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


      $TITLE('CONCURRENT CP/M 86 --- DIR 1.0 ')
      $compact

      /* Conditional compile:
          rsp=0ffh        produce a DIR.RSP type of file
          rsp=0           produce a DIR.CMD file
       */
      $set(rsp=0h)

      $include(dirm.plm)
=     /* dirm:
=         This is the module included by DIRRSP or DIRCMD.
=
=       Revised:
=         Jan  80  by Thomas Rolander
=         July 81  by Doug Huskey
=         June 82  by Bill Fitler
=         July 82  by Danny Horovitz (made an RSP)
=         Dec  82  by Fran Borda     (conditional comp)
=         Mar  83  by Bill Fitler    (   "          "  )
=         Mar  83  by Danny Horovitz (control C fixes)
=
=       Conditional compile:
=         rsp=0ffh        produce a DIR.RSP type of file
=         rsp=0           produce a DIR.CMD file
=      */
=
=     /**** Vax commands to compile DIR.RSP and DIR.CMD:
=
=         $ ccpmsetup
=         $ plm86 dircmd.plm 'p1' 'p2' 'p3' 'p4' optimize(3) debug
=         $ link86 f1:scd.obj, dircmd.obj  to dircmd.lnk
=         $ loc86 dircmd.lnk od(sm(code,dats,data,stack,const))-
=               ad(sm(code(0), dats(10000h)))  ss(stack(+32)) to dircmd.
=         $ h86 dircmd
=         $ ! DIR.RSP
=         $ ! Note: separate code and data
=         $ asm86 rhdir.a86       !Rsp Header DIR
=         $ plm86 dirrsp.plm 'p1' 'p2' 'p3' 'p4' optimize(3) debug
=         $ link86 rhdir.obj, dirrsp.obj  to dirrsp.lnk
=         $ loc86 dirrsp.lnk od(sm(code,dats,data,stack,const))-
=               ad(sm(code(0), dats(10000h)))  ss(stack(0)) to dirrsp.
=         $ h86 dirrsp
=
=     **** Then, on a micro:
=         A>vax dircmd.h86 $fans
=         A>vax dirrsp.h86 $fans
=         A>gencmd dircmd data[b1000]
=         A>ren dir.cmd=dircmd.cmd
=         A>gencmd dirrsp data[b1000]
=         A>ren dir.rsp=dirrsp.cmd
```

```
=
=   **** Notes: Both DIRCMD.PLM and DIRRSP.PLM include DIRM.PLM, after setting
=           RSP flag appropriately.
=
=   ****/
=
=
1   =   dir:
=   =   do;
=
=   =   $include (:f1:copyrt.lit)
=   =1
=   =1  /*
=   =1    Copyright (C) 1983
=   =1    Digital Research
=   =1    P.O. Box 579
=   =1    Pacific Grove, CA 93950
=   =1  */
=   =1
=
=   =   $include (:f1:comlit.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          cr              lit             '13',
=   =1          lf              lit             '10',
=   =1          tab             lit             '9';
=   =1
=
=   =   $include (:f1:mfunc.lit)
=   =1
=   =1  /* Concurrent CP/M function numbers */
=   =1
3   1   =1  dcl         m$prtbuf            lit     '9',
=   =1              m$select            lit     '14',
=   =1              m$openf             lit     '15',
=   =1              m$closef            lit     '16',
=   =1              m$deletef           lit     '19',
=   =1              m$readf             lit     '20',
=   =1              m$writef            lit     '21',
=   =1              m$makef             lit     '22',
=   =1              m$getlogin          lit     '24',
=   =1              m$curdsk            lit     '25',
=   =1              m$setdma            lit     '26',
=   =1              m$setatt            lit     '30',
=   =1              m$setusr            lit     '32',
=   =1              m$readrf            lit     '33',
=   =1              m$writerf           lit     '34',
=   =1              m$resetdrv          lit     '37',
=   =1              m$errmode           lit     '45',
=   =1              m$dirbios           lit     '50',
```

