```
ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE STATUS
OBJECT MODULE PLACED IN SYSTAT.OBJ
COMPILER INVOKED BY:  :F0: SYSTAT.PLM XREF OPTIMIZE(3) DEBUG
```



```
                $title('CCP/M-86 1.0 Systat Process - Transient')
                $compact
                                /* want 32 bit pointers */
  1             status:
                do;

                $include (:f2:copyrt.lit)
              =
              =     /*
              =       Copyright (C) 1983
              =       Digital Research
              =       P.O. Box 579
              =       Pacific Grove, CA 93950
              =     */
              =

                $include (:f2:vaxcmd.lit)
              =
              =     /**** VAX commands for generation - read the name of this program
              =             for PROGNAME below.
              =
              =         $ util := PROGNAME
              =         $ ccpmsetup              ! set up environment
              =         $ assign 'f$uirectory()' f1:        ! use local dir for temp files
              =         $ plm86 'util'.plm xref 'p1' optimize(3) debug
              =         $ link86 f2:scd.obj, 'util'.obj  to 'util'.lnk
              =         $ loc86 'util'.lnk od(sm(code,dats,data,stack,const)) -
              =               ad(sm(code(0),dats(10000h))) ss(stack(+32)) to 'util'.
              =         $ h86 'util'
              =
              =     ***** Then, on a micro:
              =         A>vax progname.h86 $fans
              =         A>gencmd progname data[b1000]
              =
              =     ***** Notes: Stack is increased for interrupts.  Const(ants) are last
              =             to force hex generation.
              =     ****/

                $include (scomon.plm)
              =
              =
              =
              =
              =
              =
              =         /* Common include Module for RSP and Transient CCPMSTAT */
              =
              =
              =     $include(:f2:newcom.lit)
              =1
  2   1       =1    declare
```

```
        =1          lit                 literally           'literally',
        =1          dcl                 lit                 'declare',
        =1          true                lit                 '0ffh',
        =1          false               lit                 '0',
        =1          no                  lit                 'not',
        =1          boolean             lit                 'byte',
        =1          forever             lit                 'while true',
        =1          tab                 lit                 ')';
        =1
 3   1  =    dcl buff (128) byte external;
        =
        =
 4   1  =    mon1:
        =      procedure (func,info) external;
 5   2  =        dcl func byte;
 6   2  =        dcl info address;
 7   2  =      end mon1;
        =
 8   1  =    mon2:
        =      procedure (func,info) byte external;
 9   2  =        dcl func byte;
10   2  =        dcl info address;
11   2  =      end mon2;
        =
12   1  =    mon3:
        =      procedure (func,info) address external;
13   2  =        dcl func byte;
14   2  =        dcl info address;
15   2  =      end mon3;
        =
16   1  =    mon4:
        =      procedure (func,info) pointer external;
17   2  =        dcl func byte;
18   2  =        dcl info address;
19   2  =      end mon4;
        =
        =
20   1  =    patch: procedure;   /* dummy area for patching code segments */
21   2  =      declare i address;
22   2  =        i=i+5;  i=i+5;  i=i+5;  i=i+5;  i=i+5;
27   2  =        i=i+5;  i=i+5;  i=i+5;  i=i+5;  i=i+5;
32   2  =        i=i+5;  i=i+5;  i=i+5;  i=i+5;  i=i+5;
37   2  =        i=i+5;  i=i+5;  i=i+5;  i=i+5;  i=i+5;
42   2  =        i=i+5;  i=i+5;  i=i+5;  i=i+5;  i=i+5;
47   2  =        i=i+5;  i=i+5;  i=i+5;  i=i+5;  i=i+5;
52   2  =        i=i+5;  i=i+5;  i=i+5;  i=i+5;  i=i+5;
57   2  =        i=i+5;  i=i+5;  i=i+5;  i=i+5;  i=i+5;
62   2  =        i=i+5;  i=i+5;  i=i+5;  i=i+5;  i=i+5;
67   2  =        i=i+5;  i=i+5;  i=i+5;  i=i+5;  i=i+5;
72   2  =        i=i+5;  i=i+5;  i=i+5;  i=i+5;  i=i+5;
77   2  =      end patch;
        =
        =
78   1  =    dcl maxpd byte initial (63);                /* Maximum # process descript's*/
79   1  =    dcl maxmd byte initial (80);                /* Maximum # memory descriptors*/
80   1  =    dcl maxqflags byte initial (40h);           /* Max Value for a queue flag  */
81   1  =    dcl freename (*)byte initial ('* FREE *');  /* For free memory partitions  */
```

```
 82   1  =   dcl maxqueues byte initial (64);          /* Maximum # of system queues  */
 83   1  =   dcl lbracket byte initial ('[');
 84   1  =   dcl rbracket byte initial (']');
 85   1  =   dcl repeat byte;                          /* Controls continuous display */
 86   1  =   dcl intrval word;                         /* Controls update timing      */
 87   1  =   dcl specified byte;                       /* Command line argument flag  */
 88   1  =   dcl flaglen byte initial (3);
          =
          =   /*****************************************************************************/
          =   /*                                                                           */
          =   /*                  Terminal-dependent Control Characters                    */
          =   /*                                                                           */
          =   /*****************************************************************************/
          =
 89   1  =   dcl clearseq (6) byte initial (2,01bh,'E',0,0,0);  /* ESC E = clear screen*/
 90   1  =   dcl homeseq (6)  byte initial (2,01bh,'H',0,0,0);  /* ESC H = home cursor */
 91   1  =   dcl CR  lit '13';                         /* CR = carriage return*/
 92   1  =   dcl LF  lit '10';                         /* LF = line feed      */
          =
          =
          =   $include (:f2:mdsat.lit)
          =1
 93   1  =1  declare md$structure literally
          =1    'structure(
          =1    link word,
          =1    start word,
          =1    length word,
          =1    plist word,
          =1    unused word)';
          =1
 94   1  =1  declare ms$structure literally
          =1    'structure(
          =1     link word,
          =1     start word,
          =1     length word,
          =1     flags word,
          =1     mau word)';
          =1
 95   1  =1  declare sat$structure literally
          =1    'structure(
          =1     start word,
          =1     len   word,
          =1     num$allocs byte)';
          =1
          =   $include (:f2:proces.lit)
          =1
          =1  /*
          =1      Proces Literals MP/M-8086 II
          =1  */
          =1
 96   1  =1  declare pnamsiz literally '8';
          =1
 97   1  =1  declare pd$hdr literally 'structure
          =1    (link word,thread word,stat byte,prior byte,flag word,
          =1    name (8) byte,uda word,dsk byte,user byte,ldsk byte,luser byte,
          =1    mem word';
          =1
```

```
 98   1  =1  declare pd$structure literally 'pd$hdr,
          =1    dvract word,wait word,org byte,net byte,parent word,
          =1    cns byte,abort byte,conmode word,lst byte,sf3 byte,sf4 byte,sf5 byte,
          =1    reservd (4) byte,pret word,scratch word)';
          =1
 99   1  =1    declare psrun                lit '00',
          =1            pspoll               lit '01',
          =1            psdelay              lit '02',
          =1            psswap               lit '03',
          =1            psterm               lit '04',
          =1            pssleep              lit '05',
          =1            psdq                 lit '06',
          =1            psnq                 lit '07',
          =1            psflagwait           lit '08',
          =1            psciowait            lit '09';
          =1
100   1  =1    declare pf$sys               lit '00001h',
          =1            pf$keep              lit '00002h',
          =1            pf$kernal            lit '00004h',
          =1            pf$pure              lit '00008h',
          =1            pf$table             lit '00010h',
          =1            pf$resource          lit '00020h',
          =1            pf$raw               lit '00040h',
          =1            pf$ctlc              lit '00080h',
          =1            pf$active            lit '00100h',
          =1            pf$tempkeep          lit '00200h',
          =1            pf$ctld              lit '00400h',
          =1            pf$childabort        lit '00800h',
          =1            pf$noctls            lit '01000h';
          =1
101   1  =1    declare pcm$ll               lit '00001h',
          =1            pcm$ctls             lit '00002h',
          =1            pcm$rout             lit '00004h',
          =1            pcm$ctlc             lit '00008h',
          =1            pcm$ctlo             lit '00080h',
          =1            pcm$rsx              lit '00300h';
          =   $include (:f2:sd.lit)
          =1
          =1  /* System Data Page */
          =1
102   1  =1    dcl sysdat$pointer pointer;
103   1  =1    dcl sysdat$ptr structure(
          =1      offset word,
          =1      segment word) at (@sysdat$pointer);
104   1  =1    declare sd based sysdat$pointer structure (
          =1        supmod (4) word,
          =1    /*  rtmmod (4) word,
          =1        memmod (4) word,
          =1        ciomod (4) word,
          =1        bdosmod (4) word,
          =1        xiosmod (4) word,
          =1        netmod (4) word,
          =1        reservd (4) word */
          =1        space(28) word,
          =1        mpmseg word,
          =1        rspseg word,
          =1        endseg word,
```

```
          =1        module$map byte,
          =1        ncns byte,
          =1        nlst byte,
          =1        nccb byte,
          =1        nflags byte,
          =1        srchdisk byte,
          =1        mmp word,
          =1        nslaves byte,
          =1        dayfile byte,
          =1        tempdisk byte,
          =1        tickspersec byte,
          =1        lul word,
          =1        ccb word,
          =1        flags word,
          =1        mdul word,
          =1        mfl word,
          =1        pul word,
          =1        qul word,
          =1        qmau (4) word,
          =1        rlr word,
          =1        dlr word,
          =1        drl word,
          =1        plr word,
          =1        slr word,
          =1        thrdrt word,
          =1        qlr word,
          =1        mal word,
          =1        version word,
          =1        vernum word,
          =1        mpmvernum word,
          =1        tod (2) word,
          =1        tod_sec byte,
          =1        ncondev byte,
          =1        nlstdev byte,
          =1        nciodev byte,
          =1        lcb (2) word,
          =1        lckmax byte,
          =1        opmax byte,
          =1        sysltot (2) word,
          =1        cmod byte );
          =1
          =1
105   1   =1  declare sd$byte based sysdat$pointer (1) byte;
          =1
          =   $include (:f2:qd.lit)
          =1
          =1  /* Queue Descriptor */
          =1
106   1   =1  dcl qnamsiz lit '8';
          =1
107   1   =1  dcl qd$structure lit 'structure(
          =1    link  word,
          =1    net byte,
          =1    ory byte,
          =1    flags word,
          =1    name(qnamsiz) byte,
          =1    msglen word,
```

```
          =1    nmsgs word,
          =1    dq word,
          =1    nq word,
          =1    msgcnt word,
          =1    msgout word,
          =1    buffer word)';
          =1
          =1  /* queue flag values */
          =1
108   1   =1  dcl qf$mx        lit '001h'; /* Mutual Exclusion */
109   1   =1  dcl qf$keep      lit '002h'; /* NO DELETE        */
110   1   =1  dcl qf$hide      lit '004h'; /* Not User writable    */
111   1   =1  dcl qf$rsp       lit '008h'; /* rsp queue        */
112   1   =1  dcl qf$table     lit '010h'; /* from qd table    */
113   1   =1  dcl qf$rpl       lit '020h'; /* rpl queue        */
114   1   =1  dcl qf$dev       lit '040h'; /* device queue     */
          =1
          =1  /* Queue Parameter Block */
          =1
115   1   =1  dcl qpb$structure lit 'structure(
          =1    flgs    byte,
          =1    net     byte,
          =1    qaddr   word,
          =1    nmsgs   word,
          =1    buffptr word,
          =1    name (qnamsiz) byte )';
          =1
          =   $include (:f2:flag.lit)
          =1
          =1  /* Flag Format */
          =1
116   1   =1  dcl flag$structure  lit 'structure(
          =1      pd word,
          =1      ignore byte)';
          =1
          =   $include (:f2:uda.lit)
          =1
          =1  /* MP/M-86 II  User Data Area format - August 8, 1981 */
          =1
117   1   =1  declare uda$structure lit 'structure (
          =1      dparam          word,
          =1      dma$ofst        word,
          =1      dma$seg         word,
          =1      func            byte,
          =1      searchl         byte,
          =1      searcha         word,
          =1      searchabase     word,
          =1      dcnt            word,
          =1      dblk            word,
          =1      error$mode      byte,
          =1      mult$cnt        byte,
          =1      df$password     (8) byte,
          =1      pd$cnt          byte)';
          =1
          =   $include (:f2:vccb.lit)
          =1      /* Concurrent CP/M Character Control Block Structure */
          =1
```

```
      =1  /*              +---------+---------+---------+---------+
      =1          00      |      attach       |       queue       |
      =1                  +---------+---------+---------+---------+
      =1          04      |  flag   | startcol| column  |  nchar  |
      =1                  +---------+---------+---------+---------+
      =1          08      |  mimic  | msource |   pc    |   vc    |
      =1                  +---------+---------+---------+---------+
      =1          0C      |  btmp   |  rsvd   |       state       |
      =1                  +---------+---------+---------+---------+
      =1          10      |     maxbufsiz     |       vinq        |
      =1                  +---------+---------+---------+---------+
      =1          14      |       voutq       |       vcmxq       |
      =1                  +---------+---------+---------+---------+
      =1          18      | qpbflgs | qpbfill |      qpbqaddr     |
      =1                  +---------+---------+---------+---------+
      =1          1C      |      qpbnmsgs     |     qpbbuffptr    |
      =1                  +---------+---------+---------+---------+
      =1          20      |       qbuff       |      cosleep      |
      =1                  +---------+---------+---------+---------+
      =1          24      |      usleep       |       vsleep      |
      =1                  +---------+---------+---------+---------+
      =1          28      |           ... reserved ...            |
      =1                  +---------+---------+---------+---------+
      =1
      =1
      =1
      =1
      =1  */
      =1
  118 1 =1  dcl ccb$structure lit 'structure (attach address, queue address,
      =1   flag byte, startcol byte, column byte, nchar byte, mimic byte, msource byte,
      =1   ccb$tail1';
  119 1 =1  dcl ccb$tail1 lit
      =1   'pc byte, vc byte, btmp byte, rsvd byte, state word, maxbufsiz word,
      =1    ccb$tail2';
  120 1 =1  dcl ccb$tail2 lit
      =1   'vinq address, voutq address, vcmxq address,
      =1    qpbflags byte, qpbresrvd byte, qpbqaddr address,
      =1    qpbnmsgs address, qpbbuffptr address, qbuff address, cosleep word,
      =1    usleep word, vsleep word, r1 word, r2 word)';
      =1
  121 1 =1    declare                                 /* flag values                    */
      =1      cf$listcp         lit       '001h',   /* control P toggle               */
      =1      cf$compc          lit       '002h',   /* suppress output                */
      =1      cf$switchs        lit       '004h',   /* XIOS supports switch screening */
      =1      cf$conout         lit       '008h',   /* XIOS console output ownership  */
      =1      cf$vout           lit       '010h';   /* process writing to VOUTQ       */
      =1
      =1  /* values of state byte */
      =1                                              /* conout goes to XIOS  */
      =1
      =1  /* state word flags */
      =1
  122 1 =1  dcl
      =1  csm$buffered          lit       '0001h',
      =1  csm$background        lit       '0002h',
      =1  csm$purging           lit       '0004h',
```

```
        =1  csm$noswitch        lit     '0008h',
        =1  csm$suspend         lit     '0010h',
        =1  csm$abort           lit     '0020h',
        =1  csm$filefull        lit     '0040h',
        =1  csm$ctrlS           lit     '0080h',
        =1  csm$ctrlQ           lit     '0100h',
        =1  csm$ctrlP           lit     '0200h';
        =1
123   1 =1  dcl x$init$offset lit '0Ch',
        =1      x$init$pointer pointer,
        =1      x$init$ptr structure (offset word, segment word) at (@x$init$pointer),
        =1      x$init based x$init$pointer structure
        =1        (tick byte, ticks$sec byte, door byte, resrvd1 (2) byte,
        =1        nvcns byte, nccb byte, nlst byte, ccb word, lcb word);
        =1
        =1
124   1 =1  dcl lcb$structure lit 'structure (attech address, queue address,
        =1   flag byte, startcol byte, column byte, nchar byte,
        =1   mimic byte, msource byte)';
        =1
        =   /*$include (:f2:ccb.lit)*/
        =
125   1 =   dcl pd$pointer pointer;  /* double word bases for MP/M-86 data structures */
126   1 =   dcl pd$ptr structure(
        =     offset word,
        =     segment word) at (@pd$pointer);
127   1 =   dcl pd based pd$pointer pd$structure;
        =
128   1 =   dcl qd$pointer pointer;
129   1 =   dcl qd$ptr structure(
        =     offset word,
        =     segment word) at (@qd$pointer);
130   1 =   dcl qd based qd$pointer qd$structure;
        =
131   1 =   dcl md$pointer pointer;
132   1 =   dcl md$ptr structure(
        =     offset word,
        =     segment word) at (@md$pointer);
133   1 =   dcl md based md$pointer md$structure;
        =
134   1 =   dcl ms$pointer pointer;
135   1 =   dcl ms$ptr structure(
        =     offset word,
        =     segment word) at (@ms$pointer);
136   1 =   dcl ms based ms$pointer ms$structure;
        =
137   1 =   dcl sat$pointer pointer;
138   1 =   dcl sat$ptr structure(
        =     offset word,
        =     segment word) at (@sat$pointer);
139   1 =   dcl sat based sat$pointer sat$structure;
        =
140   1 =   dcl flag$pointer pointer;
141   1 =   dcl flag$ptr structure(
        =     offset word,
        =     segment word) at (@flag$pointer);
142   1 =   dcl flag based flag$pointer flag$structure;
```

```
                =
143    1        =    dcl vccb$pointer pointer;
144    1        =    dcl vccb$ptr structure (
                =        offset word,
                =        segment word) at (@vccb$pointer);
145    1        =    dcl vccb based vccb$pointer ccb$structure;
                =
146    1        =    dcl uda$pointer pointer;
147    1        =    dcl uda$ptr structure (
                =        offset word,
                =        segment word) at (@uda$pointer);
148    1        =    dcl uda based uda$pointer uda$structure;
                =
                =
                =    /* BDOS Calls */
                =
149    1        =      reboot:
                =        procedure;
150    2        =          call mon1(0,0);
151    2        =        end reboot;
                =
152    1        =      conin:
                =        procedure byte;
153    2        =          return(mon2(1,0));
154    2        =        end conin;
                =
155    1        =      co:
                =        procedure (char);
156    2        =          dcl char byte;
157    2        =          call mon1 (2,char);
158    2        =        end co;
                =
159    1        =      rawconin:
                =        procedure byte;
160    2        =          return mon2(6,0fdh);
161    2        =        end rawconin;
                =
162    1        =      constat:
                =        procedure byte;
163    2        =        return(mon2(11,0));
164    2        =        end constat;
                =
165    1        =      rawco:
                =        procedure (char);
166    2        =          dcl char byte;
167    2        =          call mon1(6,char);
168    2        =        end rawco;
                =
169    1        =      delay:
                =        procedure (num);
170    2        =          dcl num address;
171    2        =          call mon1(141,num);
172    2        =        end delay;
                =
173    1        =      print$buffer:
                =        procedure (bufferadr);
174    2        =          dcl bufferadr address;
```

```
175   2  =          call mon1 (9,bufferadr);
176   2  =        end print$buffer;
         =
177   1  =      get$version:
         =        procedure word;
178   2  =          return mon3(12,0);
179   2  =        end get$version;
         =
180   1  =      terminate:
         =        procedure;
181   2  =          call mon1(143,0);
182   2  =        end terminate;
         =
183   1  =      get$sysdat:
         =        procedure pointer;
184   2  =          return mon4(154,0);
185   2  =        end get$sysdat;
         =
186   1  =      get$currpd:
         =        procedure pointer;
187   2  =          return mon4(156,0);
188   2  =        end get$currpd;
         =
         =
         =    /* utility functions */
         =
189   1  =      crlf:
         =        procedure;
190   2  =          call co (CR);
191   2  =          call co (LF);
192   2  =        end crlf;
         =
193   1  =      print$infield:                          /* Prints 'len' # of bytes in a  */
         =                                              /* left- or right-justified field*/
         =                                              /* of 'width' dimension.         */
         =        procedure (width,justify,len,dataddr);
         =
194   2  =          dcl dataddr pointer;
195   2  =          dcl (width,len) byte,
         =              (i,justify) byte;
196   2  =          dcl char based dataddr byte;
197   2  =          dcl dat$ptr structure(
         =              offset word,
         =              segment word) at (@dataddr);
         =
198   2  =        if len <= width then do;                    /* Error Check */
200   3  =           if justify = 'r' then                   /* Right Justify */
201   3  =              do i=1 to (width-len) ;              /* Pad on the left */
202   4  =                 call co(' ');
203   4  =              end;
204   3  =              do i=1 to len ;
205   4  =                 call co (char and 7fh);           /* Print the data */
206   4  =                 dat$ptr.offset = dat$ptr.offset + 1;
207   4  =              end;
208   3  =            if justify = 'l' then                   /* Left-justified*/
209   3  =               do i = 1 to (width-len);             /* Pad on the right*/
210   4  =                  call co(' ');
```

```
211   4  =              end;
212   3  =           end;
         =
213   2  =      end print$infield;
         =
         =
         =
214   1  =      dcl hex$digit (*) byte data ('0123456789ABCDEF');
         =
215   1  =      display$hex$byte:
         =        procedure (value);
216   2  =          dcl value byte;
217   2  =          call co (hex$digit(shr(value,4)));
218   2  =          call co (hex$digit(value mod 16));
219   2  =        end display$hex$byte;
         =
220   1  =      display$hex$word:
         =        procedure (value);
221   2  =          dcl value word;
222   2  =          call display$hex$byte (high(value));
223   2  =          call display$hex$byte (low (value));
224   2  =        end display$hex$word;
         =
         =
225   1  =    print$hex$byte:                          /* Prints hex byte in a right- or */
         =                                             /* left-justified field of 'width'*/
         =                                             /* dimension.                     */
         =       procedure (width,justify,val);
226   2  =       dcl (val,width) byte,
         =           (i,justify) byte;
         =
227   2  =       if width < 2 then return;            /* Must be at least 2 for hex byte */
229   2  =       else do;
230   3  =            if justify = 'r' then                    /* Right Justify */
231   3  =            do i=1 to (width - 2) ;
232   4  =                call co(' ');                       /* Pad on the left */
233   4  =            end;
234   3  =            call display$hex$byte(val);             /* Print the digits */
235   3  =            if justify = 'l' then
236   3  =            do i=1 to (width-2) ;
237   4  =                call co(' ');                       /* Pad on the right */
238   4  =            end;
239   3  =       end;
240   2  =     end print$hex$byte;
         =
         =
241   1  =    print$hex$word:                          /* Prints hex word in a right- */
         =                                             /* or left-justified field of  */
         =                                             /* 'width' dimension.          */
         =       procedure(width,justify,val);
242   2  =       dcl val word,
         =           (justify,width) byte;
243   2  =       dcl i byte;
         =
244   2  =       if width < 4 then return;                       /* Error check */
246   2  =       else do;
247   3  =       if justify = 'r' then                 /* Field is right-justified */
```

```
248   3  =          do i=1 to (width-4) ;
249   4  =              call co(' ');                          /* Pad on the left */
250   4  =           end;
251   3  =          call display$hex$word(val);                /* Print the digits */
252   3  =        if justify = '1' then
253   3  =           do i=1 to (width-4) ;
254   4  =           call co(' ');                             /* Pad on the right */
255   4  =           end;
256   3  =        end;
             =
257   2  =     end print$hex$word;
             =
258   1  =    clear: procedure;
259   2  =    dcl i byte;
             =
260   2  =       do i = 1 to clearseq(0);            /* 1st element = counter        */
261   3  =       call rawco(clearseq(i));            /* Direct clear_screen sequence */
262   3  =       end;                                /* to terminal.                 */
263   2  =    end clear;
             =
264   1  =    home: procedure;
265   2  =    dcl i byte;
             =
266   2  =      do i = 1 to homeseq(0);             /* 1st element = counter        */
267   3  =      call rawco(homeseq(i));             /* direct home cursor sequence  */
268   3  =      end;                                /* to terminal.                 */
             =
269   2  =    end home;
             =
270   1  =    skip$lines:  procedure(numlines);
271   2  =    dcl (numlines,i) byte;
             =
272   2  =      do i = 1 to numlines;
273   3  =         call co(LF);
274   3  =      end;
275   2  =    end skip$lines;
             =
276   1  =    cons$wait: procedure;
277   2  =    dcl chr byte;
             =
278   2  =      call print$buffer(.(CR,LF,
             =                          'Type any key to leave and return to main menu.$'));
279   2  =      chr = conin;
             =
280   2  =     end cons$wait;
             =
281   1  =    aschex:                                        /* Convert ascii to hex*/
             =          procedure(num) byte;
282   2  =    dcl num byte;
             =
283   2  =        if (num > 47) and (num < 58) then do;              /* 0 - 9 range */
285   3  =            num = num - '0';
286   3  =            end;
287   2  =          else do;
288   3  =            if (num > 64) and (num < 71) then do;        /* A - F range */
290   4  =                num = num - 55;
291   4  =                end;
```

```
292   3  =            else do;
293   4  =                if (num > 96) and (num < 103) then do;          /* a - f range */
295   5  =                    num = num - 87;
296   5  =                    end;
      =                else
297   4  =                    num = 015h;                                 /* Error -> Default */
298   4  =                end;
299   3  =              end;
300   2  =          return (num);
      =
301   2  =    end aschex;
      =
302   1  =    get$intrval: procedure word;
303   2  =    dcl (chr,chr1) byte,
      =         ticdelay word;
      =
304   2  =        chr = conin;
305   2  =        if (chr <> LF) and (chr <> CR) then do;         /* It's not a default */
307   3  =            chr = aschex(chr);                         /* Get true hex version */
308   3  =            chr1 = conin;                               /* wait for CR or LF */
309   3  =            if (chr1 <> LF) and (chr1 <> CR) then do;       /* 2nd hex digit */
311   4  =                chr = shl(chr,4) + aschex(chr1);
312   4  =                chr1 = conin;                               /* Get this CR LF */
313   4  =                end;
314   3  =            end;
      =        else
315   2  =            chr = 01h;                                      /* Default value */
316   2  =        ticdelay = chr * (sd.tickspersec);          /* Convert to system ticks */
317   2  =        return (ticdelay);
      =
318   2  =     end get$intrval;
      =
      =
319   1  =    disp$mainhdr: procedure;                                /* Main Menu Display */
      =
320   2  =       call home; call clear;
322   2  =       call crlf; call crlf;
324   2  =       call print$infield(34,'r',14,@('Which Option ?'));
325   2  =       call crlf; call crlf;
327   2  =       call print$infield(33,'r',7,@('H (elp)'));
328   2  =       call crlf;
329   2  =       call print$infield(35,'r',9,@('M (emory)'));
330   2  =       call crlf; call print$infield(37,'r',11,@('O (verview)'));
332   2  =       call crlf; call print$infield(44,'r',18,@('P (rocesses - all)'));
334   2  =       call crlf; call print$infield(35,'r',9,@('Q (ueues)'));
336   2  =       call crlf; call print$infield(43,'r',17,@('U (ser processes)'));
338   2  =       call crlf; call print$infield(33,'r',7,@('E (xit)'));
340   2  =       call crlf; call crlf; call print$infield(26,'r',2,@('->'));
      =
343   2  =    end disp$mainhdr;
      =
344   1  =    print$opt$err: procedure;
345   2  =        call print$buffer(.(CR,LF,'     illegal command tail.$'));
346   2  =        call crlf;
347   2  =        call terminate;
348   2  =        end print$opt$err;
      =
```

```
349   1   =   display$help: procedure;
              =
350   2   =   call home; call clear;
352   2   =   call crlf; call crlf;
354   2   =   call crlf; call print$infield(42,'r',23,@('VALID SYSTAT COMMANDS :'));
356   2   =   call crlf; call crlf; call print$infield(25,'r',6,@('SYSTAT'));
359   2   =   call crlf; call print$infield(34,'r',15,@('SYSTAT [OPTION#]'));
361   2   =   call crlf; call print$infield(36,'r',17,@('SYSTAT [OPTION C]'));
363   2   =   call crlf; call print$infield(39,'r',20,@('SYSTAT [OPTION C ##]'));
365   2   =   call crlf; call crlf; call print$infield(28,'r',9,@('- where -'));
368   2   =   call crlf; call crlf;
370   2   =   call print$infield(44,'r',25,@('-> C = continuous display'));
371   2   =   call crlf; call print$infield(47,'r',26,@('-> ## = 1-2 digit hex timer.'));
373   2   =   call crlf; call crlf; call print$infield(30,'r',11,@('-> OPTION ='));
376   2   =   call crlf; call print$infield(55,'r',33,@('M(emory)  P(rocesses)  O(verview)'));
378   2   =   call crlf; call print$infield(56,'r',33,@('U(ser Processes)  Q(ueues) H(elp)'));
380   2   =   call crlf;
              =
381   2   =   if not(specified) then
382   2   =   call cons$wait;
              =   else
383   2   =   call terminate;
              =
384   2   =   end display$help;
              =
              =
              =
385   1   =   prntheader: procedure;                    /* Used in Process Data Display */
              =
386   2   =      call home; call clear;
388   2   =      call print$infield(20,'r',18,@('Virtual    Process'));
389   2   =      call crlf;
390   2   =      call print$infield(20,'r',18,@('Console     Name  '));
391   2   =      call print$infield(37,'r',35,@('Flag  Prior   Status       Resource'));
392   2   =      call crlf;
393   2   =      call print$infield(20,'r',18,@('-------    -------'));
394   2   =      call print$infield(37,'r',35,@('----  -----   ------       --------'));
              =
395   2   =      end prntheader;
              =
              =
              =
396   1   =      disp$status: procedure(stat);          /* Prints formatted status field */
397   2   =        dcl stat byte;                      /* of Process display.           */
              =
398   2   =            if (stat >=0) and (stat < 11) then do;
400   3   =            do case stat;
401   4   =                    call print$infield(13,'l',5,@('READY'));
402   4   =                    call print$infield(13,'l',4,@('POLL'));
403   4   =                    call print$infield(13,'l',5,@('DELAY'));
404   4   =                    call print$infield(13,'l',4,@('SWAP'));
405   4   =                    call print$infield(13,'l',9,@('TERMINATE'));
406   4   =                    call print$infield(13,'l',5,@('SLEEP'));
407   4   =                    call print$infield(13,'l',7,@('READING'));
408   4   =                    call print$infield(13,'l',7,@('WRITING'));
409   4   =                    call print$infield(13,'l',9,@('FLAG WAIT'));
410   4   =                    call print$infield(13,'l',8,@('CIO WAIT'));
```

```
411   4  =                    call print$infield(13,'1',3,@('SYNCING'));
412   4  =               end;
413   3  =               end;
      =               else
414   2  =                  call print$infield(13,'1',5,@('ERROR'));
      =
415   2  =        end disp$status;
      =
      =
      =
416   1  =       disp$resource: procedure(stats,rsrce);/* Prints formatted resource field */
417   2  =          dcl stats byte;                    /* of Process display.            */
418   2  =          dcl (rsrce,offs) word;
419   2  =          dcl (count,notfound) byte;         /* For flag table traversal       */
      =
420   2  =          if (stats >=0) and (stats < 11) then do;
422   3  =          do case stats;
423   4  =                  call print$infield(13,'1',3,@('CPU          '));   /* Case 0 */
424   4  =                  do;                                                /* Case 1 */
425   5  =                   call print$infield(8,'1',8,@('DEVICE #'));
426   5  =                   call print$hex$byte(4,'1',low(rsrce));
427   5  =                  end;
428   4  =                  do;                                                /* Case 2 */
429   5  =                   call print$infield(8,'1',8,@('TICKS = '));
430   5  =                   call print$hex$word(4,'1',rsrce);
431   5  =                  end;
432   4  =                   call print$infield(13,'1',7,@('SWAPERR     '));  /* Case 3 */
433   4  =                   call print$infield(13,'1',3,@('CPU          '));  /* Case 4 */
434   4  =                  do;
435   5  =                    if rsrce = sd.rlr then                           /* Case 5 */
436   5  =                       call print$infield(13,'1',10,@('READY LIST'));
437   5  =                    else if rsrce = sd.dlr then
438   5  =                       call print$infield(13,'1',10,@('DELAY LIST'));
439   5  =                    else if rsrce = sd.drl then
440   5  =                       call print$infield(13,'1',10,@('DISPATCHER'));
441   5  =                    else if rsrce = sd.plr then
442   5  =                       call print$infield(13,'1',9,@('POLL LIST'));
      =                    else
443   5  =                       call print$infield(13,'1',5,@('OTHER'));
444   5  =                  end;
445   4  =                  do;                                                /* Case 6 */
446   5  =                   qd$ptr.offset = rsrce;
447   5  =                   call print$in$field(12,'1',pnamsiz,@qd.name);
448   5  =                  end;
449   4  =                  do;                                                /* Case 7 */
450   5  =                   qd$ptr.offset = rsrce;
451   5  =                   call print$infield(12,'1',pnamsiz,@qd.name);
452   5  =                  end;
453   4  =                  do;                                                /* Case 8 */
454   5  =                    call co(023h);                                /* A '#' sign */
455   5  =                    if rsrce <> 0ffffh then
456   5  =                    call print$hex$byte(12,'1',low(rsrce));
      =                    else
457   5  =                    call print$in$field(12,'1',1,@('?'));
458   5  =                  end;
459   4  =                  do;                                                /* Case 9 */
460   5  =                  call print$infield(10,'1',9,@('CONSOLE #'));
```

```
                call display$hex$byte(low(rsrce));
                end;
                call print$infield(13,'1',13,0('              ')); /* Case 10*/
             end; /* case */
          end;
          else                                        /* Invalid Status Value */
          call print$infield(12,'1',5,0('ERROR'));
        end disp$resource;

dcl pd$list (64) structure (          /* Stores fields of successive pd's.*/
                     cns  byte,
                     name (8) byte,
                     flag word,
                     prior byte,
                     stat byte ,
                     resource word);

dcl link$list (64) word;

display$proc: procedure(lnkfield);
   dcl lnkfield byte;          /* True = user proc's only,False = all proc's*/
   dcl (char,temp) byte;       /* Temp controls continuous printout */
   dcl (k,n) byte,
       (notfound,i) byte;

   if not(specified) then do;
   call crlf;
   call print$buffer(.(CR,LF,'    Continuous Display?$'));
   char = conin;
   if (char = 'y') or ( char = 'Y') then do;
       repeat = true;
       call print$buffer(.(CR,LF,'Time Interval (in hex) :$'));
       intrval = get$intrval;

       end;
   end;

   temp = true;
   call prntheader;

   do while (temp or repeat);            /* Display until user hits any key */
   disable;          /* critical section required to obtain list*/
   temp = false;
   n = -1;
   pd$ptr.offset = sd.thrdrt;                  /* Start at thread list root. */
getpds:                                          /* Put all pd's on the list */
   do while (pd$ptr.offset <> 0) and (n <> maxpd);
      n = n + 1;
      if not(lnkfield) or (pd.mem <> 0) then /* Is it user processes only ? */
      do;                            /* Either display all processes anyway or */
                                     /* or this is a valid user process.       */
      pd$list(n).cns = pd.cns;
       do i= 0 to pnamsiz-1;
          pd$list(n).name(i) = pd.name(i);
       end;
      pd$list(n).flag = pd.flag;
      pd$list(n).prior = pd.prior;
```

```
503   5  =          pd$list(n).stat = pd.stat;
         =                                  /* Use sysdat info to determine what each*/
         =                                  /* process is waiting for or using. Get  */
         =                                  /* Queue names,flag #'s,system ticks,list*/
         =                                  /* names and device or console #'s. Must */
         =                                  /* be done in critical region.           */
504   5  =          i = pd.stat;            /* Save this to determine resource .     */
505   5  =          if (i = 1) or (i = 2 )then
506   5  =             pd$list(n).resource = pd.wait;  /* Device # or # of ticks delayed */
         =          else
507   5  =          if i = 5 then              /* Process is sleeping - find out which list*/
508   5  =               do;
509   6  =               uda$ptr.segment = pd.uda;
510   6  =               uda$ptr.offset = 0;
511   6  =               pd$list(n).resource = uda.dparam;
512   6  =               end;
         =          else
513   5  =          if (i = 6) or (i = 7) then   /* Process is reading from or writing to */
514   5  =             do;                       /* a queue. Get address of queue name.   */
515   6  =               uda$ptr.segment = pd.uda;
516   6  =               uda$ptr.offset = 0;
517   6  =               if i = 6 then
518   6  =               pd$list(n).resource = (uda.dparam) - 12h;
         =               else
519   6  =               pd$list(n).resource = (uda.dparam) - 14h;
520   6  =             end;
         =          else
521   5  =          if i = 8 then              /* Process is waiting on a flag, get the #*/
522   5  =             do;
523   6  =             pd$list(n).resource = 0ffffh;         /* Remains if no flag found .*/
524   6  =             flag$ptr.offset = sd.flags;
525   6  =             notfound = true;
526   6  =             k = 0;                                        /* Flag counter */
527   6  =             do while notfound;
528   7  =                if (flag.pd <> pd$ptr.offset) then do;
530   8  =                   k = k+ 1;
531   8  =                   flag$ptr.offset = flag$ptr.offset + flaglen;
532   8  =                   if k > sd.nflags then                    /* End of table */
533   8  =                      notfound = false;
534   8  =                   end;
         =                else                                     /* This is the flag*/
535   7  =                   do;
536   8  =                   pd$list(n).resource = k;
537   8  =                   notfound = false;
538   8  =                   end;
539   7  =                end;
540   6  =             end;
         =          else
541   5  =          if i = 9 then do;
543   6  =             pd$list(n).resource = pd.cns;
544   6  =             end;
         =        end;                                          /* Valid processes*/
         =        else
546   4  =             n = n-1;                              /* Ignore system processes */
547   4  =          pd$ptr.offset = pd.thread;                  /* Get the next process*/
548   4  =      end getpds;
         =
```

```
549   3  =     if n = -1 then return;                          /* stop here if no pds */
           =
551   3  =   enable;
552   3  =   call home; call skip$lines(3);                    /*  enables interrupts */
554   3  =   printpds:
           =      do k = 0 to n;
555   4  =         if k <> 0 then call crlf;
557   4  =         call print$hex$byte(7,'r',pd$list(k).cns);/* print virtual console #*/
558   4  =         call print$buffer(.('      $'));
559   4  =         call print$infield(9,'l',pnamsiz,0pd$list(k).name); /* print name */
560   4  =         call print$hex$word(8,'l',pd$list(k).flag);
561   4  =         call print$hex$byte(7,'l',pd$list(k).prior);  /* print its priority */
562   4  =         call disp$status(pd$list(k).stat);          /* print process status */
563   4  =         call disp$resource(pd$list(k).stat,pd$list(k).resource);
           =                                                      /* print resource */
           =
564   4  =       end printpds;
           =
565   3  =       if n < 21 then do;
567   4  =       do k = n to 18;
568   5  =          call print$buffer(.(CR,LF,
           =          '                                                    $'));
569   5  =       end;
570   4  =       end;
           =
           =
571   3  =   if constat then do;                             /* Check for User interrupt*/
573   4  =       repeat = false;
574   4  =       char = conin;                                    /* Swallow stop char */
575   4  =       end;
576   3  =   if repeat then do;                               /* If still going,delay it */
578   4  =       call delay(intrval);                         /* and go back to loop top */
579   4  =       if (n >= 20) then
580   4  =          call prntheader;
581   4  =       end;
582   3  =   else do;
583   4  =       if not(specified) then                       /* if not comline args then */
584   4  =       call cons$wait;                              /* let them get to main menu*/
           =       else
585   4  =       call terminate;
586   4  =       end;
           =
587   3  =   end;/*while loop*/
           =
588   2  =   end display$proc;
           =
           =
589   1  =   dcl sortrecd structure ( name (8) byte,         /* Stores data when sorting*/
           =                           start    word,
           =                           pd       word,
           =                           len      word,
           =                           cns      byte );
           =
590   1  =   dcl sortarray (80) structure (                  /* For sorting and printing*/
           =                           name (8) byte,
           =                           start    word,
           =                           pd       word,
```

```
                                len       word,
                                cns       byte );

591   1   =   dcl sharearray (80) structure (                    /* For Shared Code List */
                                name (8) byte,
                                start     word,
                                disk      byte,
                                user      byte,
                                len       word,
                                cns       byte );

592   1   =   disp$memhdr :  procedure;

593   2   =     call home; call clear;
595   2   =     call print$buffer(.('Process   Virtual                    | $'));
596   2   =     call print$buffer(.('Process   Virtual  $'));
597   2   =     call print$buffer(.(CR,LF,' Name     Console  PD#   Start  Len  | $'));
598   2   =     call print$buffer(.(' Name     Console   PD#    Start   Len $'));
599   2   =     call print$buffer(.(CR,LF,'--------  -------  ----  -----  ----- | $'));
600   2   =     call print$buffer(.(' -------  -------   ----  -----   ---- $'));

601   2   =   end disp$memhdr;

602   1   =   print$sorted:                        /* Prints two columns of memory data */
                  procedure(cnt,scnt);            /* Uses sorted array of structures.  */
603   2   =     dcl (cnt,scnt) byte,              /* cnt: regular mem,scnt: shared mem */
                  (m,q,k)    byte;

604   2   =     call home;
605   2   =     call skip$lines(3);

606   2   =     if (cnt > 1) then do;                  /* Must have two per line here */
608   3   =     do m = 0 to ((cnt/2)-1) ;                  /* If odd #, hold last one */
609   4   =         call crlf;              /* Print 2 columns,ascending values*/
610   4   =         k = m;
611   4   =         do q = 1 to 2 ;
612   5   =             call print$infield(12,'1',pnamsiz,@sortarray(k).name);
613   5   =             if sortarray(k).cns = 030h then
614   5   =                call print$buffer(.('       $'));
615   5   =             else call print$hex$byte(7,'1',sortarray(k).cns);
616   5   =             call print$hex$word(6,'1',sortarray(k).pd);
617   5   =             call print$hex$word(7,'1',sortarray(k).start);
618   5   =             call print$hex$word(5,'1',sortarray(k).len);
619   5   =             if (q mod 2) <> 0 then
620   5   =                call print$buffer(.('|  $'));
621   5   =             k = k + (cnt/2);              /* Go to other half of the array */
622   5   =             end;
623   4   =         end;
624   3   =       end;

625   2   =       if (cnt mod 2) <> 0 then do;           /* Only put one on last line */
                                                           /* Print blanks on left  */
627   3   =          call print$buffer(.(CR,LF,
                                                                        |  $'));
628   3   =          call print$infield(12,'1',pnamsiz,@sortarray(cnt-1).name);
```

```
629   3  =           if sortarray(cnt-1).cns = 0a0h then
630   3  =              call print$buffer(.('       $'));
      =  =           else
631   3  =              call print$hex$byte(7,'1',sortarray(cnt-1).cns);
632   3  =           call print$hex$word(6,'1',sortarray(cnt-1).pd);
633   3  =           call print$hex$word(7,'1',sortarray(cnt-1).start);
634   3  =           call print$hex$word(5,'1',sortarray(cnt-1).len);
635   3  =        end;
      =  =                                          /* Print Shared Code,if any */
636   2  =        if (scnt > 0) then do;
638   3  =           call print$buffer(.(CR,LF,
      =  =                                  'Shared  Code  List                    $'));
639   3  =           call print$buffer(.('                $'));
640   3  =           end;
      =  =
641   2  =        if (scnt > 1) then do;
643   3  =           do m = 0 to ((scnt/2) - 1);
644   4  =              call crlf;
645   4  =              k = m;
646   4  =              do q = 1 to 2;
647   5  =                 call print$infield(12,'1',pnamsiz,@sharearray(k).name);
648   5  =                 call print$hex$byte(7,'1',sharearray(k).cns);
649   5  =                 call print$hex$byte(2,'1',sharearray(k).disk);
650   5  =                 call print$hex$byte(4,'1',sharearray(k).user);
651   5  =                 call print$hex$word(7,'1',sharearray(k).start);
652   5  =                 call print$hex$word(5,'1',sharearray(k).len);
653   5  =                 if (q mod 2) <> 0 then
654   5  =                     call print$buffer(.('|  $'));
655   5  =                 k = k + (scnt/2);
656   5  =                 end;
657   4  =              end;
658   3  =             end;
      =  =
659   2  =         if (scnt > 0) then do;
      =  =
661   3  =              if (scnt mod 2) <> 0 then do;      /* Odd # of md's.          */
      =  =                                                /* Put just 1 on last line*/
663   4  =                 call print$buffer(.(CR,LF,
      =  =                                                          '|  $'));
664   4  =                 call print$infield(12,'1',pnamsiz,@sharearray(scnt-1).name);
665   4  =                 call print$hex$byte(7,'1',sharearray(scnt-1).cns);
666   4  =                 call print$hex$byte(2,'1',sharearray(scnt-1).disk);
667   4  =                 call print$hex$byte(4,'1',sharearray(scnt-1).user);
668   4  =                 call print$hex$word(7,'1',sharearray(scnt-1).start);
669   4  =                 call print$hex$word(5,'1',sharearray(scnt-1).len);
670   4  =                 end;
      =  =
671   3  =              scnt = scnt + 1;                   /* Count its heading    */
672   3  =           end;
      =  =
673   2  =        if ((cnt+scnt)/2) < 21 then do;          /* Clear rest of screen */
675   3  =        do k = ((cnt+scnt)/2) to 18;
676   4  =           call print$buffer(.(CR,LF,
      =  =                                                                  '$'));
677   4  =           call print$buffer(.('           $'));
678   4  =           end;
679   3  =        end;
```

```
        =
        =
680   2 =   end print$sorted;
        =
        =
681   1 =   display$mem:
        =           procedure;
682   2 =     dcl (i,mdcnt,pdcnt) byte;              /* Mdcnt = memory descriptor count*/
683   2 =     dcl (x,y) byte,                        /* Pdcnt = process descr. count   */
        =         (cont,scdcnt) byte,                    /* cont = boolean 'continue' */
        =         n integer,
        =         (temp,chr) byte,
        =         savmau word;
        =
684   2 =     if not(specified) then do;
        =                                        /* Does User want continuous display?*/
686   3 =     call print$buffer(.(CR,LF,'     Continuous Display?  $'));
687   3 =     chr = conin;
688   3 =     if (chr = 'y') or ( chr = 'Y') then do;
690   4 =        repeat = true;
691   4 =        call print$buffer(.(CR,LF,'    Time Interval (in hex):  $'));
692   4 =        intrval = get$intrval;               /* Get hex value from terminal */
693   4 =        end;
694   3 =     end;
695   2 =     temp = true;
        =
696   2 =     call home; call clear;
698   2 =     call disp$memhdr;
699   2 =     do while (temp or repeat);                   /* Display at least once */
700   3 =     temp = false;
701   3 =     do pdcnt = 0 to maxpd;
702   4 =        link$list(pdcnt) = 0;
703   4 =        end;
704   3 =     n = -1;
705   3 =     pdcnt = 0;
        =
        =
706   3 =     disable;                  /* Critical section required to obtain list */
        =
707   3 =   getmemowners:
        =     pd$ptr.offset = sd.thrdrt;                 /* Start at Thread Root */
708   3 =       do while (pd$ptr.offset <> 0) and (pdcnt <> maxpd);
709   4 =          if pd.mem <> 0 then do;                 /* If it owns memory,   */
711   5 =            link$list(n:=n+1) = pd$ptr.offset;    /* put it on the list.  */
712   5 =            pdcnt = pdcnt + 1;
713   5 =          end;
714   4 =          pd$ptr.offset = pd.thread;              /* Go to next process   */
715   4 =        end;
716   3 =     if pdcnt = 0 then return;
        =
718   3 =   getmds:
        =     mdcnt = 0;
719   3 =     do i = 0 to (pdcnt-1);
720   4 =        pd$ptr.offset = link$list(i);           /* Reset Proc Descriptor */
721   4 =        ms$ptr.offset = pd.mem;
722   4 =        cont = true;
723   4 =        do while (cont and (mdcnt <= maxmd));
```

```
124   5  =            sortarray(mdcnt).pd = pd$ptr.offset;   /* Set proc descripter  */
125   5  =            cont = false;
126   5  =            md$ptr.offset = ms.mau;                /* Md is on MAU         */
127   5  =            sortarray(mdcnt).start = md.start;
128   5  =            sortarray(mdcnt).len = md.length;
129   5  =            sortarray(mdcnt).cns = pd.cns;
130   5  =            do x = 0 to 7;                            /* Get  owner's name */
131   6  =               sortarray(mdcnt).name(x) = pd.name(x); /*  A byte at a time */
132   6  =               end;
133   5  =            if ms.link <> 0 then do;                  /* More md's for this   */
       =                                                      /* process ?            */
135   6  =               savmau = ms.mau;
136   6  =               ms$ptr.offset = ms.link;
137   6  =               do while (ms.mau = savmau) and (ms.link <> 0);
138   7  =                  savmau = ms.mau;                   /* Look for a different */
139   7  =                  ms$ptr.offset = ms.link;           /* partition,same pd.   */
140   7  =                  end;
141   6  =               if (savmau <> ms.mau) then            /* Check if same mau    */
142   6  =                  cont = true;                    /* If not, go get another md*/
143   6  =               end;
144   5  =             mdcnt = mdcnt + 1;
       =
145   5  =          end;                              /* while cont : a single pd's memory*/
       =
146   4  =       end;                                                 /* All pd's memory*/
       =                                          /* That's all the used memory. Now go  */
       =                                          /* to the Memory Free List.            */
147   3  =         md$ptr.offset = sd.mfl;
148   3  =         do while md$ptr.offset <> 0 and (mdcnt < maxmd);
149   4  =            do x = 0 to 7;
150   5  =               sortarray(mdcnt).name(x) = freename(x); /*Use "FREE" as name*/
151   5  =            end;
152   4  =            sortarray(mdcnt).start = md.start;
153   4  =            sortarray(mdcnt).pd = 0;
154   4  =            sortarray(mdcnt).len = md.length;
155   4  =            sortarray(mdcnt).cns = 030h;
156   4  =            mdcnt = mdcnt+1;
157   4  =            md$ptr.offset = md.link;
158   4  =         end;
       =                                                    /* Get shared code  */
159   3  =       getshared:                                   /* Free and used    */
       =            n = -1;
160   3  =            pdcnt = 0;
161   3  =            scdcnt = 0;
162   3  =            pd$ptr.offset = sd.slr;                /* Top of shared list*/
163   3  =            do while (pd$ptr.offset <> 0) and (pdcnt < maxpd);
164   4  =               link$list(n:=n + 1) = pd$ptr.offset;
165   4  =               pdcnt = pdcnt + 1;
166   4  =               pd$ptr.offset = pd.thread;
167   4  =               end;
168   3  =            if (pdcnt > 0) then do;
170   4  =               do i = 0 to (pdcnt-1);
171   5  =                  pd$ptr.offset = link$list(i);
172   5  =                  ms$ptr.offset = pd.mem;
173   5  =                  cont = true;
174   5  =                  do while (cont and (scdcnt < maxmd));
```

```
775   6  =                    cont = false;
776   6  =                    sharearray(scdcnt).cns = pd.cns;
777   6  =                    sharearray(scdcnt).disk = pd.ldsk;
778   6  =                    sharearray(scdcnt).user = pd.luser;
779   6  =                    md$ptr.offset = ms.mau;
780   6  =                    sharearray(scdcnt).start = md.start;
781   6  =                    sharearray(scdcnt).len = md.length;
782   6  =                    do x = 0 to (pnamsiz - 1);
783   7  =                       sharearray(scdcnt).name(x) = pd.name(x);
784   7  =                       end;
785   6  =                    if (ms.link <> 0) then do;
787   7  =                       savmau = ms.mau;
788   7  =                       ms$ptr.offset = ms.link;
789   7  =                       do while (ms.mau = savmau) and (ms.link <> 0);
790   8  =                          savmau = ms.mau;
791   8  =                          ms$ptr.offset = ms.link;
792   8  =                          end;
793   7  =                       if (ms.mau <> savmau) then
794   7  =                            cont = true;
795   7  =                       end;
796   6  =                    scdcnt = scdcnt + 1;
797   6  =              end;
798   5  =           end;
799   4  =        end;
           =
800   3  =        enable;                                  /* End critical section */
           =
           =                                          /* Now sort the list of partitions */
801   3  =     sortmds :
           =              do x = 0 to (mdcnt-1);
802   4  =           sortrecd.start = sortarray(x).start;
803   4  =              do i = 0 to 7;
804   5  =              sortrecd.name(i) = sortarray(x).name(i);
805   5  =              end;
806   4  =           sortrecd.len = sortarray(x).len;
807   4  =              sortrecd.pd = sortarray(x).pd;
808   4  =           sortrecd.cns = sortarray(x).cns;
809   4  =              y = x;
810   4  =        find : do while (y > 0) and ( sortarray(y-1).start > sortrecd.start);
811   5  =              sortarray(y).start = sortarray(y-1).start;
812   5  =              do i = 0 to 7;
813   6  =              sortarray(y).name(i) = sortarray(y-1).name(i);
814   6  =           end;
815   5  =              sortarray(y).len = sortarray(y-1).len;
816   5  =              sortarray(y).pd = sortarray(y-1).pd;
817   5  =           sortarray(y).cns = sortarray(y-1).cns;
818   5  =              y = y-1;
819   5  =              end find;
           =
820   4  =              sortarray(y).start = sortrecd.start;
821   4  =              do i = 0 to 7;
822   5  =              sortarray(y).name(i) = sortrecd.name(i);
823   5  =           end;
824   4  =              sortarray(y).len = sortrecd.len;
825   4  =              sortarray(y).pd = sortrecd.pd;
826   4  =           sortarray(y).cns = sortrecd.cns;
827   4  =              end sortmds;
```

```
                =
                =
                =
828    3        =     call print$sorted(mdcnt,scdcnt);              /* Print the sorted list */
829    3        =     if constat then do;                           /* Want to loop again ?  */
831    4        =        repeat = false;
832    4        =        chr = conin;                               /* Swallow stop char     */
833    4        =        end;
834    3        =     if repeat then do;                          /* Keep looping,delay first */
836    4        =        if ((mdcnt / 2) >= 20) then
837    4        =           call disp$menuhdr;
838    4        =        call delay(intrval);
839    4        =        end;
840    3        =     else do;
841    4        =        if not(specified) then                     /* Get back to main menu*/
842    4        =        call cons$wait;
                =        else                                       /* Skip menu, -> system */
843    4        =        call terminate;
844    4        =        end;
845    3        =      end;/*while*/
                =
846    2        =  end display$mem;
                =
                =
847    1        =  dcl qlist (64) structure (
                =                  name (8) byte,
                =                  nmsgs word,
                =                  msglen word,
                =                  msgcnt word,
                =                  nq word,
                =                  dq word,
                =                  owner (8) byte,
                =                  flags byte);
                =
848    1        =  print$qhdr: procedure;                             /* Print Queue Heading */
                =
849    2        =     call home; call clear;
851    2        =     call print$buffer(.('   NAME     NMSGS   MSGLEN  MSGCNT  $'));
852    2        =     call print$buffer(.('READER   WRITER   NX-OWNER  FLAGS  $'));
853    2        =     call crlf;
854    2        =     call print$buffer(.(' --------  ------  ------  ------  $'));
855    2        =     call print$buffer(.('------   ------   --------  ----- $'));
856    2        =     call crlf;
                =
857    2        =  end print$qhdr;
                =
                =
858    1        =  display$flag$status:                           /* Print Queue Flag Status*/
                =        procedure(flag);
859    2        =    dcl prev boolean;
860    2        =    dcl (i,flag,cnt) byte;
                =
861    2        =    prev = false;
862    2        =    i = 1;
863    2        =    cnt = 0;
                =
864    2        =    do while (i <= maxqflags) ;
```

```
865   3  =     if (flag and i) <> 0 then do;
867   4  =        if prev then do;                        /* Take care of formatting */
869   5  =           call co(',');
870   5  =           cnt = cnt + 1;                       /* Count all chars         */
871   5  =           end;
872   4  =        call display$hex$byte(i);
873   4  =        cnt = cnt + 2;
874   4  =        prev = true;
875   4  =        end;
876   3  =     i = i*2;                                /* Flags are 1 bit each       */
877   3  =     end;
878   2  =     do i = cnt to 14;                       /* Clear previous line's flags */
879   3  =        call co(' ');
880   3  =     end;
      =
881   2  =  end display$flag$status;
      =
      =
882   1  =  display$queue:
      =           procedure;
883   2  =  ucl temp byte,
      =         n integer,
      =         (qdcnt,cont) byte,
      =         (chr,i) byte;
      =
884   2  =     if not(specified) then do;
886   3  =     call crlf;
887   3  =     call print$buffer(.(CR,LF,'    Continuous Display? $'));
888   3  =     chr = conin;
889   3  =     if (chr = 'y') or (chr = 'Y') then do;
891   4  =         repeat = true;
892   4  =         call print$buffer(.(CR,LF,'Time Interval (in hex) : $'));
893   4  =         intrval = get$intrval;
894   4  =         end;
895   3  =     end;
      =
896   2  =     temp = true;
897   2  =     call print$qhdr;
      =
898   2  =     do while ( temp or repeat);
899   3  =     temp = false;
900   3  =     qdcnt = 0;
901   3  =     call home; call skip$lines(2);
      =
903   3  =   getqueues:
      =       disable;                                  /* Begin Critical Section */
904   3  =     qd$ptr.offset = sd.qlr;
905   3  =     do while ((qdcnt < maxqueues) and (qd$ptr.offset <> 0)) ;
906   4  =       do i = 0 to pnamsiz-1;
907   5  =       qlist(qdcnt).name(i)= qd.name(i);
908   5  =       end;
909   4  =       qlist(qdcnt).nmsgs = qd.nmsgs;
910   4  =       qlist(qdcnt).msglen = qd.msglen;
911   4  =       qlist(qdcnt).msgcnt = qd.msgcnt;
912   4  =       qlist(qdcnt).nq = qd.nq;
913   4  =       qlist(qdcnt).dq = qd.dq;
914   4  =       if ((qd.flags mod 2) <> 0) then do;          /* It's an MX queue */
```

```
916   5  =           pd$ptr.offset = qd.buffer;
917   5  =           if pd$ptr.offset <> 0 then                    /* It has an owner */
918   5  =              do i = 0 to pnamsiz -1;
919   6  =                 qlist(qdcnt).owner(i) = pd.name(i);
920   6  =              end;
      =  =           else                                          /* No one owns it now */
921   5  =              do i = 0 to pnamsiz-1;                     /* Print blanks       */
922   6  =                 qlist(qdcnt).owner(i) = ' ';
923   6  =              end;
924   5  =         end;
      =  =        else                                           /* It's not an MX queue*/
925   4  =             do i = 0 to pnamsiz-1;
926   5  =                qlist(qdcnt).owner(i) = ' ';
927   5  =             end;
928   4  =        qlist(qdcnt).flags = qd.flags;
929   4  =        qdcnt = qdcnt + 1;
930   4  =        qd$ptr.offset = qd.link;
931   4  =       end;
932   3  =      enable;                                          /* End critical section */
      =  =
933   3  =      print$qds:                                       /* Print the Queue info */
      =  =        do i = 0 to qdcnt-1;
934   4  =        call print$buffer(.(' $'));
935   4  =        call print$infield(11,'l',pnamsiz,@qlist(i).name);/* Queue Name   */
936   4  =        call print$hex$word(8,'l',qlist(i).nmsgs);       /* Number of Msgs*/
937   4  =        call print$hex$word(8,'l',qlist(i).msglen);      /* Message Length*/
938   4  =        call print$hex$word(7,'l',qlist(i).msgcnt);      /* Message Count */
939   4  =        pd$ptr.offset = qlist(i).dq;
940   4  =        if pd$ptr.offset <> 0 then
941   4  =             call print$infield(9,'l',pnamsiz,@pd.name);
      =  =        else
942   4  =             call print$buffer(.('          $'));
943   4  =        pd$ptr.offset = qlist(i).nq;
944   4  =        if pd$ptr.offset <> 0 then
945   4  =             call print$infield(9,'l',pnamsiz,@pd.name);
      =  =        else
946   4  =             call print$buffer(.('          $'));
947   4  =        call print$infield(10,'l',pnamsiz,@qlist(i).owner);   /* Print it */
948   4  =        call display$flag$status(qlist(i).flags);      /* Print Flag Value*/
949   4  =        call crlf;
950   4  =             end;
      =  =
      =  =                                                       /* Print the Flag Key */
951   3  =      call print$buffer(.('     Flag Values :  1 = MX, 2 = NO DELETE, $'));
952   3  =      call print$buffer(.('4 = NOT USER WRITEABLE, $'));
953   3  =      call crlf;
954   3  =      call print$buffer(.('     8 = RSP, 10 = FROM QD TABLE, 20 = RPL QUEUE, $'));
955   3  =      call print$buffer(.('40 = DEVICE QUEUE.$'));
956   3  =      qdcnt = qdcnt - 1;                               /* To normalize */
      =  =
957   3  =      if (qdcnt < 22) then do;
959   4  =         do i = qdcnt to  17;
960   5  =           call print$buffer(.(CR,LF,
      =  =                                                                    $'));
961   5  =              end;
962   4  =         end;
      =  =
```

```
 963   3  =     if constat then do;                          /* Check for continue */
 965   4  =        repeat = false;
 966   4  =        chr = conin;
 967   4  =        end;
 968   3  =     if repeat then do;                        /* Keep going,delay first */
 970   4  =        call delay(intrval);
 971   4  =        if (qdcnt >= 19) then do;
 973   5  =        call print$qhdr;
 974   5  =        end;
 975   4  =        end;
 976   3  =     else do;                                    /* Stop display       */
 977   4  =        if not(specified) then                   /* Go to main menu    */
 978   4  =        call cons$wait;
       =        else                                     /* Go back to system  */
 979   4  =        call terminate;
 980   4  =        end;
 981   3  =     end; /* while loop */
       =
 982   2  =  end display$queue;
       =
       =
       =
 983   1  =  dcl driv$letter (*) byte data ('ABCDEFGHIJKLMNOP');
       =
 984   1  =  display$gen: procedure(vers);                 /* Overview Display Routine*/
 985   2  =     dcl vers byte;
 986   2  =     dcl (count,qsize) word;
 987   2  =     dcl mode word;
       =
       =
 988   2  =     call home; call clear;
 990   2  =     pd$pointer = get$currpd;                 /* Get PD for Current Process*/
 991   2  =     vccb$ptr.offset = sd.ccb + (pd.cns * size(vccb));
       =
 992   2  =     call crlf; call print$infield(42,'r',17,0('Default Disk =    '));
 994   2  =     call co(driv$letter(pd.dsk));
 995   2  =     call co(':');
       =
 996   2  =     call crlf; call print$infield(42,'r',24,0('Default User Number =    '));
 998   2  =     call display$hex$byte(pd.user);
       =
 999   2  =     call crlf; call print$infield(42,'r',20,0('Default Printer =    '));
1001   2  =     if (vers) then                                   /* It's MPM-86 */
1002   2  =     call display$hex$byte(pd.lst - sd.ncondev);
       =     else                                            /* It's Concurrent */
1003   2  =     call display$hex$byte(pd.lst);
       =
1004   2  =     call crlf; call print$infield(42,'r',28,0('Current Virtual Console =    '));
1006   2  =     call display$hex$byte(pd.cns);
       =
1007   2  =     if (not vers) then do;
1009   3  =     call crlf;call print$infield(42,'r',20,0('Background Mode =    '));
1011   3  =     mode = (vccb.state);
1012   3  =     if (mode >= 0) and (mode <= 0200h ) then do;
1014   4  =        if (mode mod 2) = 0 then
1015   4  =        call print$buffer(.('DYNAMIC $'));
       =        else
```

```
1016   4  =          call print$buffer(.('BUFFERED$'));
1017   4  =          end;
1018   3  =       else call print$buffer(.('OTHER $'));
1019   3  =       end;
          =
1020   2  =       if (not vers) then do;
1022   3  =       call crlf; call print$infield(42,'r',17,@('Buffer Space =    '));
1024   3  =       call display$hex$word(vccb.maxbufsiz);
1025   3  =       call co('K');
1026   3  =       end;
          =
1027   2  =       call crlf;
1028   2  =       call print$infield(42,'r',31,@('Maximum Memory Per Process =    '));
1029   2  =       call display$hex$word(sd.mmp);
1030   2  =       call print$buffer(.('  PARA$'));
          =
1031   2  =       call crlf; call print$infield(42,'r',31,@('Number of Virtual Consoles =    '));
1033   2  =       call display$hex$byte(sd.ncns);
          =
1034   2  =       call crlf; call print$infield(42,'r',23,@('Number of Printers =    '));
1036   2  =       call display$hex$byte(sd.nlst);
          =
1037   2  =       call crlf; call print$infield(42,'r',20,@('Temporary Drive =    '));
1039   2  =       call co(driv$letter(sd.tempdisk));
1040   2  =       call co(':');
          =
1041   2  =       call crlf; call print$infield(42,'r',17,@('System Drive =    '));
1043   2  =       call co(driv$letter(sd.srchdisk));
1044   2  =       call co(':');
          =
1045   2  =       call crlf; call print$infield(42,'r',21,@('Ticks Per Second =    '));
1047   2  =       call display$hex$byte(sd.tickspersec);
          =
1048   2  =       call crlf; call print$infield(42,'r',20,@('Day File Option =    '));
1050   2  =       if sd.dayfile = 0 then
1051   2  =          call print$buffer(.('NO $'));
          =       else
1052   2  =          call print$buffer(.('YES$'));
          =
          =
1053   2  =       call crlf; call print$infield(42,'r',23,@('BDOS Compatability =    '));
1055   2  =       if sd.cmod <> 0 then
1056   2  =          call print$buffer(.('YES$'));
          =       else
1057   2  =          call print$buffer(.('NO $'));
          =
1058   2  =       call crlf; call print$infield(42,'r',20,@('Number of Flags =    '));
1060   2  =       call display$hex$byte(sd.nflags);
          =
1061   2  =       call crlf; call print$infield(42,'r',27,@('Free Queue Descriptors =    '));
1063   2  =          pd$ptr.offset = sd.qul;
1064   2  =          count = 0;
1065   2  =          do while pd$ptr.offset <> 0;
1066   3  =             count = count + 1;
1067   3  =             pd$ptr.offset = pd.link;
1068   3  =          end;
1069   2  =       call display$hex$byte(count);
```

```
                =
1070    2       =       call crlf; call print$infield(42,'r',22,0('Free Queue Buffer =    '));
1072    2       =            md$ptr.offset = .sd.qmau(0);
1073    2       =            sat$ptr.segment = md.start;          /* start of qbuffer SAT       */
1074    2       =            sat$ptr.offset = size(sat);          /* skip 1st 5 bytes bookeeping*/
1075    2       =        qsize = 0;
1076    2       =        do while sat.start <> 0;          /* byte offset for queue buffer*/
1077    3       =              if sat.numfallocs = 0 then          /* dooes anyone own this piece?*/
1078    3       =                 qsize = qsize + sat.len;
1079    3       =              sat$ptr.offset = size(sat) + sat$ptr.offset; /* Get next entry   */
1080    3       =            end;
1081    2       =       call display$hex$word(qsize);
1082    2       =       call print$buffer(.(' BYTES$'));
                =
1083    2       =       call crlf; call print$infield(42,'r',29,0('Free Process Descriptors =    '));
1085    2       =          pd$ptr.offset = sd.pul;
1086    2       =          count = 0;
1087    2       =          do while pd$ptr.offset <> 0;
1088    3       =             count = count + 1;
1089    3       =             pd$ptr.offset = pd.link;
1090    3       =          end;
1091    2       =       call display$hex$byte(count);
                =
1092    2       =       call crlf; call print$infield(42,'r',28,0('Free Memory Descriptors =    '));
1094    2       =          pd$ptr.offset = sd.mdul;
1095    2       =          count = 0;
1096    2       =          do while pd$ptr.offset <> 0;
1097    3       =             count = count + 1;
1098    3       =             pd$ptr.offset = pd.link;
1099    3       =          end;
1100    2       =       call display$hex$byte(count);
                =
1101    2       =       call crlf;
1102    2       =       if not(specified) then
1103    2       =       call cons$wait;
                =       else
1104    2       =       call terminate;
                =
1105    2       =   end displaygen;
                =
                =
                =
```



```
            /*****************************************************************************

                                        MAIN PROGRAM

            *****************************************************************************/


1106    1       plmstart: procedure public;
1107    2         dcl (i,version) byte,
                      ver address,
                      validchar byte;
1108    2         dcl bdosversion lit '30h';          /* BDOS 3.0 or later */
```

```
1109   2        dcl osproduct lit '14h';          /* CCP/M-86          */
1110   2        dcl mpmproduct lit '11h';         /* MP/M-86           */

1111   2        dcl vers$str$pointer pointer;
1112   2        dcl vers$str$ptr structure (
                    offset word,
                    segment word) at (@vers$str$pointer);
1113   2        dcl (doscan,chr) byte;

1114   2        ver = get$version;
1115   2        if low(ver) < bdosversion or
                  ( (high(ver) <> osproduct) and (high(ver) <> mpmproduct) ) then
1116   2        do;
1117   3           call print$buffer (.('Requires Concurrent CP/M-86 or MP/M-86$'));
1118   3           call reboot;                                  /* use CP/M exit */
1119   3        end;
                 else
1120   2        do;
1121   3          version = high(ver) mod 2;              /* 0 = CCP/M-86, 1 = MP/M-86 */
1122   3          sysdat$pointer = get$sysdat;
1123   3          flag$ptr.segment,md$ptr.segment,ms$ptr.segment,
                   sat$ptr.segment,qd$ptr.segment,pd$ptr.segment,vccb$ptr.segment
                   = sysdat$ptr.segment;

1124   3          doscan = true;
1125   3          repeat = false;
1126   3          specified = false;                                  /* Default  */
1127   3          intrval = 01h;                                      /* Default  */
                                                                /* Scan for option  */
1128   3            do while doscan ;                             /* Loop until Q(uit)*/
1129   4              if buff(0) <> 0 then do;                    /* Command line arg */
1131   5                 i = 1;                                   /* was used. Get it.*/
1132   5                 do while buff(i) = ' ' ;          /* Skip intervening  blanks */
1133   6                       i = i + 1;
1134   6                       end;
1135   5                 if buff(i) = lbracket then
1136   5                       i = i + 1;
                          else
1137   5                       call print$opt$err;
1138   5                 chr = buff(i);                                /* 1st arg */
1139   5                 i = i + 1;
1140   5                 if (buff(i) = ',') or (buff(i) = ' ') or (buff(i) = ']') then
1141   5                       i = i + 1;                        /* Skip blank or comma */
                          else
1142   5                       call print$opt$err;
1143   5                 if (buff(i-1) <> rbracket) then do;       /* Keep going,more args*/
1145   6                    if (buff(i) = 'c') or (buff(i) = 'C') then do;
1147   7                       repeat = true;
1148   7                       i = i + 1;
1149   7                       end;
                            else
1150   6                       call print$opt$err;
1151   6                  if (buff(i) <> rbracket) then do;             /* Still more ?*/
1153   7                              if (buff(i) = ' ') or (buff(i) = ',') then do;
1155   8                                  i = i + 1;
1156   8                                  end;
                                    else
```

```
1157  7                              call print$optserr;
                                                /* Get ascii hex interval data */
1158  7                         intrval = ascnex(buff(i));
1159  7                         i = i + 1;
1160  7                         if (buff(i) <> rbracket) then do;
1162  8                            intrval = shl(intrval,4);
1163  8                            intrval = intrval + ascnex(buff(i));
1164  8                         end;                  /* Now convert to system ticks */
1165  7                         intrval = intrval * sd.tickspersec;
1166  7                   end;
1167  6              end;

1168  5              buff(0) = 0;                                  /* Go back to menu*/
1169  5              specified = true;                             /* next time.     */
1170  5              end;
1171  4         else do;                                           /* No args's given*/
1172  5              call disp$mainhdr;                            /* Show the menu */
1173  5              chr = conin;
1174  5              end;

1175  4              validchar = false;
1176  4              do while not(validchar);                      /* Select action  */
1177  5              validchar = true;
1178  5              if (chr = 'h') or (chr = 'H') then do;
1180  6              call display$help;
1181  6              end;
1182  5              else if (chr = 'm') or (chr = 'M') then do;
1184  6              call display$mem;
1185  6              end;
1186  5              else if (chr = 'o') or (chr = 'O') then do;
1188  6              call display$gen(version);
1189  6              end;
1190  5              else if (chr = 'e') or (chr = 'E') then do;
1192  6              call terminate;                                  /* The Exit */
1193  6              end;
1194  5              else if (chr = 'p') or (chr = 'P') then do;
1196  6              call display$proc(0);
1197  6              end;
1198  5              else if (chr = 'q') or (chr = 'Q') then do;
1200  6              call display$queue;
1201  6              end;
1202  5              else if (chr = 'u') or (chr = 'U') then do;
1204  6              call display$proc(1);
1205  6              end;
1206  5              else do;                          /* Incorrect character was used */
1207  6               validchar = false;

1208  6               if not(specified) then do;       /* Invalid char was from menu   */
1210  7              if (chr = CR) then
1211  7                    call print$buffer(.('                              ->$'));
1212  7                 call print$buffer(.('     Invalid Option.$'));
1213  7                 call co(CR);                  /* Move left to beginning of line */
1214  7                 call print$buffer(.('                                 ->$'));
1215  7                 chr =  conin;        /* Get another char, hopefully a good one */
1216  7                 end;

                       else                    /* Invalid char was from the command line */
```

```
1217  6                 call print$opt$err;

1218  6               end;                          /* incorrect char case          */
1219  5             end;                      /* inner while loop - validchar */

1220  4           end;                    /* outer while loop - doscan    */
1221  3         end;

1222  2       end plmstart;

1223  1       end status;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
127	0021H	1	ABORT. . . . . BYTE MEMBER(PD)
281	03CFH	68	ASCHEX . . . . . PROCEDURE BYTE STACK=0004H 307 311 1158 1163
145	0000H	2	ATTACH . . . . . WORD MEMBER(VCCB)
1108			BDOSVERSION. . . . LITERALLY 1115
2			BOOLEAN. . . . . LITERALLY 859
145	000CH	1	BTMP . . . . . BYTE MEMBER(VCCB)
3	0000H	128	BUFF . . . . . BYTE ARRAY(128) EXTERNAL(0) 1129 1132 1135 1138 1140 1143 1145 1151 1153 1158 1160 1163 1168
130	0014H	2	BUFFER . . . . . WORD MEMBER(QD) 916
173	0004H	2	BUFFERADR. . . . . WORD PARAMETER AUTOMATIC 174 175
104	0054H	2	CCB. . . . . WORD MEMBER(SD) 991
123	0008H	2	CCB. . . . . WORD MEMBER(XINIT)
118			CCBSTRUCTURE . . . LITERALLY 145
119			CCBTAIL1 . . . . . LITERALLY 145
120			CCBTAIL2 . . . . . LITERALLY 145
121			CFCOMPC. . . . . LITERALLY
121			CFCONOUT . . . . . LITERALLY
121			CFLISTCP . . . . . LITERALLY
121			CFSWITCHS. . . . . LITERALLY
121			CFVOUT . . . . . LITERALLY
472	04A4H	1	CHAR . . . . . BYTE 478 479 574
165	0004H	1	CHAR . . . . . BYTE PARAMETER AUTOMATIC 166 167
196	0000H	1	CHAR . . . . . BYTE BASED(DATADDR) 205
155	0004H	1	CHAR . . . . . BYTE PARAMETER AUTOMATIC 156 157
883	14F1H	1	CHR. . . . . BYTE 886 889 966
1113	14F7H	1	CHR. . . . . BYTE 1138 1173 1178 1182 1186 1190 1194 1198 1202 1210 1215
303	00E6H	1	CHR. . . . . BYTE 304 305 307 311 315 316
277	00E5H	1	CHR. . . . . BYTE 279
683	0E2AH	1	CHR. . . . . BYTE 687 688 832
303	00E7H	1	CHR1 . . . . . BYTE 308 309 311 312
258	0355H	36	CLEAR. . . . . PROCEDURE STACK=000EH 321 351 387 594 697 850 989
89	00D3H	6	CLEARSEQ . . . . . BYTE ARRAY(6) INITIAL 260 261
104	0090H	1	CMOD . . . . . BYTE MEMBER(SD) 1055
589	00DEH	1	CNS. . . . . BYTE MEMBER(SORTRECD) 808 826
590	00DFH	1	CNS. . . . . BYTE MEMBER(SORTARRAY) 613 615 629 631 729 755 808 817 826
591	00E0H	1	CNS. . . . . BYTE MEMBER(SHAREARRAY) 648 655 776
468	0000H	1	CNS. . . . . BYTE MEMBER(POLTST) 497 557
127	0020H	1	CNS. . . . . BYTE MEMBER(PD) 497 543 729 776 991 1006
602	0006H	1	CNT. . . . . BYTE PARAMETER AUTOMATIC 603 606 608 621 625 628 629 631 632 633 634 673 675
860	14E0H	1	CNT. . . . . BYTE 863 870 873 878
155	013CH	19	CO . . . . . PROCEDURE STACK=000AH 190 191 202 205 210 217 218 232 237 249 254 273 454 869 879 994 995 1025 1039 1040 1043 1044 1213
145	0006H	1	COLUMN . . . . . BYTE MEMBER(VCCB)
152	012DH	15	CONIN. . . . . PROCEDURE BYTE STACK=0008H 279 304 308 312 478 574 687 832 888 966 1173 1215
127	0022H	2	CONMODE. . . . . WORD MEMBER(PD)

