINITPTR
1000H	0148H	BAS	XINIT	1000H	0144H	BAS	CCB
1000H	014CH	SYM	VOUTQMSG	0000H	0082H	SYM	SLEEP
STACK	0004H	SYM	ADDR	0000H	0097H	SYM	WAKEUP
STACK	0004H	SYM	ADDR	0000H	00ACH	SYM	READCHANGEMXQ
0000H	00C6H	SYM	WRITECHANGEMXQ	1000H	01C4H	SYM	WRITING
1000H	01C5H	SYM	DELETEFLAG	1000H	01C6H	SYM	DELETED
1000H	01C7H	SYM	FILEISEMPTY	1000H	014CH	SYM	RRR
1000H	0150H	SYM	WRR	0000H	00E0H	SYM	DELETEFILE
0000H	0103H	SYM	MAKEFILE	0000H	0146H	SYM	RESETFILE
1000H	01C8H	SYM	INBUF	1000H	0152H	SYM	INPTR
1000H	0248H	SYM	PURGEBUF	1000H	0154H	SYM	PURGEPTR
1000H	0156H	SYM	NUMPURGEBUFCHARS	0000H	0162H	SYM	WRITEBUF
0000H	01E9H	SYM	READBUF	1000H	02C8H	SYM	RET
1000H	02C9H	SYM	ACTIVEMSG	0000H	0266H	SYM	READVOUTQ
0000H	029BH	SYM	DRAINVOUTQ	STACK	0004H	SYM	CHARADR
STACK	0004H	BAS	CHAR	1000H	02CAH	SYM	HAVEACHAR
1000H	02CBH	SYM	QEMPTY	0000H	0307H	SYM	PUTCHAR
0000H	033AH	SYM	GETCHAR	STACK	0004H	SYM	CHARADR
STACK	0004H	BAS	CHAR	0000H	03DEH	SYM	FULLDISK
1000H	02CCH	SYM	WRITEPENDING	0000H	03FFH	SYM	BUFFER
0000H	0434H	SYM	PURGE	1000H	02CDH	SYM	CHAR
1000H	02CFH	SYM	COUNT	1000H	02CFH	SYM	CONTROLP
1000H	02D0H	SYM	MOREINFILE	0000H	0582H	SYM	ABORT
1000H	0158H	SYM	JUNK	0000H	05C7H	SYM	INITQ
STACK	0004H	SYM	QDADDR	1000H	02D1H	SYM	RET
STACK	0004H	BAS	IQD	1000H	015AH	SYM	PDPOINTER
1000H	015AH	BAS	PD	1000H	02D2H	SYM	VOUTQBUF
1000H	015EH	SYM	VOUTQ	1000H	02F2H	SYM	VINQBUF
1000H	017AH	SYM	VINQ	1000H	0196H	SYM	VCMXQ
1000H	01B2H	SYM	QPB	1000H	0356H	SYM	DUMMY
1000H	0332H	SYM	FCB	0000H	05FCH	SYM	PLMSTART
1000H	01C2H	SYM	SAVESTATE	0000H	0082H	LIN	56
0000H	0035H	LIN	58	0000H	0093H	LIN	59
0000H	0097H	LIN	60	0000H	009AH	LIN	62
0000H	00A8H	LIN	63	0000H	00ACH	LIN	64
0000H	00AFH	LIN	65	0000H	00BAH	LIN	66
0000H	00C4H	LIN	67	0000H	00C6H	LIN	68
0000H	00C9H	LIN	69	0000H	00D4H	LIN	70
0000H	00DEH	LIN	71	0000H	00E0H	LIN	80
0000H	00E3H	LIN	81	0000H	00EDH	LIN	82
0000H	00F7H	LIN	83	0000H	00FCH	LIN	84
0000H	0101H	LIN	85	0000H	0103H	LIN	86
0000H	0106H	LIN	87	0000H	0114H	LIN	88
0000H	0121H	LIN	89	0000H	0128H	LIN	90
0000H	0139H	LIN	91	0000H	013DH	LIN	92
0000H	0142H	LIN	93	0000H	0146H	LIN	94
0000H	0146H	LIN	95	0000H	0149H	LIN	96
0000H	0153H	LIN	97	0000H	0158H	LIN	98
0000H	0160H	LIN	99	0000H	0162H	LIN	106

0000H	0165H	LIN	107	0000H	016CH	LIN	109
0000H	0175H	LIN	110	0000H	0177H	LIN	111
0000H	0177H	LIN	112	0000H	0185H	LIN	113
0000H	0185H	LIN	114	0000H	018EH	LIN	116
0000H	0198H	LIN	117	0000H	019DH	LIN	119
0000H	01A3H	LIN	120	0000H	01A6H	LIN	121
0000H	01B9H	LIN	122	0000H	01BDH	LIN	123
0000H	01C2H	LIN	124	0000H	01C8H	LIN	125
0000H	01E5H	LIN	126	0000H	01E9H	LIN	127
0000H	01E9H	LIN	128	0000H	01ECH	LIN	130
0000H	01F3H	LIN	132	0000H	01FCH	LIN	133
0000H	01FFH	LIN	134	0000H	0203H	LIN	136
0000H	020AH	LIN	138	0000H	0214H	LIN	139
0000H	0219H	LIN	141	0000H	021FH	LIN	142
0000H	0227H	LIN	143	0000H	0233H	LIN	144
0000H	0258H	LIN	145	0000H	025EH	LIN	146
0000H	0261H	LIN	147	0000H	0266H	LIN	148
0000H	0266H	LIN	150	0000H	0269H	LIN	151
0000H	0270H	LIN	152	0000H	0272H	LIN	153
0000H	0270H	LIN	154	0000H	0283H	LIN	155
0000H	0280H	LIN	156	0000H	0294H	LIN	157
0000H	0299H	LIN	158	0000H	029BH	LIN	159
0000H	029EH	LIN	163	0000H	02A9H	LIN	164
0000H	02AFH	LIN	165	0000H	02B9H	LIN	166
0000H	02CAH	LIN	167	0000H	02D8H	LIN	168
0000H	02E7H	LIN	169	0000H	02ECH	LIN	170
0000H	02EEH	LIN	171	0000H	02F6H	LIN	172
0000H	02FDH	LIN	173	0000H	0301H	LIN	174
0000H	0307H	LIN	175	0000H	0307H	LIN	176
0000H	030AH	LIN	177	0000H	030FH	LIN	178
0000H	0316H	LIN	180	0000H	0318H	LIN	181
0000H	032CH	LIN	182	0000H	0331H	LIN	183
0000H	0336H	LIN	184	0000H	033AH	LIN	185
0000H	033AH	LIN	186	0000H	033DH	LIN	188
0000H	0347H	LIN	189	0000H	034EH	LIN	191
0000H	0354H	LIN	192	0000H	035AH	LIN	193
0000H	035CH	LIN	194	0000H	0363H	LIN	196
0000H	0373H	LIN	197	0000H	0378H	LIN	198