```
        =1              m$makeq             lit     '134',
        =1              m$openq             lit     '135',
        =1              m$deleteq           lit     '136',
        =1              m$readq             lit     '137',
        =1              m$creadq            lit     '138',
        =1              m$writeq            lit     '139',
        =1              m$cwriteq           lit     '140',
        =1              m$delay             lit     '141',
        =1              m$dispatch          lit     '142',
        =1              m$setprior          lit     '145',
        =1              m$attach            lit     '146',
        =1              m$detach            lit     '147',
        =1              m$setcns            lit     '148',
        =1              m$parse             lit     '152',
        =1              m$getcns            lit     '153',
        =1              m$sysdat            lit     '154',
        =1              m$getpd             lit     '156',
        =1              m$abort             lit     '157';
        =1
        =1  /* Internal calls */
        =1
4   1   =1  dcl             mi$sleep            lit     '0212H',
        =1              mi$wakeup           lit     '0213H';
        =1
        =
        =   $include (:f1:proces.lit)
        =1
        =1  /*
        =1      Proces Literals MP/M-8086 II
        =1  */
        =1
5   1   =1  declare pnamsiz literally '8';
        =1
6   1   =1  declare pd$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
7   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
8   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
9   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
10   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';
        =
        =    $include (:f1:qd.lit)
        =1
        =1   /* Queue Descriptor */
        =1
11   1  =1   dcl qnamsiz lit '8';
        =1
12   1  =1   dcl qd$structure lit 'structure(
        =1     link  word,
        =1     net byte,
        =1     org 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
13   1  =1   dcl qf$mx       lit '001h'; /* Mutual Exclusion */
14   1  =1   dcl qf$keep     lit '002h'; /* NO DELETE        */
15   1  =1   dcl qf$hide     lit '004h'; /* Not User writable    */
16   1  =1   dcl qf$rsp      lit '008h'; /* rsp queue        */
17   1  =1   dcl qf$table    lit '010h'; /* from qd table    */
18   1  =1   dcl qf$rpl      lit '020h'; /* rpl queue        */
19   1  =1   dcl qf$dev      lit '040h'; /* device queue     */
        =1
        =1   /* Queue Parameter Block */
        =1
20   1  =1   dcl qpb$structure lit 'structure(
        =1     flgs    byte,
        =1     net     byte,
        =1     qaddr   word,
        =1     nmsgs   word,
        =1     buffptr word,
        =1     name (qnamsiz) byte )';
```

```
        =1
        =
        =     /**************************************
        =      *                                    *
        =      *       B D O S   INTERFACE          *
        =      *                                    *
        =      **************************************/
        =
        =
21   1  =     mon1:
        =       procedure (func,info) external;
22   2  =         declare func byte;
23   2  =         declare info address;
24   2  =       end mon1;
        =
25   1  =     mon2:
        =       procedure (func,info) byte external;
26   2  =         declare func byte;
27   2  =         declare info address;
28   2  =       end mon2;
        =
29   1  =     mon3:
        =       procedure (func,info) address external;
30   2  =         declare func byte;
31   2  =         declare info address;
32   2  =       end mon3;
        =
33   1  =     mon4:
        =       procedure (func,info) pointer external;
34   2  =         declare func byte;
35   2  =         declare info address;
36   2  =       end mon4;
        =
        =
        =
37   1  =   patch: procedure public;    /* dummy area for patching code segments */
38   2  =     declare i address;
        =     /* first statement = 9 bytes, rest are 5 bytes */
39   2  =       i=i+5;  i=i+5;  i=i+5;  i=i+5;  i=i+5;
44   2  =       i=i+5;  i=i+5;  i=i+5;  i=i+5;  i=i+5;
49   2  =       i=i+5;  i=i+5;  i=i+5;  i=i+5;  i=i+5;    /* about 79 bytes */
54   2  =     end patch;
        =
        =
        =   $if rsp
        =     declare fcb (36) byte;                /* 1st default fcb    */
        =     declare fcb16 (1) byte at (@fcb(16));  /* 2nd default fcb    */
        =   $else
55   1  =     declare fcb (1)   byte    external;    /* 1st default fcb    */
56   1  =     declare fcb16 (1) byte    external;    /* 2nd default fcb    */
        =   $endif
        =
57   1  =     write$console:
        =       procedure (char);
58   2  =         declare char byte;
59   2  =         call mon1 (2,char);
60   2  =       end write$console;
```