```
 162  015EH    15  CONSTAT. . . . . PROCEDURE BYTE STACK=0008H          571  829  963
 276  03B0H    18  CONSWAIT . . . . . PROCEDURE STACK=0006H          362  564  842  973 1103
 683  0E27H     1  CONT . . . . . . . BYTE       722  725  726  742  773  774  775  784
 883  14F0H     1  CONT . . . . . . . BYTE
 145  0022H     2  COSLEEP. . . . . WORD MEMBER(VCOB)
 986  00B6H     2  COUNT. . . . . . WORD       1064 1066 1069 1086 1088 1091 1095 1097 1100
 419  00E8H     1  COUNT. . . . . . BYTE
  91             CR . . . . . . . . LITERALLY       190  278  305  309  345  477  482  568  597  599  627
                                     638  663  676  686  691  887  892  950 1210 1213
 189  01DCH    17  CRLF . . . . . . PROCEDURE STACK=000EH          322  323  325  326  328  330  332  334
                                     336  338  340  341  346  352  353  354  356  357  359  361  363  365
                                     366  368  369  371  373  374  376  378  380  389  392  476  556  609
                                     644  853  856  886  949  953  992  996  999 1004 1009 1022 1027 1031
                                    1034 1037 1041 1045 1048 1053 1058 1061 1070 1083 1092 1101
 122             CSMABORT . . . . . LITERALLY
 122             CSMBACKGROUND. . . LITERALLY
 122             CSMBUFFERED. . . . LITERALLY
 122             CSMCTRLC . . . . . LITERALLY
 122             CSMCTRLP . . . . . LITERALLY
 122             CSMCTRLS . . . . . LITERALLY
 122             CSMFILEFULL. . . . LITERALLY
 122             CSMNOSWITCH. . . . LITERALLY
 122             CSMPURGING . . . . LITERALLY
 122             CSMSUSPEND . . . . LITERALLY
 193  0004H     4  DATADDR. . . . . . POINTER PARAMETER AUTOMATIC          194  196  197  205
 197  0004H     4  DATPTR . . . . . . STRUCTURE AT AUTOMATIC          206
 104  004FH     1  DAYFILE. . . . . . BYTE MEMBER(SD)      1050
 148  000EH     2  DBLK . . . . . . . WORD MEMBER(UDA)
   2             DCL. . . . . . . . LITERALLY
 148  000CH     2  DCNT . . . . . . . WORD MEMBER(UDA)
 169  0180H    16  DELAY. . . . . . . PROCEDURE STACK=000AH          578  838  970
 148  0012H     8  DFPASSWORD . . . . BYTE ARRAY(8) MEMBER(UDA)
 591  000AH     1  DISK . . . . . . . BYTE MEMBER(SHAREARRAY)          649  666  777
 858  16D6H   107  DISPLAYFLAGSTATUS. PROCEDURE STACK=0016H          948
 984  1A91H  1006  DISPLAYGEN . . . . PROCEDURE STACK=001EH         1168
 349  0552H   280  DISPLAYHELP. . . . PROCEDURE STACK=001CH         1180
 215  0264H    44  DISPLAYHEXBYTE . . PROCEDURE STACK=0010H          222  223  234  461  872  998 1002 1003
                                    1006 1033 1036 1047 1060 1069 1091 1100
 220  0290H    23  DISPLAYHEXWORD . . PROCEDURE STACK=0016H          251 1024 1029 1081
 681  10CEH  1499  DISPLAYMEM . . . . PROCEDURE STACK=002CH         1184
 470  094DH   872  DISPLAYPROC. . . . PROCEDURE STACK=002EH         1196 1204
 882  1741H   848  DISPLAYQUEUE . . . PROCEDURE STACK=0024H         1200
 319  046FH   209  DISPMAINHDR. . . . PROCEDURE STACK=001CH         1172
 592  0CB5H    53  DISPMEMHDR . . . . PROCEDURE STACK=0012H          698  837
 416  07BEH   399  DISPRESOURCE . . . PROCEDURE STACK=0028H          563
 396  06D5H   233  DISPSTATUS . . . . PROCEDURE STACK=001EH          562
 104  006AH     2  DLR. . . . . . . . WORD MEMBER(SD)       437
 148  0002H     2  DMAOFST. . . . . . WORD MEMBER(UDA)
 148  0004H     2  DMASEG . . . . . . WORD MEMBER(UDA)
 123  0002H     1  DOOR . . . . . . . BYTE MEMBER(XINIT)
1113  14F6H     1  DOSCAN . . . . . . BYTE       1124 1128
 148  0000H     2  DPARAM . . . . . . WORD MEMBER(UDA)          511  518  519
 847  0010H     2  DQ . . . . . . . . WORD MEMBER(QLIST)        913  939
 130  0012H     2  DQ . . . . . . . . WORD MEMBER(QD)       913
 983  0010H    16  DRIVLETTER . . . . BYTE ARRAY(16) DATA        994 1059 1063
 104  006CH     2  DRL. . . . . . . . WORD MEMBER(SD)       439
 127  0012H     1  DSK. . . . . . . . BYTE MEMBER(PD)       994
```