0000H	037FH	LIN	199	0000H	0387H	LIN	200
0000H	0387H	LIN	201	0000H	038EH	LIN	203
0000H	0393H	LIN	204	0000H	0398H	LIN	206
0000H	0398H	LIN	208	0000H	03A4H	LIN	210
0000H	03A2H	LIN	211	0000H	03BCH	LIN	212
0000H	03BEH	LIN	213	0000H	03C4H	LIN	215
0000H	03C6H	LIN	217	0000H	03CDH	LIN	218
0000H	03D9H	LIN	219	0000H	03DEH	LIN	220
0000H	03DEH	LIN	221	0000H	03E1H	LIN	222
0000H	03E4H	LIN	223	0000H	03F4H	LIN	224
0000H	03FAH	LIN	225	0000H	03FDH	LIN	226
0000H	03FFH	LIN	228	0000H	0402H	LIN	229
0000H	0409H	LIN	230	0000H	040EH	LIN	235
0000H	041EH	LIN	236	0000H	0421H	LIN	237
0000H	042DH	LIN	239	0000H	0430H	LIN	240
0000H	0432H	LIN	242	0000H	0432H	LIN	243
0000H	0434H	LIN	245	0000H	0437H	LIN	249
0000H	043CH	LIN	250	0000H	045AH	LIN	251
0000H	045DH	LIN	252	0000H	0474H	LIN	253
0000H	0480H	LIN	255	0000H	0481H	LIN	256

```
0000H   048FH  LIN    257          0000H   0496H  LIN    258
0000H   0498H  LIN    259          0000H   04A1H  LIN    260
0000H   04A9H  LIN    261          0000H   04B7H  LIN    262
0000H   04C9H  LIN    263          0000H   04DFH  LIN    265
0000H   04F1H  LIN    266          0000H   04F8H  LIN    267
0000H   050FH  LIN    269          0000H   0513H  LIN    270
0000H   0515H  LIN    271          0000H   051EH  LIN    272
0000H   0521H  LIN    273          0000H   052CH  LIN    274
0000H   052EH  LIN    275          0000H   0531H  LIN    276
0000H   0534H  LIN    277          0000H   053DH  LIN    279
0000H   0543H  LIN    280          0000H   054FH  LIN    281
0000H   0552H  LIN    282          0000H   0564H  LIN    284
0000H   056AH  LIN    285          0000H   0572H  LIN    287
0000H   0575H  LIN    289          0000H   0580H  LIN    290
0000H   0582H  LIN    291          0000H   0585H  LIN    293
0000H   0590H  LIN    295          0000H   0593H  LIN    296
0000H   059DH  LIN    297          0000H   05A0H  LIN    298
0000H   05A3H  LIN    299          0000H   05A8H  LIN    300
0000H   05B4H  LIN    301          0000H   05BAH  LIN    302
0000H   05C5H  LIN    303          0000H   05C7H  LIN    304
0000H   05CAH  LIN    308          0000H   05D8H  LIN    309
0000H   05E7H  LIN    310          0000H   05F5H  LIN    311
0000H   05FCH  LIN    312          0000H   05FCH  LIN    324
0000H   05FFH  LIN    326          0000H   0609H  LIN    327
0000H   0612H  LIN    328          0000H   0618H  LIN    329
0000H   062DH  LIN    330          0000H   063FH  LIN    331
0000H   0656H  LIN    332          0000H   0664H  LIN    333
0000H   0673H  LIN    334          0000H   0681H  LIN    335
0000H   0690H  LIN    336          0000H   069EH  LIN    337
0000H   06A0H  LIN    338          0000H   06B9H  LIN    339
0000H   06C6H  LIN    340          0000H   06C9H  LIN    341
0000H   06D3H  LIN    342          0000H   06D3H  LIN    343
0000H   06DAH  LIN    344          0000H   06DDH  LIN    345
0000H   06E0H  LIN    346          0000H   06F2H  LIN    347
0000H   0705H  LIN    348          0000H   070AH  LIN    349
0000H   071AH  LIN    350          0000H   0710H  LIN    351
0000H   0724H  LIN    352          0000H   0729H  LIN    353
0000H   072CH  LIN    354          0000H   072EH  LIN    355
0000H   0082H  LIN    356
```


MEMORY MAP OF MODULE RVOUT

SEGMENT MAP

START	STOP		LENGTH	ALIGN	NAME	CLASS	OVERLAY
00000H	0072FH	C	0730H	G	CODE	CODE	
00200H	00216H		0017H	A	(ABSOLUTE)		
10000H	1013FH		0140H	G	DATS	DATA	
10140H	10355H		0216H	W	DATA	DATA	
10356H	10356H		0001H	W	CONST	CONST	
10358H	10358H		0000H	W	STACK	STACK	
10360H	10360H		0000H	G	??SEG		
10360H	10360H		0000H	W	MEMORY	MEMORY	

GROUP MAP

ADDRESS	GROUP OR SEGMENT NAME
10000H	DGROUP
	DATS
	CONST
	DATA
00000H	CGROUP
	CODE

```
VAX/VMS 8086/8087/8088 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE RVOUT
OBJECT MODULE PLACED IN RHVOUT.OBJ