```
      =
61  1 =    print$buf:
      =      procedure (buffer$address);
62  2 =        declare buffer$address address;
63  2 =        call mon1 (9,buffer$address);
64  2 =      end print$buf;
      =
65  1 =    check$ctrl$c:
      =      procedure byte;
      =  $if rsp
      =        if (dir$pd.flag and pf$ctlC) <> 0 then
      =        do;
      =           dir$pd.flag = dir$pd.flag and not double(pf$ctlC);
      =           return(true);
      =        end;
      =  $endif
66  2 =        return (false);
67  2 =      end check$ctrl$c;
      =
68  1 =    search$first:
      =      procedure (fcb$address) byte;
69  2 =        declare fcb$address address;
70  2 =        return mon2 (17,fcb$address);
71  2 =      end search$first;
      =
72  1 =    search$next:
      =      procedure (fcb$address) byte;
73  2 =        declare fcb$address address;
74  2 =        return mon2 (18,fcb$address);
75  2 =      end search$next;
      =
76  1 =    setdma: procedure(dma);
77  2 =      declare dma address;
78  2 =      call mon1(26,dma);
79  2 =      end setdma;
      =
      =
80  1 =    get$user$code:
      =      procedure byte;
81  2 =        return mon2 (32,0ffh);
82  2 =      end get$user$code;
      =
83  1 =     set$user$code:
      =      procedure(user);
84  2 =        declare user byte;
85  2 =        call mon1 (32,user);
86  2 =      end set$user$code;
      =
87  1 =  terminate:
      =      procedure;
88  2 =        call mon1 (0,0);
89  2 =      end terminate;
      =
90  1 =  declare
      =      parse$fn structure (
      =        buff$adr  address,
      =        fcb$adr   address),
```

```
          =       delimiter based parse$fn.buff$adr byte;
 91   1   =     declare tail$len address;
          =
 92   1   =     parse: procedure address;
 93   2   =       return mon3(152,.parse$fn);
 94   2   =       end parse;
          =
          =
 95   1   =       crlf:
          =       procedure;
 96   2   =         call write$console (0dh);
 97   2   =         call write$console (0ah);
 98   2   =       end crlf;
          =
          =
          =   /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
          =
          =
          =                   * * *  GLOBAL VARIABLES  * * *
          =
          =
          =    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
          =
          =
          =
 99   1   =   declare dir$title (*) byte initial
          =     ('Directory for User  x:','$');
          =
100   1   =   declare (sys,temp,dcnt,cnt,user) byte;
101   1   =   declare
          =      i          byte,
          =      new$user   byte,
          =      sys$exists byte,
          =      incl$sys   byte,
          =      option     byte;
          =
102   1   =   declare
          =      dirbuf (128) byte;
          =
          =
          =   /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
          =
          =
          =                   * * *  DIRECTORY DISPLAY  * * *
          =
          =
          =    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
          =
          =
          =                     /* display directory heading */
103   1   =   heading: procedure;
          =
104   2   =       if user > 9 then
105   2   =       do;
106   3   =         dir$title(19) = '1';
107   3   =         dir$title(20) = user - 10 + '0';
108   3   =       end;
```

```
        =       else
109   2 =       do;
110   3 =         dir$title(19) = ' ';
111   3 =         dir$title(20) = user + '0';
112   3 =       end;
113   2 =       call print$buf (.dir$title);
114   2 =       end heading;
        =
        =   /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
        =
        =   /*
        =   help: procedure;
        =       call mon1(m$prt$buf, .(cr, lf, tab, tab, tab ,'DIR EXAMPLES$'));
        =       call mon1(m$prt$buf, .(cr, lf, lf, 'dir', tab, tab,
        =         '(show all directory files on current drive and user)
        =       call mon1(m$prt$buf, .(cr, lf, 'dir [g3]', tab, tab, tab, tab,
        =         '(show non system files under user 3)$'));
        =       call mon1(m$prt$buf, .(cr, lf, 'dir a: b: [s]', tab, tab, tab,
        =         tab, '(show all files under current user on a: and b:)$'));
        =       call terminate;
        =   end help;
        =   */
        =
        =   /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
        =
        =                       /* do next directory display */
115   1 =   directory: procedure boolean;
        =
116   2 =       shown$nothing = false;
117   2 =       if new$user then do;
119   3 =           call heading;
120   3 =           new$user = false;
121   3 =           end;
122   2 =       sys$exists = false;
123   2 =       cnt = -1;
        =       /* if drive is 0 (default)
        =          then set to current disk */
124   2 =       if fcb(0) = 0
        =         then fcb(0) = mon2 (m$curdsk,0) + 1;
126   2 =       if fcb(1) = ' ' then
        =       /* check for blank filename => wildcard */
127   2 =       do i = 1 to 11;
128   3 =         fcb(i) = '?';
129   3 =       end;
        =       /* get first file */
130   2 =       if (dcnt := search$first (.fcb)) <> 0ffh then
131   2 =       do while dcnt <> 0ffh;
132   3 =         temp = shl(dcnt,5);
133   3 =         sys = ((dirbuf(temp+10) and 80h) = 80h);
134   3 =         if (dirbuf(temp) = user) and
        =            (incl$sys or not sys) then
135   3 =         do;
136   4 =           if ((cnt:=cnt+1) mod 4) = 0 then
137   4 =           do;
138   5 =             call crlf;
139   5 =             call write$console ('A'+fcb(0)-1);
140   5 =           end;
```