```
 127  0018H     2  OVRACT . . . . . .  WORD MEMBER(PD)
 104  0044H     2  ENDSEG . . . . . .  WORD MEMBER(SD)
 148  0010H     1  ERRORMODE. . . . .  BYTE MEMBER(UDA)
   2             FALSE. . . . . .  LITERALLY       439  533  537  573  700  725  775  831  861  899  965
                                      1125 1126 1175 1207
 810  154CH        FIND . . . . . .  LABEL
 127  0006H     2  FLAG . . . . . .  WORD MEMBER(PD)       501
 142  0000H     3  FLAG . . . . . .  STRUCTURE BASED(FLAGPOINTER)        528
 145  0004H     1  FLAG . . . . . .  BYTE MEMBER(VCCB)
 468  0009H     2  FLAG . . . . . .  WORD MEMBER(POLIST)        501  560
 858  0004H     1  FLAG . . . . . .  BYTE PARAMETER AUTOMATIC        860  865
  88  00D2H     1  FLAGLEN. . . . . .  BYTE INITIAL          531
 140  0020H     4  FLAGPOINTER. . . .  POINTER          141  142  528
 141  0020H     4  FLAGPTR. . . . . .  STRUCTURE AT         524  531 1123
 136  0006H     2  FLAGS. . . . . .  WORD MEMBER(MS)
 104  0056H     2  FLAGS. . . . . .  WORD MEMBER(SD)       524
 847  001AH     1  FLAGS. . . . . .  BYTE MEMBER(QLIST)        928  948
 130  0004H     2  FLAGS. . . . . .  WORD MEMBER(QD)       914  928
 116             FLAGSTRUCTURE. . .  LITERALLY       142
   2             FOREVER. . . . . .  LITERALLY
  81  00C5H     8  FREENAME . . . . .  BYTE ARRAY(8) INITIAL          750
   8  0000H     1  FUNC . . . . . .  BYTE PARAMETER          9
  12  0000H     1  FUNC . . . . . .  BYTE PARAMETER         13
 148  0006H     1  FUNC . . . . . .  BYTE MEMBER(UDA)
  16  0000H     1  FUNC . . . . . .  BYTE PARAMETER         17
   4  0000H     1  FUNC . . . . . .  BYTE PARAMETER          5
 186  01CDH    15  GETCURRPD. . . . .  PROCEDURE POINTER STACK=0008H        930
 302  0413H    92  GETINTRVAL . . . .  PROCEDURE WORD STACK=000CH         483  692  893
 718  1183H        GETMOS . . . . . .  LABEL
 707  1154H        GETMEMOWNERS . . .  LABEL
 493  09B5H        GETPDS . . . . . .  LABEL
 903  17A6H        GETQUEUES. . . . .  LABEL
 759  1354H        GETSHARED. . . . .  LABEL
 183  01BEH    15  GETSYSDAT. . . . .  PROCEDURE POINTER STACK=0008H      1122
 177  01A0H    15  GETVERSION . . . .  PROCEDURE WORD STACK=0008H       1114
 214  0000H    16  HEXDIGIT . . . . .  BYTE ARRAY(16) DATA        217  218
                   HIGH . . . . . .  BUILTIN         222 1115 1121
 264  0379H    36  HOME . . . . . .  PROCEDURE STACK=000EH          320  350  386  552  593  604  696  849
                                       901  988
  90  00D9H     6  HOMESEQ. . . . . .  BYTE ARRAY(6) INITIAL          266  267
 860  14ECH     1  I. . . . . .  BYTE        862  864  865  872  876  878
 682  0E22H     1  I. . . . . .  BYTE        719  720  770  771  803  804  812  813  821  822
 473  04AFH     1  I. . . . . .  BYTE        498  499  504  505  507  513  517  521  541
  21  0000H     2  I. . . . . .  WORD         22   23   24   25   26   27   28   29   30   31   32   33
                                   34   35   36   37   38   39   40   41   42   43   44   45   46   47
                                   48   49   50   51   52   53   54   55   56   57   58   59   60   61
                                   62   63   64   65   66   67   68   69   70   71   72   73   74   75
                                   76
 259  00E2H     1  I. . . . . .  BYTE        260  261
 243  00E1H     1  I. . . . . .  BYTE        248  253
 226  00E0H     1  I. . . . . .  BYTE        231  236
 195  00DFH     1  I. . . . . .  BYTE        201  204  209
1107  14F3H     1  I. . . . . .  BYTE       1131 1132 1133 1135 1136 1138 1139 1140 1141 1143 1145 1148
                                  1151 1153 1155 1158 1159 1160 1163
 265  00E3H     1  I. . . . . .  BYTE        266  267
 271  00E4H     1  I. . . . . .  BYTE        272
 883  14F2H     1  I. . . . . .  BYTE        906  907  918  919  921  922  925  926  933  935  936  937
```