NO INVOCATION LINE CONTROLS

LOC  OBJ                  LINE     SOURCE

                             1     ;       Code and Data Interface for VOUT RSP
                             2     ;       (for virtual console support)
                             3     ;       March 30, 1983
                             4
                             5     name    rvout
                             6
                             7     cgroup  group   code
                             8     dgroup  group   dats
                             9
                            10     assume   cs:cgroup,ds:dgroup
                            11
                            12             public  xdos,mon1,mon2,mon3,mon4,intsys
                            13             public rsplink,pd,udaseg,u_retseg,u_wrkseg
                            14             public u_insys, ncopies
                            15             extrn  plmstart:near
                            16
                            17
----                        18     dats    segment public  'DATA'
0000                        19     org     0
                            20
  0010                      21     rsphdr_len      equ     16
  0030                      22     pd_len          equu    30H
  0100                      23     uda_len         equ     100H
  0060                      24     insysoff        equ     60H
  0000                      25     rsp_top         equ     0
  0010                      26     rsp_pd          equ     rsp_top + rsphdr_len
  0040                      27     rsp_uda         equ     rsp_pd + pd_len
  0140                      28     rsp_bottom      equ     rsp_uda + uda_len
0000                        29             org     rsp_top
                            30                                     ;RSP header
0000 0000                   31     rsplink dw      0               ;becomes system data page paragraph
0002 4700                   32     sdatvar dw      nvcns           ;gensys makes a copy per virtual cons
0004 00                     33     ncopies db      0
0005 0000                   34             dw      0,0,0,0, 0
0007 0000
0009 0000
0008 0000
000D 0000
000F 00                     35             db      0
0010                        36             org     rsp_pd
0010 0000                   37     pd      dw      0,0             ;link fields
0012 0000
0014 00                     38             db      0               ;status
0015 B4                     39             db      180             ;initial priority - better than TMPs and PIN
0016 0300                   40             dw      3               ;flags - system and keep
0018 564F5554202020         41             db      'VOUT    '      ;name
     20
0020 0400                   42     udaseg  dw      rsp_uda/10h     ;uda paragraph
0022 00                     43             db      0,0             ;disk,user
0023 00
```

```
LOC  OBJ                  LINE      SOURCE

0024 00                     44              db      0,0             ;ldisk,luser
0025 00
0026 FF00                   45              dw      0ffh            ;puramem - re-entrant
                            46                                      ;rest of pd
                            47
0040                        48      org     rsp_uda                 ;start of uda
0040 0000                   49      uda             dw      0
0042 0000                   50                      dw      0       ;init DMA, must be set by program
0044 0000                   51                      dw      0,0,0,0, 0,0,0,0, 0,0,0,0
0046 0000
0048 0000
004A 0000
004C 0000
004E 0000
0050 0000
0052 0000
0054 0000
0056 0000
0058 0000
005A 0000