```
        =           else
141   4 =           do;
142   5 =             call write$console (' ');
143   5 =           end;
144   4 =           call write$console (':');
145   4 =           call write$console (' ');
146   4 =           do i = 1 to 11;
147   5 =             if i = 9 then call write$console (' ');
149   5 =             call write$console
        =               (dirbuf(temp+i) and 7fh);
150   5 =             if check$ctrl$c then
151   5 =               return(false);
152   5 =           end;
153   4 =         end;
154   3 =         else if sys then
155   3 =           sys$exists = true;
        =         dcnt = search$next (.fcb);
157   3 =       end;
158   2 =       if cnt = -1 then
159   2 =       do;
160   3 =         call print$buf (.(0dh,0ah,
        =           'File not found.','$'));
161   3 =       end;
162   2 =       if sys$exists then
163   2 =         call print$buf (.(0dh,0ah,
        =           'System Files Exist','$'));
164   2 =       return(true);
165   2 =       end directory;
        =
        =   /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        =
        =
        =                       * * *  PARSING  * * *
        =
        =
        =    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
        =
        =
        =         /* parse one file name, return true if got one */
166   1 =   parse$file: procedure boolean;
167   2 =     dcl i address;
168   2 =     dcl buf based parse$fn.buff$adr (1) byte;
169   2 =     dcl parse$ret address;
        =
170   2 =     if (parse$ret := parse$fn.buff$adr) = 0 then
171   2 =       return(false);
172   2 =     fcb(0), i = 0;
173   2 =     parse$ret = parse;                          /* kludge around        */
174   2 =     do while parse$ret = 0 and buf(i) = '[';  /* parse file name bugs */
175   3 =       if (i := findb(@buf(i), ']', tail$len - i)) <> 0ffffh then
176   3 =       do;
177   4 =         parse$fn.buff$adr = .buf(i) + 1;        /* skip right bracket */
178   4 =         i = 0;
179   4 =         parse$ret = parse;
180   4 =       end;
        =       else
181   3 =         buf(i) = 0;
```

```
182   3  =     end;
183   2  =     parse$fn.buff$adr = parse$ret;
         =
         =
184   2  =     if parse$fn.buff$adr <> 0ffffh then
185   2  =     do;
186   3  =       if fcb(1) <> ' ' then
187   3  =       do;
188   4  =         if parse$fn.buff$adr <> 0 and delimiter <> '[' and delimiter <> 0 then
189   4  =           parse$fn.buff$adr = parse$fn.buff$adr + 1;
190   4  =         return(true);              /* parse$fn.buff$adr could = 0 */
191   4  =       end;
192   3  =       else if fcb(0) <> 0 and fcb(1) = ' ' then  /* drive spec */
193   3  =       do;
194   4  =         call setb('?', @fcb(1), 11);
195   4  =         return(true);
196   4  =       end;
         =     end;
         =     else /* if parse$fn.buff$adr = 0ffffh then */
198   2  =     do;
199   3  =       call print$buf(.(cr, lf, 'Invalid filespec.$'));
200   3  =       shown$nothing = false;          /* don't show directory */
201   3  =       return(false);    /* also if parse$fn.buf$adr = 0 and fcb(0) = ' ' */
202   3  =     end;
203   2  =   end parse$file;
         =
         =
         =   /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
         =
         =
         =              /* parse & interpret all options - assume global */
204   1  =   parse$options: procedure boolean;
205   2  =     dcl (n,i) word;
206   2  =     dcl (options, in$brackets, error) boolean;
207   2  =     i = 0;           /* parse file name doesn't work with delimiters */
208   2  =     parse$fn.fcb$adr = .dirbuf;
209   2  =     error = false;
210   2  =     options = true;
211   2  =     do while options and not error;
212   3  =       if (n := findb(@tbuff(i), '[', tail$len - i)) = 0ffffh then
213   3  =         options = false;
         =       else
214   3  =       do;
215   4  =         i = i + n + 1;
216   4  =         parse$fn.buff$adr = .tbuff(i);
217   4  =         in$brackets = true;
218   4  =         do while in$brackets and not error;
219   5  =           if (parse$fn.buff$adr := parse) <> 0ffffh then
220   5  =           do;
221   6  =             if dirbuf(1) = 'S' then
222   6  =               incl$sys = true;
223   6  =             else if dirbuf(1) = 'G' then
224   6  =             do;
225   7  =               if dirbuf(3) <> ' ' then
226   7  =                 temp = dirbuf(3) - '0' * 10;
227   7  =               else if dirbuf(2) <> ' ' then
228   7  =                 temp = dirbuf(2) - '0';
```