```
                                                 938  939  940  947  948  959
142  0002H     1  IGNORE . . . . . .   BYTE MEMBER(FLAGS)
  8  0000H     2  INFO . . . . . .     WORD PARAMETER          10
 16  0000H     2  INFO . . . . . .     WORD PARAMETER          18
  4  0000H     2  INFO . . . . . .     WORD PARAMETER           6
 12  0000H     2  INFO . . . . . .     WORD PARAMETER          14
 86  0002H     2  INTRVAL. . . . . .   WORD              483  578  692  838  893  970 1127 1158 1162 1163 1165
225  0006H     1  JUSTIFY. . . . . .   BYTE PARAMETER AUTOMATIC          226  230  235
193  0008H     1  JUSTIFY. . . . . .   BYTE PARAMETER AUTOMATIC          195  200  208
241  0006H     1  JUSTIFY. . . . . .   BYTE PARAMETER AUTOMATIC          242  247  252
603  0E21H     1  K. . . . . .         BYTE              610  612  613  615  616  617  618  621  645  647  648  649
                                        650  651  652  655  675
473  04ACH     1  K. . . . . .         BYTE              526  530  532  536  554  555  557  559  560  561  562  563
                                        567
 83  00CEH     1  LBRACKET . . . . .   BYTE INITIAL           1135
123  000AH     2  LCB. . . . . .       WORD MEMBER(XINIT)
104  0086H     4  LCB. . . . . .       WORD ARRAY(2) MEMBER(SD)
124             LCBSTRUCTURE . . .     LITERALLY
104  0084H     1  LCKMAX . . . . . .   BYTE MEMBER(SD)
127  0014H     1  LDSK . . . . . .     BYTE MEMBER(PD)          777
589  000CH     2  LEN. . . . . .       WORD MEMBER(SORTRECD)                 806  824
193  0008H     1  LEN. . . . . .       BYTE PARAMETER AUTOMATIC          195  198  201  204  209
590  000CH     2  LEN. . . . . .       WORD MEMBER(SORTARRAY)            618  634  728  754  806  815  824
139  0002H     2  LEN. . . . . .       WORD MEMBER(SAT)              1078
591  000CH     2  LEN. . . . . .       WORD MEMBER(SHAREARRAY)           652  669  781
136  0004H     2  LENGTH . . . . . .   WORD MEMBER(MS)
133  0004H     2  LENGTH . . . . . .   WORD MEMBER(MD)          728  754  781
 92             LF . . . . . .        LITERALLY         191  273  278  305  309  345  477  482  568  597  599
                                        627  638  663  676  686  691  887  892  960
127  0000H     2  LINK . . . . . .     WORD MEMBER(PD)         1067 1089 1098
133  0000H     2  LINK . . . . . .     WORD MEMBER(MD)          757
136  0000H     2  LINK . . . . . .     WORD MEMBER(MS)          733  736  737  739  785  788  789  791
130  0000H     2  LINK . . . . . .     WORD MEMBER(QD)          930
469  0030H   128  LINKLIST . . . . .   WORD ARRAY(64)           702  711  720  764  771
  2             LIT. . . . . .        LITERALLY           2   91   92   99  100  101  106  107  108  109  110
                                        111  112  113  114  115  116  117  118  119  120  121  122  123  124
                                       1108 1109 1110
470  0004H     1  LNKFIELD . . . . .   BYTE PARAMETER AUTOMATIC          471  495
                 LOW. . . . . .       BUILTIN           223  426  456  461 1115
127  0024H     1  LST. . . . . .       BYTE MEMBER(PD)         1002 1003
104  0052H     2  LUL. . . . . .       WORD MEMBER(SD)
127  0015H     1  LUSER. . . . . .     BYTE MEMBER(PD)          778
603  0E1FH     1  M. . . . . .         BYTE              608  610  643  645
104  0076H     2  MAL. . . . . .       WORD MEMBER(SD)
136  0008H     2  MAU. . . . . .       WORD MEMBER(MS)          726  735  737  738  741  779  787  789  790  793
145  0010H     2  MAXBUFSTZ. . . . .   WORD MEMBER(VCCB)            1024
 79  00C3H     1  MAXMD. . . . . .     BYTE INITIAL             723  748  774
 78  00C2H     1  MAXPD. . . . . .     BYTE INITIAL             493  701  708  763
 80  00C4H     1  MAXQFLAGS. . . . .   BYTE INITIAL             864
 82  00CDH     1  MAXQUEUES. . . . .   BYTE INITIAL             905
133  0000H    10  MD . . . . . .       STRUCTURE BASED(MDPOINTER)                727  728  752  754  757  780  781
                                       1073
682  0E23H     1  MDCNT. . . . . .     BYTE              718  723  724  727  728  729  731  744  748  750  752  753
                                        754  755  756  801  828  836
131  0014H     4  MDPOINTER. . . . .   POINTER           132  133  727  728  752  754  757  780  781 1073
132  0014H     4  MDPTR. . . . . .     STRUCTURE AT             726  747  748  757  779 1072 1123
 93             MDSTRUCTURE. . . .     LITERALLY         133
```