005C 0000                   52                      dw      0,0,0,0, 0,0,0,0, 0,0,0,0
005E 0000
0060 0000
0062 0000
0064 0000
0066 0000
0068 0000
006A 0000
006C 0000
006E 0000
0070 0000
0072 0000
0074 3A01           R       53                      dw      offset stk_top
0076 0000                   54                      dw      0,0,0,0, 0,0,0,0
0078 0000
007A 0000
007C 0000
007E 0000
0080 0000
0082 0000
0084 0000
0086 0000                   55      u_wrkseg        dw      0
0088 0000                   56      u_retseg        dw      0
                            57
00A0                        58      org     rsp_uda + insysoff
00A0 01                     59      u_insys         db      1
00A1 00                     60                      db      0       ;status save
00A2 0000                   61                      dw      0       ;ccb
00A4 0000                   62                      dw      0       ;lcb
00A6 00                     63                      db      0       ;delimiter for print string
                            64
00A7 (147                   65                      db      93h dup (0cch)  ;fill UDA stack with INT 3s
     CC
     )
                            66
```

```
LOC  OBJ                  LINE     SOURCE

013A 0000            E      67     stk_top dw      plmstart
013C 0000                   68             dw      0,0             ;segment and flags - unknown
013E 0000
                            69
----                        70     dats ends
                            71
----                        72     code    segment public 'CODE'
                            73
0000 434F5059524947         74             db      'COPYRIGHT (C) 1983,'
     485420283329
     313938332C
0013 20444947495441         75             db      ' DIGITAL RESEARCH '
     4C205245534541
     52434820
                            76
0025                        77     xdos    proc
0025 55                     78             push    bp
0026 8BEC                   79             mov     bp,sp
0028 8B5604                 80             mov     dx,[bp+4]
002B 8B4E06                 81             mov     cx,[bp+6]
002E CDE0                   82             int     224
0030 5D                     83             pop     bp
0031 C20400                 84             ret     4
                            85     xdos    endp
                            86
  0025                      87     mon1    equ     xdos
  0025                      88     mon2    equ     xdos
  0025                      89     mon3    equ     xdos
  0025                      90     mon4    equ     xdos
                            91
  0000                      92     supervisor      equ     0
  0068                      93     rlr             equ     68h
  0010                      94     p_uda           equ     10h
  0047                      95     nvcns           equ     47h
                            96
0034                        97     intsys  proc                            ;call D.S. as if we are
0034 55                     98             push bp                         ;already in it
0035 8BEC                   99             mov bp,sp
                           100
0037 8B5E04                101             mov bx, [bp+4]
003A 8B5606                102             mov dx, [bp+6]
003D 8B4E08                103             mov cx, [bp+8]
                           104
                           105
0040 1E                    106             push ds
0041 8E1E0000        R     107             mov ds, rsplink                 ;DS = Sysdat Segment
0045 8B366800              108             mov si, ds:word ptr rlr
0049 8E4410                109             mov es, [si + p_uda]