```
  104  0058H     2  MODUL . . . . . .   WORD MEMBER(SD)      1094
  127  0016H     2  MEM. . . . . .      WORD MEMBER(PD)       495  709  721  772
  104  005AH     2  MFL. . . . . .      WORD MEMBER(SD)       747
  145  0008H     1  MIMIC. . . . . .    BYTE MEMBER(VCCB)
  104  004CH     2  MNR. . . . . .      WORD MEMBER(SD)      1029
  987  003AH     2  MODE . . . . . .    WORD       1011 1012 1014
  104  0046H     1  MODULEMAP. . . . .  BYTE MEMBER(SD)
    4  0000H        MON1 . . . . . .    PROCEDURE EXTERNAL(1) STACK=0000H          150  157  167  171  175  181
    8  0000H        MON2 . . . . . .    PROCEDURE BYTE EXTERNAL(2) STACK=0000H     153  160  163
   12  0000H        MON3 . . . . . .    PROCEDURE WORD EXTERNAL(3) STACK=0000H     173
   16  0000H        MON4 . . . . . .    PROCEDURE POINTER EXTERNAL(4) STACK=0000H       184  187
 1110               MPMPRODUCT . . . .  LITERALLY      1115
  104  0040H     2  MPMSEG . . . . . .  WORD MEMBER(SD)
  104  007CH     2  MPMVERNUM. . . . .  WORD MEMBER(SD)
  136  0000H    10  MS . . . . . .      STRUCTURE BASED(MSPOINTER)          726  733  735  736  737  738  739
                                        741  779  785  787  788  789  790  791  793
  847  000CH     2  MSGCNT . . . . . .  WORD MEMBER(QLIST)        911  938
  130  0016H     2  MSGCNT . . . . . .  WORD MEMBER(QD)       911
  847  000AH     2  MSGLEN . . . . . .  WORD MEMBER(QLIST)        910  937
  130  000EH     2  MSGLEN . . . . . .  WORD MEMBER(QD)       910
  130  0018H     2  MSGOUT . . . . . .  WORD MEMBER(QD)
  145  0009H     1  MSOURCE. . . . . .  BYTE MEMBER(VCCB)
  134  0018H     4  MSPOINTER. . . . .  POINTER          135  136  726  733  735  736  737  738  739  741  779
                                        785  787  788  789  790  791  793
  135  0018H     4  MSPTR. . . . . .    STRUCTURE AT           721  735  739  772  788  791 1123
   94               MSSTRUCTURE. . . .  LITERALLY      136
  148  0011H     1  MULTCNT. . . . . .  BYTE MEMBER(UDA)
  683  0080H     2  N. . . . . .        INTEGER          704  711  759  764
  473  04ADH     1  N. . . . . .        BYTE       491  493  494  497  499  501  502  503  506  511  518  519
                                        523  536  543  546  549  554  565  567  579
  883  0084H     2  N. . . . . .        INTEGER
  590  0000H     8  NAME . . . . . .    BYTE ARRAY(8) MEMBER(SORTARRAY)          612  628  731  750  804  813
                                        822
  591  0000H     8  NAME . . . . . .    BYTE ARRAY(8) MEMBER(SHAREARRAY)         647  664  783
  589  0000H     8  NAME . . . . . .    BYTE ARRAY(8) MEMBER(SORTRECD)      804  822
  847  0000H     8  NAME . . . . . .    BYTE ARRAY(8) MEMBER(QLIST)         907  935
  468  0001H     8  NAME . . . . . .    BYTE ARRAY(8) MEMBER(PDLIST)        499  559
  130  0006H     8  NAME . . . . . .    BYTE ARRAY(8) MEMBER(QD)       447  451  907
  127  0008H     8  NAME . . . . . .    BYTE ARRAY(8) MEMBER(PD)       499  731  783  919  941  945
  123  0006H     1  NCCB . . . . . .    BYTE MEMBER(XINIT)
  104  0049H     1  NCCB . . . . . .    BYTE MEMBER(SD)
  145  0007H     1  NCHAR. . . . . .    BYTE MEMBER(VCCB)
  104  0085H     1  NCIODEV. . . . . .  BYTE MEMBER(SD)
  104  0047H     1  NCNS . . . . . .    BYTE MEMBER(SD)      1033
  104  0083H     1  NCONDEV. . . . . .  BYTE MEMBER(SD)      1002
  127  001DH     1  NET. . . . . .      BYTE MEMBER(PD)
  130  0002H     1  NET. . . . . .      BYTE MEMBER(QD)
  104  004AH     1  NFLAGS . . . . . .  BYTE MEMBER(SD)       532 1060
  123  0007H     1  NLST . . . . . .    BYTE MEMBER(XINIT)
  104  0048H     1  NLST . . . . . .    BYTE MEMBER(SD)      1036
  104  0084H     1  NLSTDEV. . . . . .  BYTE MEMBER(SD)
  847  0008H     2  NMSGS. . . . . .    WORD MEMBER(QLIST)        909  936
  130  0010H     2  NMSGS. . . . . .    WORD MEMBER(QD)       909
    2               NO . . . . . .      LITERALLY
  473  04AEH     1  NOTFOUND . . . . .  BYTE       525  527  533  537
  419  00E9H     1  NOTFOUND . . . . .  BYTE
  847  000EH     2  NQ . . . . . .      WORD MEMBER(QLIST)        912  943
```

```
 130  0014H     2  NQ . . . . . .   WORD MEMBER(QD)       912
 104  004EH     1  NSLAVES. . . . .  BYTE MEMBER(SD)
 281  0004H     1  NUM. . . . . .   BYTE PARAMETER AUTOMATIC       282  283  285  288  290  293  295  297
                                     300
 169  0004H     2  NUA. . . . . .   WORD PARAMETER AUTOMATIC       170  171
 139  0004H     1  NUMALLOCS. . . .  BYTE MEMBER(SAT)        1077
 270  0004H     1  NUMLINES . . . .  BYTE PARAMETER AUTOMATIC       271  272
 123  0005H     1  NVCNS. . . . . .  BYTE MEMBER(XINIT)
 418  002EH     2  OFFS . . . . . .  WORD
 126  0000H     2  OFFSET . . . . .  WORD MEMBER(PDPTR)        492  493  528  547  707  708  711  714  720
                                      724  762  763  764  766  771  916  917  939  940  943  944 1063 1065
                                     1067 1085 1087 1089 1094 1095 1098
1112  0000H     2  OFFSET . . . . .  WORD MEMBER(VERSSTRPTR)
 123  0000H     2  OFFSET . . . . .  WORD MEMBER(XINITPTR)
 197  0000H     2  OFFSET . . . . .  WORD MEMBER(DATPTR)        206
 129  0000H     2  OFFSET . . . . .  WORD MEMBER(QDPTR)        446  450  904  905  930
 147  0000H     2  OFFSET . . . . .  WORD MEMBER(UDAPTR)       510  516
 144  0000H     2  OFFSET . . . . .  WORD MEMBER(VCCBPTR)      991
 141  0000H     2  OFFSET . . . . .  WORD MEMBER(FLAGPTR)      524  531
 138  0000H     2  OFFSET . . . . .  WORD MEMBER(SATPTR)       1074 1079
 135  0000H     2  OFFSET . . . . .  WORD MEMBER(RSPTR)        721  736  739  772  788  791
 132  0000H     2  OFFSET . . . . .  WORD MEMBER(RDPTR)        726  747  748  757  779 1072
 103  0000H     2  OFFSET . . . . .  WORD MEMBER(SYSDATPTR)
 104  008BH     1  OPMAX. . . . . .  BYTE MEMBER(SD)
 127  001CH     1  ORG. . . . . .   BYTE MEMBER(PD)
 130  0003H     1  ORG. . . . . .   BYTE MEMBER(QD)
1109           OSPRODUCT. . . .  LITERALLY       1115
 847  0012H     8  OWNER. . . . . .  BYTE ARRAY(8) MEMBER(QLIST)       919  922  926  947
 127  001EH     2  PARENT . . . . .  WORD MEMBER(PD)
  20  0002H   285  PATCH. . . . . .  PROCEDURE STACK=0002H
 145  000AH     1  PC . . . . . .   BYTE MEMBER(VCCB)
 101           PCH11. . . . . .  LITERALLY
 101           PCHCTLC. . . . .  LITERALLY
 101           PCHCTLO. . . . .  LITERALLY
 101           PCHCTLS. . . . .  LITERALLY
 101           PCHROUT. . . . .  LITERALLY
 101           PCHRSX . . . . .  LITERALLY
 127  0000H    48  PD . . . . . .   STRUCTURE BASED(PDPOINTER)       495  497  499  501  502  503  504
                                      506  509  515  543  547  709  714  721  729  731  766  772  776  777
                                      778  783  919  941  945  991  994  998 1002 1003 1006 1067 1089 1098
 590  000AH     2  PD . . . . . .   WORD MEMBER(SORTARRAY)       616  632  724  753  807  816  825
 589  000AH     2  PD . . . . . .   WORD MEMBER(SORTRECD)        807  825
 142  0000H     2  PD . . . . . .   WORD MEMBER(FLAG)       528
 682  0E24H     1  PDCNT. . . . . .  BYTE       701  702  705  708  712  716  719  760  763  765  768  770
 148  001AH     1  PDCNT. . . . . .  BYTE MEMBER(UDA)
  97           PDHDR. . . . . .  LITERALLY       127
 468  00EAH   960  PDLIST . . . . . .  STRUCTURE ARRAY(64)       497  499  501  502  503  505  511  518  519
                                      523  536  543  557  559  560  561  562  563
 125  000CH     4  PDPOINTER. . . . .  POINTER          126  127  495  497  499  501  502  503  504  506  509
                                      515  543  547  709  714  721  729  731  766  772  776  777  778  783
                                      919  941  945  990  991  994  998 1002 1003 1006 1067 1089 1098
 126  000CH     4  PDPTR. . . . . .   STRUCTURE AT         492  493  528  547  707  708  711  714  720  724
                                      762  763  764  766  771  916  917  939  940  943  944 1063 1065 1067
                                     1085 1087 1089 1094 1096 1098 1123
  98           PDSTRUCTURE. . . .  LITERALLY       127
 100           PFACTIVE . . . . .  LITERALLY
 100           PFCHILDABORT . . .  LITERALLY
```

```
100                 PFCTLC . . . . . LITERALLY
100                 PFCTLD . . . . . LITERALLY
100                 PFKEEP . . . . . LITERALLY
100                 PFKERNAL . . . . LITERALLY
100                 PFNOCTLS . . . . LITERALLY
100                 PFPURE . . . . . LITERALLY
100                 PFRAW. . . . . LITERALLY
100                 PFRESOURCE . . . LITERALLY
100                 PFSYS. . . . . LITERALLY
100                 PFTABLE. . . . . LITERALLY
100                 PFTEMPKEEP . . . LITERALLY
133   0006H     2   PLIST. . . . . WORD MEMBER(HD)
1106  1E7FH   659   PLMSTART . . . . PROCEDURE PUBLIC STACK=0032H
104   006EH     2   PLR. . . . . WORD MEMBER(SD)       441
 96                 PNAMSIZ. . . . . LITERALLY        447   451   498   559   612   628   647   664   782   906   918
                                      921   925   935   941   945   947
127   002CH     2   PRET . . . . . WORD MEMBER(PD)
859   14EBH     1   PREV . . . . . BYTE         861   867   874
173   0190H    16   PRINTBUFFER. . . PROCEDURE STACK=000AH          278   345   477   482   558   568   595   596
                                      597   598   599   600   614   620   627   630   638   639   654   663   676   677
                                      686   691   851   852   854   855   887   892   934   942   946   951   952   954
                                      955   960  1015  1016  1018  1030  1051  1052  1056  1057  1082  1117  1211  1212
                                     1214
225   02A7H    87   PRINTHEXBYTE . . . PROCEDURE STACK=001AH          426   456   557   561   615   631   648   649
                                      650   665   666   667
241   02FEH    87   PRINTHEXWORD . . . PROCEDURE STACK=0020H          430   560   616   617   618   632   633   634
                                      651   652   668   669   936   937   938
193   01EDH   119   PRINTINFIELD . . . PROCEDURE STACK=0018H          324   327   329   331   333   335   337   339
                                      342   355   358   360   362   364   367   370   372   375   377   379   388   390
                                      391   393   394   401   402   403   404   405   406   407   408   409   410   411
                                      414   423   425   429   432   433   436   438   440   442   443   447   451   457
                                      460   463   466   559   612   628   647   664   935   941   945   947   993   997
                                     1000  1005  1010  1023  1028  1032  1035  1038  1042  1046  1049  1054  1059  1062
                                     1071  1084  1093
344   0540H    18   PRINTOPTERR. . . . PROCEDURE STACK=0012H         1137  1142  1150  1157  1217
554   08A9H         PRINTPDS . . . . . LABEL
933   18F7H         PRINTQDS . . . . . LABEL
848   1649H    45   PRINTQHDR. . . . . PROCEDURE STACK=0012H          897   973
602   0CEAH   996   PRINTSORTED. . . . PROCEDURE STACK=0028H          828
127   0005H     1   PRIOR. . . . . BYTE MEMBER(PD)       502
468   000BH     1   PRIOR. . . . . BYTE MEMBER(PDLIST)        502   561
385   066AH   107   PRNTHEADER . . . . PROCEDURE STACK=001CH          487   580
 99                 PSCIOWAIT. . . . . LITERALLY
 99                 PSDELAY. . . . . LITERALLY
 99                 PSDQ . . . . . LITERALLY
 99                 PSFLAGWAIT . . . . LITERALLY
 99                 PSNQ . . . . . LITERALLY
 99                 PSPOLL . . . . . LITERALLY
 99                 PSRUN. . . . . LITERALLY
 99                 PSSLEEP. . . . . LITERALLY
 99                 PSSWAP . . . . . LITERALLY
 99                 PSTERM . . . . . LITERALLY
104   005CH     2   PUL. . . . . WORD MEMBER(SD)      1085
603   0E20H     1   Q. . . . . BYTE         611   619   646   653
145   0020H     2   QBUFF. . . . . WORD MEMBER(VCCB)
130   0000H    28   QD . . . . . STRUCTURE BASED(QDPOINTER)          447   451   907   909   910   911   912
                                      913   914   916   928   930
```

```
883  14EFH     1  QDCNT. . . . . .  BYTE        900  905  907  909  910  911  912  913  919  922  926  928
                                     929  933  956  957  959  971
128  0010H     4  QDPOINTER. . . . .  POINTER        129  130  447  451  907  909  910  911  912  915  914
                                     916  926  930
129  0010H     4  QDPTR. . . . . .  STRUCTURE AT        446  450  904  905  930 1123
107             QDSTRUCTURE. . . .  LITERALLY       130
114             QFDEV. . . . . .  LITERALLY
110             QFHIDE . . . . . .  LITERALLY
109             QFKEEP . . . . . .  LITERALLY
108             QFMX . . . . . .  LITERALLY
113             QFRPL. . . . . .  LITERALLY
111             QFRSP. . . . . .  LITERALLY
112             QFTABLE. . . . . .  LITERALLY
847  0E2BH  1728  QLIST. . . . . .  STRUCTURE ARRAY(64)        907  909  910  911  912  913  919  922  925
                                     928  935  936  937  938  939  943  947  948
104  0074H     2  QLR. . . . . .  WORD MEMBER(SD)        904
104  0060H     8  QMAU . . . . . .  WORD ARRAY(4) MEMBER(SD)        1072
106             QNAMSIZ. . . . . .  LITERALLY       130
145  001EH     2  QPBBUFFPTR . . . .  WORD MEMBER(VCCB)
145  0018H     1  QPBFLAGS . . . . .  BYTE MEMBER(VCCB)
145  001CH     2  QPBNMSGS . . . . .  WORD MEMBER(VCCB)
145  001AH     2  QPBQADDR . . . . .  WORD MEMBER(VCCB)
145  0019H     1  QPBRESRVD. . . . .  BYTE MEMBER(VCCB)
115             QPBSTRUCTURE . . .  LITERALLY
986  0086H     2  QSIZE. . . . . .  WORD       1075 1078 1081
145  0002H     2  QUEUE. . . . . .  WORD MEMBER(VCCB)
104  005EH     2  QUL. . . . . .  WORD MEMBER(SD)       1063
145  0028H     2  R1 . . . . . .  WORD MEMBER(VCCB)
145  002AH     2  R2 . . . . . .  WORD MEMBER(VCCB)
165  0160H    19  RAWCO. . . . . .  PROCEDURE STACK=000AH            261  267
159  014FH    15  RAWCONIN . . . . .  PROCEDURE BYTE STACK=0008H
 84  00CFH     1  RBRACKET . . . . .  BYTE INITIAL         1143 1151 1160
149  011FH    14  REBOOT . . . . . .  PROCEDURE STACK=0008H          1118
 85  00D0H     1  REPEAT . . . . . .  BYTE        481  488  573  576  690  699  831  834  891  898  965  968
                                     1125 1147
127  0028H     4  RESERVD. . . . . .  BYTE ARRAY(4) MEMBER(PD)
468  000DH     2  RESOURCE . . . . .  WORD MEMBER(PDLIST)        506  511  518  519  523  536  543  563
123  0003H     2  RESRVD1. . . . . .  BYTE ARRAY(2) MEMBER(XINIT)
104  0068H     2  RLR. . . . . .  WORD MEMBER(SD)       435
104  0042H     2  RSPSEG . . . . . .  WORD MEMBER(SD)
416  0004H     2  RSRCE. . . . . .  WORD PARAMETER AUTOMATIC          418  426  430  435  437  439  441  446
                                     450  455  456  461
145  000DH     1  RSVD . . . . . .  BYTE MEMBER(VCCB)
139  0000H     5  SAT. . . . . .  STRUCTURE BASED(SATPOINTER)         1074 1076 1077 1078 1079
137  001CH     4  SATPOINTER . . . .  POINTER        138  139 1076 1077 1078
138  001CH     4  SATPTR . . . . . .  STRUCTURE AT        1073 1074 1079 1123
 95             SATSTRUCTURE . . .  LITERALLY       139
683  00B2H     2  SAVMAU . . . . . .  WORD        735  737  738  741  787  789  790  793
683  0E28H     1  SCDCNT . . . . . .  BYTE        761  774  776  777  778  780  781  783  796  828
602  0004H     1  SCNT . . . . . .  BYTE PARAMETER AUTOMATIC        603  636  641  643  655  659  661  664
                                     665  666  667  668  669  671  673  675
127  002AH     2  SCRATCH. . . . . .  WORD MEMBER(PD)
104  0000H   145  SD . . . . . .  STRUCTURE BASED(SYSDATPOINTER)        316  435  437  439  441  492  524
                                     532  707  747  762  904  991 1002 1029 1033 1036 1039 1043 1047 1050
                                    1055 1060 1063 1072 1085 1094 1165
105  0000H     1  SDBYTE . . . . . .  BYTE BASED(SYSDATPOINTER) ARRAY(1)
148  0008H     2  SEARCHA. . . . . .  WORD MEMBER(UDA)
```

```
 148  000AH     2  SEARCHBASE. . . .  WORD MEMBER(SD)
 148  0007H     1  SEARCHL. . . . . .  BYTE MEMBER(SD)
 126  0002H     2  SEGMENT. . . . . .  WORD MEMBER(PDPTR)          1123
 123  0002H     2  SEGMENT. . . . . .  WORD MEMBER(XIOSPTR)
 197  0002H     2  SEGMENT. . . . . .  WORD MEMBER(DATPTR)
 147  0002H     2  SEGMENT. . . . . .  WORD MEMBER(DDAPTR)          509   515
 144  0002H     2  SEGMENT. . . . . .  WORD MEMBER(VCCBPTR)        1123
 141  0002H     2  SEGMENT. . . . . .  WORD MEMBER(FLAGPTR)        1123
 138  0002H     2  SEGMENT. . . . . .  WORD MEMBER(SATPTR)         1073  1123
 135  0002H     2  SEGMENT. . . . . .  WORD MEMBER(LSPTR)          1123
 132  0002H     2  SEGMENT. . . . . .  WORD MEMBER(MDPTR)          1123
 103  0002H     2  SEGMENT. . . . . .  WORD MEMBER(SYSDATPTR)            1123
1112  0002H     2  SEGMENT. . . . . .  WORD MEMBER(VERSSTAPTR)
 129  0002H     2  SEGMENT. . . . . .  WORD MEMBER(CDPTR)          1123
 127  0025H     1  SF3. . . . . . . .  BYTE MEMBER(PD)
 127  0026H     1  SF4. . . . . . . .  BYTE MEMBER(PD)
 127  0027H     1  SF5. . . . . . . .  BYTE MEMBER(PD)
 591  096FH  1200  SHAREARRAY . . . .  STRUCTURE ARRAY(80)          647   648   649   650   651   652   664   665   666
                                       667   668   669   776   777   778   780   781   783
                   SHL. . . . . . . .  BUILTIN              311  1162
                   SHR. . . . . . . .  BUILTIN              217
                   SIZE . . . . . . .  BUILTIN              991  1074  1079
 270  0390H    32  SKIPLINES. . . . .  PROCEDURE STACK=0010H              553   605   902
 104  0070H     2  SLR. . . . . . . .  WORD MEMBER(SD)            762
 590  048FH  1200  SORTARRAY. . . . .  STRUCTURE ARRAY(80)          612   613   615   616   617   618   628   629   631
                                       632   633   634   724   727   728   729   731   750   752   753   754   755   802
                                       804   806   807   808   810   811   813   815   816   817   820   822   824   825
                                       826
 801  1408H        SORTMOS. . . . . .  LABEL
 589  0480H    15  SORTRECD . . . . .  STRUCTURE            802   804   806   807   808   810   820   822   824   825   826
 104  0008H    56  SPACE. . . . . . .  WORD ARRAY(28) MEMBER(SD)
  87  00D1H     1  SPECIFIED. . . . .  BYTE              381   474   583   684   841   884   977  1102  1126  1169  1208
 104  004BH     1  SRCHDISK . . . . .  BYTE MEMBER(SD)          1043
 591  0008H     2  START. . . . . . .  WORD MEMBER(SHAREARRAY)              651   668   780
 590  0008H     2  START. . . . . . .  WORD MEMBER(SORTARRAY)              617   633   727   752   802   810   811   820
 589  0008H     2  START. . . . . . .  WORD MEMBER(SORTRECD)               802   810   820
 139  0000H     2  START. . . . . . .  WORD MEMBER(SAT)              1076
 136  0002H     2  START. . . . . . .  WORD MEMBER(MS)
 133  0002H     2  START. . . . . . .  WORD MEMBER(MD)          727   752   780  1073
 145  0005H     1  STARTCOL . . . . .  BYTE MEMBER(VCCB)
 127  0004H     1  STAT . . . . . . .  BYTE MEMBER(PD)          503   504
 468  000CH     1  STAT . . . . . . .  BYTE MEMBER(POLTST)      503   562   563
 396  0004H     1  STAT . . . . . . .  BYTE PARAMETER AUTOMATIC          397   398   400
 145  000EH     2  STATE. . . . . . .  WORD MEMBER(VCCB)         1011
 416  0006H     1  STATS. . . . . . .  BYTE PARAMETER AUTOMATIC          417   420   422
   1  0002H        STATUS . . . . . .  PROCEDURE STACK=0000H
 104  0000H     8  SUPMOD . . . . . .  WORD ARRAY(4) MEMBER(SD)
 102  0004H     4  SYSDATPOINTER. . .  POINTER              103   104   105   316   435   437   439   441   492   524   532
                                       707   747   762   904   991  1002  1029  1033  1036  1039  1043  1047  1050  1055
                                      1060  1063  1072  1085  1094  1122  1165
 103  0004H     4  SYSDATPTR. . . . .  STRUCTURE AT            1123
 104  008CH     4  SYSLTOT. . . . . .  WORD ARRAY(2) MEMBER(SD)
   2               TAB. . . . . . . .  LITERALLY
 683  0E29H     1  TEMP . . . . . . .  BYTE              695   699   700
 472  04ABH     1  TEMP . . . . . . .  BYTE              486   488   490
 883  14EEH     1  TEMP . . . . . . .  BYTE              896   898   899
 104  0050H     1  TEMPDISK . . . . .  BYTE MEMBER(SD)          1039
```

```
 180  01AFH   15  TERMINATE. . . . .  PROCEDURE STACK=0099H          347  395  585  843  979 1104 1192
 104  0072H    2  THRDRT . . . . . .  WORD MEMBER(SD)       492  707
 127  0002H    2  THREAD . . . . . .  WORD MEMBER(PD)       547  714  756
 303  002CH    2  TICDELAY . . . . .  WORD        516  517
 123  0000H    1  TICK . . . . . .  BYTE MEMBER(XINIT)
 104  0051H    1  TICKSPERSEC. . . .  BYTE MEMBER(SD)       316 1047 1165
 123  0001H    1  TICKSSEC . . . . .  BYTE MEMBER(XINIT)
 104  007EH    4  TOD. . . . . .  WORD ARRAY(2) MEMBER(SD)
 104  0082H    1  TOD_SEC. . . . . .  BYTE MEMBER(SD)
   2           TRUE . . . . . .  LITERALLY        481  486  525  690  695  722  742  773  794  874  891
                                  896 1124 1147 1169 1177
 127  0010H    2  UDA. . . . . .  WORD MEMBER(PD)       509  515
 148  0000H   27  UDA. . . . . .  STRUCTURE BASED(UDAPOINTER)          511  518  519
 146  0026H    4  UDAPOINTER . . . .  POINTER          147  148  511  518  519
 147  0028H    4  UDAPTR . . . . . .  STRUCTURE AT         509  510  515  516
 117           UDASTRUCTURE . . .  LITERALLY        148
 133  0008H    2  UNUSED . . . . . .  WORD MEMBER(MD)
 127  0013H    1  USER . . . . . .  BYTE MEMBER(PD)       998
 591  000BH    1  USER . . . . . .  BYTE MEMBER(SHAREARRAY)          650  667  778
 145  0024H    2  USLEEP . . . . . .  WORD MEMBER(VCCB)
 241  0004H    2  VAL. . . . . .  WORD PARAMETER AUTOMATIC          242  251
 225  0004H    1  VAL. . . . . .  BYTE PARAMETER AUTOMATIC          226  234
1107  14F5H    1  VALIDCHAR. . . . .  BYTE       1175 1176 1177 1207
 220  0004H    2  VALUE. . . . . .  WORD PARAMETER AUTOMATIC          221  222  223
 215  0004H    1  VALUE. . . . . .  BYTE PARAMETER AUTOMATIC          216  217  218
 145  0008H    1  VC . . . . . .  BYTE MEMBER(VCCB)
 145  0000H   44  VCCB . . . . . .  STRUCTURE BASED(VCCBPOINTER)          991 1011 1024
 143  0024H    4  VCCBPOINTER. . . .  POINTER          144  145 1011 1024
 144  0024H    4  VCCBPTR. . . . . .  STRUCTURE AT          991 1123
 145  0016H    2  VCMXQ. . . . . .  WORD MEMBER(VCCB)
1107  00BCH    2  VER. . . . . .  WORD       1114 1115 1121
 104  007AH    2  VERNUM . . . . . .  WORD MEMBER(SD)
 984  0004H    1  VERS . . . . . .  BYTE PARAMETER AUTOMATIC          985 1001 1007 1020
 104  0078H    2  VERSION. . . . . .  WORD MEMBER(SD)
1107  14F4H    1  VERSION. . . . . .  BYTE       1121 1188
1111  00BEH    4  VERSSTRPOINTER . .  POINTER        1112
1112  00BEH    4  VERSSTRPTR . . . .  STRUCTURE AT
 145  0012H    2  VINQ . . . . . .  WORD MEMBER(VCCB)
 145  0014H    2  VOUTQ. . . . . .  WORD MEMBER(VCCB)
 145  0026H    2  VSLEEP . . . . . .  WORD MEMBER(VCCB)
 127  001AH    2  WAIT . . . . . .  WORD MEMBER(PD)       506
 241  0008H    1  WIDTH. . . . . .  BYTE PARAMETER AUTOMATIC          242  244  248  253
 225  0008H    1  WIDTH. . . . . .  BYTE PARAMETER AUTOMATIC          226  227  231  236
 193  000CH    1  WIDTH. . . . . .  BYTE PARAMETER AUTOMATIC          195  198  201  209
 683  0E25H    1  X. . . . . .  BYTE        730  731  749  750  782  783  801  802  804  806  807  808
                                  809
 123  0000H   12  XINIT. . . . . .  STRUCTURE BASED(XINITPOINTER)
 123           XINITOFFSET. . . .  LITERALLY
 123  0008H    4  XINITPOINTER . . .  POINTER          123
 123  0008H    4  XINITPTR . . . . .  STRUCTURE AT
 683  0E26H    1  Y. . . . . .  BYTE        809  810  811  813  815  816  817  818  820  822  824  825
                                  826
```


MODULE INFORMATION:

```
CODE AREA SIZE      = 2112H   8466D
CONSTANT AREA SIZE  = 0985H   2437D
VARIABLE AREA SIZE  = 14F8H   5368D
MAXIMUM STACK SIZE  = 0032H     50D
1772 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
ISIS-II MCS-86 LOCATER, V1.2 INVOKED BY:
:F0: SYSTAT.LNK ODC(SMCODE,DATS,DATA,STACK,CONST)) TO(SYSCODE(0),DATS(10000H)) SS(STACK(+32)) TO SYSTAT.

SYMBOL TABLE OF MODULE SCD
READ FROM FILE SYSTAT.LNK
WRITTEN TO FILE SYSTAT.