004C FF1E0000              110             call ds:dword ptr [supervisor]
0050 1F                    111             pop ds
                           112
0051 5D                    113             pop bp
0052 C20600                114             ret 6
                           115     intsys  endp
                           116
```

```
LOC  OBJ                  LINE      SOURCE

----                       117      code    ends
                           118      end

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

VAX/VMS 8086/8087/8088 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE PXIOS
OBJECT MODULE PLACED IN PXIOS.OBJ
NO INVOCATION LINE CONTROLS

```
LOC  OBJ                  LINE     SOURCE

                             1     ;       Interface to call Physical XIOS
                             2     ;       From a process not in the O.S.
                             3     ;       code reentrant, separate data areas per process
                             4     ;       used by VOUT and PIN RSPs
                             5
                             6     name    pxios
                             7
                             8     cgroup  group   code
                             9     dgroup  group   dats
                            10
                            11     assume  cs:cgroup
                            12     assume  ds:dgroup
                            13
----                        14     dats    segment public 'DATA'
                            15             extrn   rsplink:word    ;segment of SYSDAT
                            16             extrn   udaseg:word     ;UDA must be in ES for XIOS call
                            17             extrn   u_retseg:word, u_wrkseg:word, u_insys:byte
----                        18     dats    ends
                            19
----                        20     code    segment public 'CODE'
                            21             public  pxios1, pxios2, pxios3, pxios4
  0028                      22             xiosmod equ     28h
  0010                      23             p_uda   equ     10h
  0068                      24             rlr     equ     68h
                            25
  0000                      26     pxios2  equ     pxios1
  0000                      27     pxios3  equ     pxios1
  0000                      28     pxios4  equ     pxios1
                            29
0000                        30     pxios1  proc
0000 55                     31             push    bp
0001 8BEC                   32             mov     bp,sp
                            33
0003 8B4608                 34             mov     ax,[bp+8]               ;set up registers
0006 8B4E06                 35             mov     cx,[bp+6]
0009 8B5604                 36             mov     dx,[bp+4]
                            37
000C 1E                     38             push    ds
000D 8E1E0000         E     39             mov     ds,rsplink              ;SYSDAT
0011 8B366800               40             mov     si,ds:word ptr rlr      ;ready list root
0015 8E4410                 41             mov     es,[p_uda+si]           ;UDA
0018 FF1E2800               42             call    ds:dword ptr [xiosmod]
001C 1F                     43             pop     ds
                            44
001D 5D                     45             pop     bp
001E C20600                 46             ret     6
                            47     pxios1  endp
                            48
----                        49     code    ends
                            50     end
```

8086/8087/8088 MACRO ASSEMBLER PXIOS

LOC OBJ LINE SOURCE

ASSEMBLY COMPLETE, NO ERRORS FOUND