BASE    OFFSET TYPE SYMBOL              BASE    OFFSET TYPE SYMBOL

0000H   1F7FH  PUB  PLMSTART           0000H   005DH  PUB  XDOS
0000H   005DH  PUB  MON1               0000H   005DH  PUB  MON2
0000H   005DH  PUB  MON3               0000H   005DH  PUB  MON4
1000H   0004H  PUB  BDISK              1000H   0006H  PUB  MAXA
1000H   0050H  PUB  CMDRV              1000H   0051H  PUB  PASS0
1000H   0053H  PUB  LEN0               1000H   0054H  PUB  PASS1
1000H   0056H  PUB  LEN1               1000H   005CH  PUB  FCB
1000H   006CH  PUB  FCB16              1000H   007CH  PUB  CR
1000H   007DH  PUB  RR                 1000H   007FH  PUB  RO
1000H   0080H  PUB  BUFF               1000H   0080H  PUB  TBUFF
1000H   0080H  PUB  BUFFA              1000H   005CH  PUB  FCBA
1000H   0100H  PUB  SAVEAX             1000H   0102H  PUB  SAVEBX
1000H   0104H  PUB  SAVECX             1000H   0106H  PUB  SAVEDX

STATUS:  SYMBOLS AND LINES
1000H   2010H  SYM  MEMORY             0000H   0102H  SYM  PATCH
1000H   0108H  SYM  I                  1000H   01CAH  SYM  MAXPD
1000H   01CBH  SYM  MAXMD              1000H   01CCH  SYM  MAXQFLAGS
1000H   01CDH  SYM  FREENAME           1000H   01D5H  SYM  MAXQUEUES
1000H   01D6H  SYM  LBRACKET           1000H   01D7H  SYM  RBRACKET
1000H   01D8H  SYM  REPEAT             1000H   01DAH  SYM  INTRVAL
1000H   01D9H  SYM  SPECIFIED          1000H   01DAH  SYM  FLAGLEN
1000H   01DBH  SYM  CLEARSEQ           1000H   01E1H  SYM  HOMESEQ
1000H   01DCH  SYM  SYSDATPOINTER      1000H   01DCH  SYM  SYSDATPTR
1000H   01DCH  BAS  SD                 1000H   01DCH  BAS  SDBYTE
1000H   01E0H  SYM  XINITPOINTER       1000H   01E0H  SYM  XINITPTR
1000H   01E0H  BAS  XINIT              1000H   01E4H  SYM  PDPOINTER
1000H   01E4H  SYM  PDPTR              1000H   01E4H  BAS  PD
1000H   01E8H  SYM  QDPOINTER          1000H   01E8H  SYM  QDPTR
1000H   01E8H  BAS  QD                 1000H   01ECH  SYM  MDPOINTER
1000H   01ECH  SYM  MDPTR              1000H   01ECH  BAS  MD
1000H   01F0H  SYM  MSPOINTER          1000H   01F0H  SYM  MSPTR
1000H   01F0H  BAS  MS                 1000H   01F4H  SYM  SATPOINTER
1000H   01F4H  SYM  SATPTR             1000H   01F4H  BAS  SAT
1000H   01F8H  SYM  FLAGPOINTER        1000H   01F8H  SYM  FLAGPTR
1000H   01F8H  BAS  FLAG               1000H   01FCH  SYM  VCCBPOINTER
1000H   01FCH  SYM  VCCBPTR            1000H   01FCH  BAS  VCCB
1000H   0200H  SYM  UDAPOINTER         1000H   0200H  SYM  UDAPTR
1000H   0200H  BAS  UDA                0000H   021FH  SYM  REBOOT
0000H   0220H  SYM  CONIN              0000H   023CH  SYM  CO
STACK   0004H  SYM  CHAR               0000H   024FH  SYM  RAWCONIN
0000H   025EH  SYM  CONSTAT           0000H   026DH  SYM  RAWCO
STACK   0004H  SYM  CHAR               0000H   0280H  SYM  DELAY
STACK   0004H  SYM  NUM                0000H   0290H  SYM  PRINTBUFFER
STACK   0004H  SYM  BUFFERADR          0000H   02A0H  SYM  GETVERSION
0000H   02AFH  SYM  TERMINATE          0000H   02BEH  SYM  GETSYSDAT
0000H   02CDH  SYM  GETCURRPD          0000H   02DCH  SYM  CRLF
0000H   02EDH  SYM  PRINTINFIELD       STACK   000CH  SYM  WIDTH
STACK   000AH  SYM  JUSTIFY            STACK   0008H  SYM  LEN
STACK   0004H  SYM  DATADDR            1000H   01E7H  SYM  I
STACK   0004H  BAS  CHAR               STACK   0004H  SYM  DATPTR
```

```
1000H  1652H  SYM  HEXDIGIT
STACK  0004H  SYM  VALUE
STACK  0004H  SYM  VALUE
STACK  0008H  SYM  WIDTH
STACK  0004H  SYM  VAL
0000H  03FEH  SYM  PRINTHEXWORD
STACK  0006H  SYM  JUSTIFY
1000H  01E9H  SYM  I
1000H  01EAH  SYM  I
1000H  01EBH  SYM  I
STACK  0004H  SYM  NUMLINES
0000H  048DH  SYM  CONSWAIT
0000H  04CFH  SYM  ASCHEX
0000H  0513H  SYM  GETINTRVAL
1000H  01EFH  SYM  CHR1
0000H  056FH  SYM  DISPMAINHDR
0000H  065CH  SYM  DISPLAYHELP
0000H  07D5H  SYM  DISPSTATUS
0000H  08BEH  SYM  DISPRESOURCE
STACK  0004H  SYM  RSRCE
1000H  01F0H  SYM  COUNT
1000H  01F2H  SYM  PDLIST
0000H  0A4DH  SYM  DISPLAYPROC
1000H  0582H  SYM  CHAR
1000H  05B4H  SYM  K
1000H  05B6H  SYM  NOTFOUND
0000H  0AB5H  SYM  GETPDS
1000H  05B8H  SYM  SORTRECD
1000H  0A77H  SYM  SHAREARRAY
0000H  0DEAH  SYM  PRINTSORTED
STACK  0004H  SYM  SCNT
1000H  0F28H  SYM  Q
0000H  11CEH  SYM  DISPLAYMEM
1000H  0F2BH  SYM  MDCNT
1000H  0F2DH  SYM  X
1000H  0F2FH  SYM  CONT
1000H  01B8H  SYM  N
1000H  0F32H  SYM  CHR
0000H  1254H  SYM  GETMEMOWNERS
0000H  1454H  SYM  GETSHARED
0000H  164CH  SYM  FIND
0000H  17A9H  SYM  PRINTQHDR

STACK  0004H  SYM  FLAG
1000H  15F4H  SYM  I
0000H  1841H  SYM  DISPLAYQUEUE
1000H  018CH  SYM  N
1000H  15F8H  SYM  CONT
1000H  15FAH  SYM  I
0000H  19F7H  SYM  PRINTQDS
0000H  1891H  SYM  DISPLAYGEN
1000H  01BEH  SYM  COUNT
1000H  01C2H  SYM  MODE
1000H  15FBH  SYM  I
1000H  01C4H  SYM  VER
1000H  01C6H  SYM  VERSSTRPOINTER
1000H  15FEH  SYM  DOSCAN
0000H  0102H  LIN      20
0000H  0110H  LIN      23
0000H  011AH  LIN      25

0000H  0554H  SYM  DISPLAYHEXBYTE
0000H  0390H  SYM  DISPLAYHEXWORD
0000H  0347H  SYM  PRINTHEXBYTE
STACK  0006H  SYM  JUSTIFY
1000H  01E6H  SYM  I
STACK  0008H  SYM  WIDTH
STACK  0004H  SYM  VAL
0000H  0455H  SYM  CLEAR
0000H  0479H  SYM  HOME
0000H  049DH  SYM  SKIPLINES
1000H  01ECH  SYM  I
1000H  01EDH  SYM  CHR
STACK  0004H  SYM  NUM
1000H  01EEH  SYM  CHR
1000H  0134H  SYM  TICDELAY
0000H  0640H  SYM  PRINTOPTERR
0000H  076AH  SYM  PRNTHEADER
STACK  0004H  SYM  STAT
STACK  0006H  SYM  STATS
1000H  0136H  SYM  OFFS
1000H  01F1H  SYM  NOTFOUND
1000H  0138H  SYM  LINKLIST
STACK  0004H  SYM  LNKFIELD
1000H  05B3H  SYM  TEMP
1000H  0585H  SYM  N
1000H  0587H  SYM  I
0000H  0CA9H  SYM  PRINTPDS
1000H  05C7H  SYM  SORTARRAY
0000H  0D35H  SYM  DISPMEMHDR
STACK  0006H  SYM  CNT
1000H  0F27H  SYM  N
1000H  0F29H  SYM  K
1000H  0F2AH  SYM  I
1000H  0F2CH  SYM  PDCNT
1000H  0F2EH  SYM  Y
1000H  0F30H  SYM  SCDCNT
1000H  0F31H  SYM  TEMP
1000H  01BAH  SYM  SAVHAU
0000H  1283H  SYM  GETMDS
0000H  15D8H  SYM  SORTMDS
1000H  0F33H  SYM  QLIST
0000H  17D6H  SYM  DISPLAYFLAGSTATU
-S
1000H  15F3H  SYM  PREV
1000H  15F5H  SYM  CNT
1000H  15F6H  SYM  TEMP
1000H  15F7H  SYM  QDCNT
1000H  15F9H  SYM  CHR
0000H  18A6H  SYM  GETQUEUES
1000H  1662H  SYM  DRIVLETTER
STACK  0004H  SYM  VERS
1000H  01C0H  SYM  QSIZE
0000H  1F7FH  SYM  PLMSTART
1000H  15FCH  SYM  VERSION
1000H  15FDH  SYM  VALIDCHAR
1000H  01C6H  SYM  VERSSTRPTR
1000H  15FFH  SYM  CHR
0000H  0105H  LIN      22
0000H  0115H  LIN      24
0000H  011FH  LIN      26
```

```
0000H  0124H  LIN    27      0000H  0129H  LIN    28
0000H  012EH  LIN    29      0000H  0133H  LIN    30
0000H  0138H  LIN    31      0000H  013DH  LIN    32
0000H  0142H  LIN    33      0000H  0147H  LIN    34
0000H  014CH  LIN    35      0000H  0151H  LIN    36
0000H  0156H  LIN    37      0000H  015BH  LIN    38
0000H  0160H  LIN    39      0000H  0165H  LIN    40
0000H  016AH  LIN    41      0000H  016FH  LIN    42
0000H  0174H  LIN    43      0000H  0179H  LIN    44
0000H  017EH  LIN    45      0000H  0183H  LIN    46
0000H  0188H  LIN    47      0000H  018DH  LIN    48
0000H  0192H  LIN    49      0000H  0197H  LIN    50
0000H  019CH  LIN    51      0000H  01A1H  LIN    52
0000H  01A6H  LIN    53      0000H  01ABH  LIN    54
0000H  01B0H  LIN    55      0000H  01B5H  LIN    56
0000H  01BAH  LIN    57      0000H  01BFH  LIN    58
0000H  01C4H  LIN    59      0000H  01C9H  LIN    60
0000H  01CEH  LIN    61      0000H  01D3H  LIN    62
0000H  01D8H  LIN    63      0000H  01DDH  LIN    64
0000H  01E2H  LIN    65      0000H  01E7H  LIN    66
0000H  01ECH  LIN    67      0000H  01F1H  LIN    68
0000H  01F6H  LIN    69      0000H  01FBH  LIN    70
0000H  0200H  LIN    71      0000H  0205H  LIN    72
0000H  020AH  LIN    73      0000H  020FH  LIN    74
0000H  0214H  LIN    75      0000H  0219H  LIN    76
0000H  021DH  LIN    77      0000H  021FH  LIN   149
0000H  0222H  LIN   150      0000H  0227H  LIN   151
0000H  0220H  LIN   152      0000H  0230H  LIN   153
0000H  023CH  LIN   154      0000H  023CH  LIN   155
0000H  023FH  LIN   157      0000H  0246H  LIN   158
0000H  024FH  LIN   159      0000H  0252H  LIN   160
0000H  025EH  LIN   161      0000H  025EH  LIN   162
0000H  0261H  LIN   163      0000H  026DH  LIN   164
0000H  0260H  LIN   165      0000H  0270H  LIN   167
0000H  027CH  LIN   168      0000H  0280H  LIN   169
0000H  0283H  LIN   171      0000H  028CH  LIN   172
0000H  0290H  LIN   173      0000H  0293H  LIN   175
0000H  029CH  LIN   176      0000H  02A0H  LIN   177
0000H  02A3H  LIN   178      0000H  02AFH  LIN   179
0000H  02AFH  LIN   180      0000H  02B2H  LIN   181
0000H  02BCH  LIN   182      0000H  02BEH  LIN   183
0000H  02C1H  LIN   184      0000H  02CDH  LIN   185
0000H  02CDH  LIN   186      0000H  02D0H  LIN   187
0000H  02DCH  LIN   188      0000H  02DCH  LIN   189
0000H  02DFH  LIN   190      0000H  02E5H  LIN   191
0000H  02EBH  LIN   192      0000H  02EDH  LIN   193
0000H  02F0H  LIN   198      0000H  02F8H  LIN   200
0000H  02FEH  LIN   201      0000H  030FH  LIN   202
0000H  0315H  LIN   203      0000H  031BH  LIN   204
0000H  0328H  LIN   205      0000H  0334H  LIN   206
0000H  0337H  LIN   207      0000H  033DH  LIN   208
0000H  0343H  LIN   209      0000H  0354H  LIN   210
0000H  035AH  LIN   211      0000H  0360H  LIN   213
0000H  0364H  LIN   215      0000H  0367H  LIN   217
0000H  0377H  LIN   218      0000H  038CH  LIN   219
0000H  0390H  LIN   220      0000H  0393H  LIN   222
0000H  039CH  LIN   223      0000H  03A3H  LIN   224
0000H  03A7H  LIN   225      0000H  03AAH  LIN   227
0000H  03B0H  LIN   230      0000H  03B6H  LIN   231
0000H  03C6H  LIN   232      0000H  03CCH  LIN   233
```

```
0000H  03D2H  LIN   234        0000H  03D6H  LIN   235
0000H  03DEH  LIN   236        0000H  03EEH  LIN   237
0000H  03F4H  LIN   238        0000H  03FAH  LIN   239
0000H  03FEH  LIN   241        0000H  0401H  LIN   244
0000H  0407H  LIN   247        0000H  040DH  LIN   248
0000H  0410H  LIN   249        0000H  0423H  LIN   250
0000H  0429H  LIN   251        0000H  042FH  LIN   252
0000H  0435H  LIN   253        0000H  0445H  LIN   254
0000H  0448H  LIN   255        0000H  0451H  LIN   257
0000H  0455H  LIN   258        0000H  0456H  LIN   260
0000H  0466H  LIN   261        0000H  0471H  LIN   262
0000H  0477H  LIN   263        0000H  0479H  LIN   264
0000H  047CH  LIN   266        0000H  048AH  LIN   267
0000H  0495H  LIN   268        0000H  049BH  LIN   269
0000H  049DH  LIN   270        0000H  04A0H  LIN   272
0000H  04ADH  LIN   273        0000H  04B3H  LIN   274
0000H  04B9H  LIN   275        0000H  04BDH  LIN   276
0000H  04C0H  LIN   278        0000H  04C7H  LIN   279
0000H  04CDH  LIN   280        0000H  04CFH  LIN   281
0000H  04D2H  LIN   283        0000H  04DEH  LIN   285
0000H  04E2H  LIN   286        0000H  04E4H  LIN   288
0000H  04F0H  LIN   290        0000H  04F4H  LIN   291
0000H  04F6H  LIN   293        0000H  0502H  LIN   295
0000H  0506H  LIN   296        0000H  0508H  LIN   297
0000H  050CH  LIN   300        0000H  0513H  LIN   301
0000H  0513H  LIN   302        0000H  0516H  LIN   304
0000H  051CH  LIN   305        0000H  0527H  LIN   307
0000H  052EH  LIN   308        0000H  0534H  LIN   309
0000H  053FH  LIN   311        0000H  0552H  LIN   312
0000H  0558H  LIN   314        0000H  055AH  LIN   315
0000H  055FH  LIN   316        0000H  056DH  LIN   317
0000H  056FH  LIN   318        0000H  056FH  LIN   319
0000H  0572H  LIN   320        0000H  0575H  LIN   321
0000H  0578H  LIN   322        0000H  057BH  LIN   323
0000H  057EH  LIN   324        0000H  0590H  LIN   325
0000H  0593H  LIN   326        0000H  0596H  LIN   327
0000H  05A8H  LIN   328        0000H  05ABH  LIN   329
0000H  05BDH  LIN   330        0000H  05C0H  LIN   331
0000H  05D2H  LIN   332        0000H  05D5H  LIN   333
0000H  05E7H  LIN   334        0000H  05EAH  LIN   335
0000H  05FCH  LIN   336        0000H  05FFH  LIN   337
0000H  0611H  LIN   338        0000H  0614H  LIN   339
0000H  0626H  LIN   340        0000H  0629H  LIN   341
0000H  062CH  LIN   342        0000H  063EH  LIN   343
0000H  0640H  LIN   344        0000H  0643H  LIN   345
0000H  064AH  LIN   346        0000H  064DH  LIN   347
0000H  0650H  LIN   348        0000H  0652H  LIN   349
0000H  0655H  LIN   350        0000H  0658H  LIN   351
0000H  065BH  LIN   352        0000H  065EH  LIN   353
0000H  0661H  LIN   354        0000H  0664H  LIN   355
0000H  0676H  LIN   356        0000H  0679H  LIN   357
0000H  067CH  LIN   358        0000H  068EH  LIN   359
0000H  0691H  LIN   360        0000H  06A3H  LIN   361
0000H  06A6H  LIN   362        0000H  06B8H  LIN   363
0000H  06BBH  LIN   364        0000H  06CDH  LIN   365
0000H  06D0H  LIN   366        0000H  06D3H  LIN   367
0000H  06E5H  LIN   368        0000H  06E8H  LIN   369
0000H  06EBH  LIN   370        0000H  06FDH  LIN   371
0000H  0700H  LIN   372        0000H  0712H  LIN   373
0000H  0715H  LIN   374        0000H  0718H  LIN   375
```

```
0000H 072AH LIN   376        0000H 0725H LIN   377
0000H 073FH LIN   378        0000H 0742H LIN   379
0000H 0754H LIN   380        0000H 0757H LIN   381
0000H 0760H LIN   382        0000H 0765H LIN   383
0000H 0768H LIN   384        0000H 076AH LIN   385
0000H 076DH LIN   386        0000H 0770H LIN   387
0000H 0773H LIN   388        0000H 0785H LIN   389
0000H 0788H LIN   390        0000H 079AH LIN   391
0000H 07ACH LIN   392        0000H 07AFH LIN   393
0000H 07C1H LIN   394        0000H 07D3H LIN   395
0000H 07D5H LIN   396        0000H 07D8H LIN   398
0000H 07E1H LIN   400        0000H 07EDH LIN   401
0000H 07FCH LIN   402        0000H 0800H LIN   403
0000H 081AH LIN   404        0000H 0829H LIN   405
0000H 0838H LIN   406        0000H 0847H LIN   407
0000H 0856H LIN   408        0000H 0865H LIN   409
0000H 0874H LIN   410        0000H 0883H LIN   411
0000H 0892H LIN   412        0000H 08A8H LIN   413
0000H 08A8H LIN   414        0000H 08BAH LIN   415
0000H 08BEH LIN   416        0000H 08C1H LIN   420
0000H 08CAH LIN   422        0000H 08D6H LIN   423
0000H 08E5H LIN   425        0000H 08F5H LIN   426
0000H 08FAH LIN   427        0000H 08FAH LIN   429
0000H 090AH LIN   430        0000H 0916H LIN   431
0000H 0919H LIN   432        0000H 0928H LIN   433
0000H 0937H LIN   435        0000H 0944H LIN   436
0000H 0953H LIN   437        0000H 0960H LIN   438
0000H 096FH LIN   439        0000H 097CH LIN   440
0000H 0988H LIN   441        0000H 0998H LIN   442
0000H 09A7H LIN   443        0000H 09B6H LIN   450
0000H 09CFH LIN   452        0000H 09CFH LIN   454
0000H 09D5H LIN   455        0000H 09DBH LIN   456
0000H 09EAH LIN   457        0000H 09F9H LIN   458
0000H 09F9H LIN   460        0000H 0A08H LIN   461
0000H 0A12H LIN   462        0000H 0A14H LIN   463
0000H 0A21H LIN   464        0000H 0A37H LIN   465
0000H 0A37H LIN   466        0000H 0A49H LIN   467
0000H 0A4DH LIN   470        0000H 0A50H LIN   474
0000H 0A59H LIN   476        0000H 0A5CH LIN   477
0000H 0A63H LIN   478        0000H 0A69H LIN   479
0000H 0A77H LIN   481        0000H 0A7CH LIN   482
0000H 0A85H LIN   483        0000H 0A89H LIN   486
0000H 0A8EH LIN   487        0000H 0A91H LIN   488
0000H 0A9FH LIN   489        0000H 0AA0H LIN   490
0000H 0AA5H LIN   491        0000H 0AAAH LIN   492
0000H 0AB5H LIN   493        0000H 0AD7H LIN   494
0000H 0AD8H LIN   495        0000H 0AF2H LIN   497
0000H 0B07H LIN   498        0000H 0B13H LIN   499
0000H 0B30H LIN   500        0000H 0B36H LIN   501
0000H 0B48H LIN   502        0000H 0B53H LIN   503
0000H 0B58H LIN   504        0000H 0B5FH LIN   505
0000H 0B69H LIN   506        0000H 0B7AH LIN   507
0000H 0B81H LIN   509        0000H 0B8CH LIN   510
0000H 0B92H LIN   511        0000H 0BA4H LIN   512
0000H 0BA4H LIN   513        0000H 0BB2H LIN   515
0000H 0B3DH LIN   516        0000H 0BC3H LIN   517
0000H 0BCAH LIN   518        0000H 0BD6H LIN   519
0000H 0BEFH LIN   520        0000H 0BF2H LIN   521
0000H 0BF9H LIN   523        0000H 0C08H LIN   524
0000H 0C13H LIN   525        0000H 0C18H LIN   526
```

```
0000H  0C1DH  LIN    527
0000H  0C31H  LIN    530
0000H  0C42H  LIN    532
0000H  0C54H  LIN    534
0000H  0C69H  LIN    537
0000H  0C63H  LIN    541
0000H  0C86H  LIN    545
0000H  0C8AH  LIN    547
0000H  0C95H  LIN    549
0000H  0CA0H  LIN    552
0000H  0CA9H  LIN    554
0000H  0CBEH  LIN    556
0000H  0CD7H  LIN    558
0000H  0CF9H  LIN    560
0000H  0D25H  LIN    562
0000H  0D49H  LIN    564
0000H  0D59H  LIN    567
0000H  0D6DH  LIN    569
0000H  0D7AH  LIN    573
0000H  0D85H  LIN    576
0000H  0D93H  LIN    579
0000H  0D9DH  LIN    581
0000H  0DA6H  LIN    584
0000H  0DAEH  LIN    587
0000H  0DB5H  LIN    592
0000H  0DB8H  LIN    594
0000H  0DC5H  LIN    596
0000H  0DD3H  LIN    598
0000H  0DE1H  LIN    600
0000H  0DEAH  LIN    602
0000H  0DF0H  LIN    605
0000H  0DFFH  LIN    608
0000H  0E21H  LIN    610
0000H  0E36H  LIN    612
0000H  0E61H  LIN    614
0000H  0E80H  LIN    616
0000H  0EACH  LIN    618
0000H  0ED2H  LIN    620
0000H  0EF0H  LIN    622
0000H  0F02H  LIN    625
0000H  0F1CH  LIN    628
0000H  0F48H  LIN    630
0000H  0F6CH  LIN    632
0000H  0F9CH  LIN    634
0000H  0FBAH  LIN    638
0000H  0FC8H  LIN    641
0000H  0FF0H  LIN    644
0000H  0FF9H  LIN    646
0000H  1023H  LIN    648
0000H  104FH  LIN    650
0000H  107BH  LIN    652
0000H  10A1H  LIN    654
0000H  10BFH  LIN    656
0000H  10D1H  LIN    659
0000H  10EDH  LIN    663
0000H  1111H  LIN    665
0000H  1141H  LIN    667
0000H  1171H  LIN    669
0000H  118CH  LIN    673
0000H  11B6H  LIN    676

0000H  0C2AH  LIN    528
0000H  0C39H  LIN    531
0000H  0C4FH  LIN    533
0000H  0C56H  LIN    536
0000H  0C6BH  LIN    540
0000H  0C72H  LIN    543
0000H  0C86H  LIN    546
0000H  0C95H  LIN    548
0000H  0C9FH  LIN    551
0000H  0CA3H  LIN    553
0000H  0CBAH  LIN    555
0000H  0CC1H  LIN    557
0000H  0CDEH  LIN    559
0000H  0D0FH  LIN    561
0000H  0D35H  LIN    563
0000H  0D52H  LIN    565
0000H  0D66H  LIN    568
0000H  0D73H  LIN    571
0000H  0D7FH  LIN    574
0000H  0D8CH  LIN    578
0000H  0D9AH  LIN    580
0000H  0D9DH  LIN    583
0000H  0DA8H  LIN    585
0000H  0DB1H  LIN    588
0000H  0DB8H  LIN    593
0000H  0DBEH  LIN    595
0000H  0DCCH  LIN    597
0000H  0DDAH  LIN    599
0000H  0DE8H  LIN    601
0000H  0DEDH  LIN    604
0000H  0DF6H  LIN    606
0000H  0E1EH  LIN    609
0000H  0E27H  LIN    611
0000H  0E51H  LIN    613
0000H  0E6AH  LIN    615
0000H  0E96H  LIN    617
0000H  0EC2H  LIN    619
0000H  0ED9H  LIN    621
0000H  0EF9H  LIN    623
0000H  0F15H  LIN    627
0000H  0F39H  LIN    629
0000H  0F54H  LIN    631
0000H  0F84H  LIN    633
0000H  0FB4H  LIN    636
0000H  0FC1H  LIN    639
0000H  0FD1H  LIN    643
0000H  0FF3H  LIN    645
0000H  1008H  LIN    647
0000H  1039H  LIN    649
0000H  1055H  LIN    651
0000H  1091H  LIN    653
0000H  10A8H  LIN    655
0000H  10C8H  LIN    657
0000H  10DAH  LIN    661
0000H  10F4H  LIN    664
0000H  1129H  LIN    666
0000H  1159H  LIN    668
0000H  1189H  LIN    671
0000H  11A0H  LIN    675
0000H  11BDH  LIN    677
```

```
0000H  11C4H  LIN   678
0000H  11CEH  LIN   681
0000H  11D4H  LIN   686
0000H  11E7H  LIN   688
0000H  11FAH  LIN   691
0000H  1207H  LIN   695
0000H  120FH  LIN   697
0000H  1215H  LIN   699
0000H  1228H  LIN   701
0000H  1242H  LIN   703
0000H  124EH  LIN   705
0000H  1254H  LIN   707
0000H  127EH  LIN   709
0000H  129CH  LIN   712
0000H  12AAH  LIN   715
0000H  12B1H  LIN   717
0000H  12B8H  LIN   719
0000H  12DAH  LIN   721
0000H  12EAH  LIN   723
0000H  1312H  LIN   725
0000H  1323H  LIN   727
0000H  1337H  LIN   729
0000H  1351H  LIN   731
0000H  1374H  LIN   733
0000H  1385H  LIN   736
0000H  13ADH  LIN   741
0000H  13BFH  LIN   744
0000H  13C6H  LIN   746
0000H  13DAH  LIN   748
0000H  1405H  LIN   750
0000H  1423H  LIN   752
0000H  143EH  LIN   754
0000H  144BH  LIN   756
0000H  1454H  LIN   759
0000H  145FH  LIN   761
0000H  146FH  LIN   763
0000H  14A1H  LIN   765
0000H  14AFH  LIN   767
0000H  14B9H  LIN   770
0000H  14DBH  LIN   772
0000H  14E8H  LIN   774
0000H  1507H  LIN   776
0000H  1524H  LIN   778
0000H  1538H  LIN   780
0000H  154CH  LIN   782
0000H  1575H  LIN   784
0000H  1585H  LIN   787
0000H  1592H  LIN   789
0000H  15C2H  LIN   794
0000H  15C8H  LIN   797
0000H  15D7H  LIN   800
0000H  15EBH  LIN   802
0000H  1607H  LIN   804
0000H  1626H  LIN   806
0000H  163FH  LIN   808
0000H  164CH  LIN   810
0000H  1689H  LIN   812
0000H  16B8H  LIN   814
0000H  16D9H  LIN   816
0000H  16E9H  LIN   818

0000H  11C8H  LIN   679
0000H  11D1H  LIN   684
0000H  11E1H  LIN   687
0000H  11F5H  LIN   690
0000H  1201H  LIN   693
0000H  120CH  LIN   696
0000H  1212H  LIN   698
0000H  1223H  LIN   700
0000H  1236H  LIN   702
0000H  1248H  LIN   704
0000H  1253H  LIN   706
0000H  125FH  LIN   708
0000H  1289H  LIN   711
0000H  12A0H  LIN   714
0000H  12AAH  LIN   716
0000H  12B3H  LIN   718
0000H  12C3H  LIN   720
0000H  12F5H  LIN   722
0000H  1301H  LIN   724
0000H  1317H  LIN   726
0000H  132FH  LIN   728
0000H  1345H  LIN   730
0000H  136EH  LIN   732
0000H  137EH  LIN   735
0000H  138BH  LIN   737
0000H  138AH  LIN   742
0000H  13C3H  LIN   745
0000H  13CFH  LIN   747
0000H  13F9H  LIN   749
0000H  1410H  LIN   751
0000H  1438H  LIN   753
0000H  1446H  LIN   755
0000H  144FH  LIN   757
0000H  145AH  LIN   760
0000H  1464H  LIN   762
0000H  148EH  LIN   764
0000H  14A5H  LIN   766
0000H  14AFH  LIN   768
0000H  14CCH  LIN   771
0000H  14E6H  LIN   773
0000H  1502H  LIN   775
0000H  151CH  LIN   777
0000H  152CH  LIN   779
0000H  1544H  LIN   781
0000H  1558H  LIN   783
0000H  157BH  LIN   785
0000H  158CH  LIN   788
0000H  15B4H  LIN   793
0000H  15C7H  LIN   796
0000H  15CEH  LIN   798
0000H  15D8H  LIN   801
0000H  15FBH  LIN   803
0000H  1620H  LIN   805
0000H  1637H  LIN   807
0000H  1646H  LIN   809
0000H  1676H  LIN   811
0000H  1696H  LIN   813
0000H  16BEH  LIN   815
0000H  16E1H  LIN   817
0000H  16EDH  LIN   819
```

```
0000H  16EDH  LIN   820
0000H  170AH  LIN   822
0000H  1728H  LIN   824
0000H  1741H  LIN   826
0000H  1752H  LIN   828
0000H  1764H  LIN   831
0000H  176FH  LIN   834
0000H  1787H  LIN   837
0000H  1791H  LIN   839
0000H  179CH  LIN   842
0000H  17A4H  LIN   845
0000H  17A9H  LIN   848
0000H  17AFH  LIN   850
0000H  17B9H  LIN   852
0000H  17C3H  LIN   854
0000H  17D1H  LIN   856
0000H  17D6H  LIN   858
0000H  17DEH  LIN   862
0000H  17E3H  LIN   864
0000H  17F6H  LIN   867
0000H  1803H  LIN   870
0000H  180EH  LIN   873
0000H  1818H  LIN   876
0000H  1824H  LIN   878
0000H  1837H  LIN   880
0000H  1841H  LIN   882
0000H  184DH  LIN   886
0000H  1857H  LIN   888
0000H  1868H  LIN   891
0000H  1877H  LIN   893
0000H  1882H  LIN   897
0000H  1893H  LIN   899
0000H  189DH  LIN   901
0000H  18A6H  LIN   903
0000H  18B2H  LIN   905
0000H  18DFH  LIN   907
0000H  1902H  LIN   909
0000H  191FH  LIN   911
0000H  192FH  LIN   913
0000H  1946H  LIN   916
0000H  1954H  LIN   918
0000H  197DH  LIN   920
0000H  1991H  LIN   922
0000H  19ADH  LIN   924
0000H  19B8H  LIN   926
0000H  19D7H  LIN   928
0000H  19F0H  LIN   930
0000H  19F6H  LIN   932
0000H  1A0AH  LIN   934
0000H  1A2CH  LIN   936
0000H  1A58H  LIN   938
0000H  1A7EH  LIN   940
0000H  1A9CH  LIN   942
0000H  1AB3H  LIN   944
0000H  1AD1H  LIN   946
0000H  1AF3H  LIN   948
0000H  1B06H  LIN   950
0000H  1B16H  LIN   952
0000H  1B20H  LIN   954
0000H  1B2EH  LIN   956

0000H  16FEH  LIN   821
0000H  1722H  LIN   823
0000H  1739H  LIN   825
0000H  1749H  LIN   827
0000H  175DH  LIN   829
0000H  1759H  LIN   832
0000H  1776H  LIN   836
0000H  178AH  LIN   838
0000H  1793H  LIN   841
0000H  17A1H  LIN   843
0000H  17A7H  LIN   846
0000H  17ACH  LIN   849
0000H  17B2H  LIN   851
0000H  17C0H  LIN   853
0000H  17CAH  LIN   855
0000H  17D4H  LIN   857
0000H  17D9H  LIN   861
0000H  17E3H  LIN   863
0000H  17F1H  LIN   865
0000H  17FDH  LIN   869
0000H  1807H  LIN   872
0000H  1813H  LIN   874
0000H  1822H  LIN   877
0000H  1831H  LIN   879
0000H  183DH  LIN   881
0000H  1844H  LIN   884
0000H  1850H  LIN   887
0000H  185DH  LIN   889
0000H  1870H  LIN   892
0000H  187DH  LIN   896
0000H  1885H  LIN   898
0000H  1898H  LIN   900
0000H  18A0H  LIN   902
0000H  18A7H  LIN   904
0000H  18D3H  LIN   906
0000H  18FCH  LIN   908
0000H  1917H  LIN   910
0000H  1927H  LIN   912
0000H  1937H  LIN   914
0000H  1940H  LIN   917
0000H  1960H  LIN   919
0000H  1985H  LIN   921
0000H  19A7H  LIN   923
0000H  19AFH  LIN   925
0000H  19D1H  LIN   927
0000H  19ECH  LIN   929
0000H  19F6H  LIN   931
0000H  19F7H  LIN   933
0000H  1A11H  LIN   935
0000H  1A42H  LIN   937
0000H  1A6EH  LIN   939
0000H  1A85H  LIN   941
0000H  1AA3H  LIN   943
0000H  1ABAH  LIN   945
0000H  1AD8H  LIN   947
0000H  1B03H  LIN   949
0000H  1B0FH  LIN   951
0000H  1B1DH  LIN   953
0000H  1B27H  LIN   955
0000H  1B36H  LIN   957
```


SER. #_____

```
0000H  1B3AH  LIN   959
0000H  1B4BH  LIN   961
0000H  1B58H  LIN   965
0000H  1B63H  LIN   968
0000H  1B71H  LIN   971
0000H  1B7AH  LIN   975
0000H  1B84H  LIN   978
0000H  1B8CH  LIN   981
0000H  1B91H  LIN   984
0000H  1B97H  LIN   989
0000H  1BA5H  LIN   991
0000H  1BC2H  LIN   993
0000H  1BE5H  LIN   995
0000H  1BEEH  LIN   997
0000H  1C08H  LIN   999
0000H  1C20H  LIN  1001
0000H  1C38H  LIN  1003
0000H  1C49H  LIN  1005
0000H  1C66H  LIN  1007
0000H  1C72H  LIN  1010
0000H  1C8FH  LIN  1012
0000H  1C9FH  LIN  1015
0000H  1CA9H  LIN  1017
0000H  1CB0H  LIN  1020
0000H  1CBCH  LIN  1023
0000H  1CD9H  LIN  1025
0000H  1CE2H  LIN  1028
0000H  1CFFH  LIN  1030
0000H  1D09H  LIN  1032
0000H  1D26H  LIN  1034
0000H  1D38H  LIN  1036
0000H  1D49H  LIN  1038
0000H  1D6CH  LIN  1040
0000H  1D75H  LIN  1042
0000H  1D98H  LIN  1044
0000H  1DA1H  LIN  1046
0000H  1DBEH  LIN  1048
0000H  1DD3H  LIN  1050
0000H  1DE3H  LIN  1052
0000H  1DEDH  LIN  1054
0000H  1E0BH  LIN  1055
0000H  1E17H  LIN  1058
0000H  1E2CH  LIN  1060
0000H  1E3AH  LIN  1062
0000H  1E57H  LIN  1064
0000H  1E64H  LIN  1066
0000H  1E72H  LIN  1068
0000H  1E78H  LIN  1070
0000H  1E90H  LIN  1072
0000H  1EA5H  LIN  1074
0000H  1EB1H  LIN  1076
0000H  1EC2H  LIN  1078
0000H  1ECFH  LIN  1080
0000H  1ED8H  LIN  1082
0000H  1EE2H  LIN  1084
0000H  1EFFH  LIN  1086
0000H  1F0CH  LIN  1088
0000H  1F1AH  LIN  1090
0000H  1F23H  LIN  1092
0000H  1F38H  LIN  1094

0000H  1C44H  LIN   960
0000H  1C51H  LIN   963
0000H  1C5DH  LIN   966
0000H  1C6AH  LIN   970
0000H  1C78H  LIN   973
0000H  1C72H  LIN   977
0000H  1C89H  LIN   979
0000H  1C8FH  LIN   982
0000H  1C94H  LIN   986
0000H  1C9AH  LIN   990
0000H  1CBFH  LIN   992
0000H  1CD4H  LIN   994
0000H  1CE5H  LIN   996
0000H  1D00H  LIN   998
0000H  1D0EH  LIN  1000
0000H  1D27H  LIN  1002
0000H  1D46H  LIN  1004
0000H  1D5BH  LIN  1006
0000H  1D6FH  LIN  1009
0000H  1D84H  LIN  1011
0000H  1D94H  LIN  1014
0000H  1DA4H  LIN  1016
0000H  1DA9H  LIN  1018
0000H  1DB9H  LIN  1022
0000H  1DCEH  LIN  1024
0000H  1DDFH  LIN  1027
0000H  1DF4H  LIN  1029
0000H  1E06H  LIN  1031
0000H  1E18H  LIN  1033
0000H  1E29H  LIN  1035
0000H  1E46H  LIN  1037
0000H  1E5BH  LIN  1039
0000H  1E72H  LIN  1041
0000H  1E87H  LIN  1043
0000H  1E9EH  LIN  1045
0000H  1EB3H  LIN  1047
0000H  1EC1H  LIN  1049
0000H  1EDEH  LIN  1051
0000H  1EEAH  LIN  1053
0000H  1EFFH  LIN  1055
0000H  1F10H  LIN  1057
0000H  1F1AH  LIN  1059
0000H  1F37H  LIN  1061
0000H  1F4CH  LIN  1063
0000H  1F5DH  LIN  1065
0000H  1F68H  LIN  1067
0000H  1F74H  LIN  1069
0000H  1F7EH  LIN  1071
0000H  1F9AH  LIN  1073
0000H  1FABH  LIN  1075
0000H  1FBBH  LIN  1077
0000H  1FCAH  LIN  1079
0000H  1FD1H  LIN  1081
0000H  1FDFH  LIN  1083
0000H  1FF4H  LIN  1085
0000H  2005H  LIN  1087
0000H  2010H  LIN  1089
0000H  201CH  LIN  1091
0000H  2026H  LIN  1093
0000H  2043H  LIN  1095
```

```
0000H  1F49H  LIN   1096
0000H  1F54H  LIN   1098
0000H  1F60H  LIN   1100
0000H  1F6AH  LIN   1102
0000H  1F78H  LIN   1104
0000H  1F7FH  LIN   1106
0000H  1F88H  LIN   1115
0000H  1F84H  LIN   1118
0000H  1F89H  LIN   1121
0000H  1FD6H  LIN   1123
0000H  1FF3H  LIN   1125
0000H  1FFDH  LIN   1127
0000H  200DH  LIN   1129
0000H  201CH  LIN   1132
0000H  2020H  LIN   1134
0000H  203FH  LIN   1136
0000H  2048H  LIN   1138
0000H  2058H  LIN   1140
0000H  2073H  LIN   1142
0000H  2088H  LIN   1145
0000H  20A1H  LIN   1148
0000H  20A7H  LIN   1150
0000H  20BAH  LIN   1153
0000H  20C6H  LIN   1156
0000H  20CBH  LIN   1158
0000H  20E5H  LIN   1160
0000H  20FDH  LIN   1163
0000H  211DH  LIN   1168
0000H  2127H  LIN   1170
0000H  212CH  LIN   1173
0000H  2137H  LIN   1176
0000H  2148H  LIN   1178
0000H  2159H  LIN   1181
0000H  2169H  LIN   1184
0000H  216EH  LIN   1186
0000H  2183H  LIN   1189
0000H  2193H  LIN   1192
0000H  2198H  LIN   1194
0000H  21AAH  LIN   1197
0000H  21B8H  LIN   1200
0000H  21BDH  LIN   1202
0000H  21D1H  LIN   1205
0000H  21D8H  LIN   1208
0000H  21E8H  LIN   1211
0000H  21F6H  LIN   1213
0000H  2203H  LIN   1215
0000H  220BH  LIN   1217
0000H  2210H  LIN   1220
0000H  0102H  LIN   1223

0000H  1F50H  LIN   1097
0000H  1F5EH  LIN   1099
0000H  1F67H  LIN   1101
0000H  1F73H  LIN   1103
0000H  1F7AH  LIN   1105
0000H  1F82H  LIN   1114
0000H  1FADH  LIN   1117
0000H  1FB7H  LIN   1119
0000H  1FC8H  LIN   1122
0000H  1FEEH  LIN   1124
0000H  1FF8H  LIN   1126
0000H  2003H  LIN   1128
0000H  2017H  LIN   1131
0000H  2029H  LIN   1133
0000H  202FH  LIN   1135
0000H  2045H  LIN   1137
0000H  2055H  LIN   1139
0000H  2060H  LIN   1141
0000H  2076H  LIN   1143
0000H  209CH  LIN   1147
0000H  20A5H  LIN   1149
0000H  20AAH  LIN   1151
0000H  20C2H  LIN   1155
0000H  20C8H  LIN   1157
0000H  20DDH  LIN   1159
0000H  20F3H  LIN   1162
0000H  210AH  LIN   1165
0000H  2122H  LIN   1169
0000H  2129H  LIN   1172
0000H  2132H  LIN   1175
0000H  2143H  LIN   1177
0000H  2156H  LIN   1180
0000H  215BH  LIN   1182
0000H  216CH  LIN   1185
0000H  217CH  LIN   1188
0000H  2185H  LIN   1190
0000H  2196H  LIN   1193
0000H  21A6H  LIN   1196
0000H  21AAH  LIN   1198
0000H  21BBH  LIN   1201
0000H  21CBH  LIN   1204
0000H  21D3H  LIN   1207
0000H  21E1H  LIN   1210
0000H  21EFH  LIN   1212
0000H  21FCH  LIN   1214
0000H  2209H  LIN   1216
0000H  220EH  LIN   1219
0000H  2210H  LIN   1222
```


MEMORY MAP OF MODULE SCD
READ FROM FILE SYSTAT.LNK
WRITTEN TO FILE SYSTAT.

SEGMENT MAP

START	STOP	LENGTH	ALIGN	NAME	CLASS
00000H	02211H	2212H	G	CODE	CODE
10000H	10107H	0108H	G	DATS	DATA

10108H	115FFH	14F8H	W	DATA	DATA
11600H	11651H	0052H	W	STACK	STACK
11652H	12006H	09B5H	W	CONST	CONST
12010H	12010H	0000H	C	??SEG	
12010H	12010H	0000H	W	MEMORY	MEMORY

GROUP MAP

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