```
                =                 if temp < 16 then
230     7       =                 do;
231     8       =                   call mon1(m$setusr, (user:=temp));
232     8       =                   new$user = true;
233     8       =                 end;
234     7       =               end;
                =               else
235     6       =                 error = true;
236     6       =             end;  /* if parse */
237     5       =             if delimiter = ']' or parse$fn.buff$adr = 0 or
                =               parse$fn.buff$adr = 0ffffh then
238     5       =               in$brackets = false;
239     5       =           end;  /* while in$brackets */
240     4       =         end;  /* else                */
241     3       =       end;  /* while options       */
                =
242     2       =       if error then
243     2       =       do;
244     3       =         call print$buf(.(cr, lf, 'Invalid Command Option$'));
245     3       =         return(false);
                =         /* call help; */
246     3       =       end;
247     2       =       return(true);
248     2       =     end parse$options;
                =
                =
                =
                =     /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                =
                =
                =                 * * *  M A I N   P R O G R A M  * * *
                =
                =
                =      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
                =
                =     $if rsp
                =      declare cpd$pointer pointer;              /* Calling PD pointer stuff */
                =      declare cpd$ptr structure (
                =         offset address, segment address) at (@cpd$pointer);
                =      declare calling$pd based cpd$pointer pd$structure;
                =      declare dpd$pointer pointer;             /* DIR RSP PD pointer stuff */
                =      declare dpd$ptr structure (
                =         offset address, segment address) at (@dpd$pointer);
                =      declare dir$pd based dpd$pointer pd$structure;
                =      declare qdbuf (131) byte;
                =      declare dirqd qd$structure initial
                =        (0, 0, 0, qf$keep + qf$rsp, 'DIR     ', 131, 1, 0, 0, 0, 0, .qdbuf);
                =      declare qpbbuf (131) byte;
                =      declare cpd$offset address at (@qpbbuf(0));
                =      declare tbuff (128) byte at (@qpbbuf(2));
                =      declare dirqpb qpb$structure initial
                =        (0, 0, 0, 0, .qpbbuf, ']IR     ');
                =     $else
249     1       =      declare tbuff (128) byte external;
                =     $endif
                =
250     1       =     declare shown$nothing boolean;
```

```
        =
251   1 =   plmstart: procedure public;
        =
        =     /* initialization */
        =
        =   $if rsp
        =     call mon1(m$make$q, .dirqd);
        =     call mon1(m$open$q, .dirqpb);
        =     call mon1(m$set$prior, 200);      /* Set priority same as other transients*/
        =   $else
252   2 =     user = get$user$code;           /* ?????   whf */
253   2 =     incl$sys = (fcb(1) = 'S');        /* ?????  why exclude if rsp?  whf */
        =   $endif
        =
254   2 =     call setdma(.dirbuf);
        =   $if rsp
        =     cpd$pointer,dpd$pointer = mon4(m$sysdat, 0);
        =     dpd$ptr.offset = mon3(m$getpd,0);
        =     /* Don't allow control S, turn on tempkeep for control C checking */
        =     dir$pd.flag = dir$pd.flag or pf$noctls or pf$tempkeep;
        =     /* Read RSP Queue forever */
        =     do forever;
        =        call mon1(m$readq, .dirqpb);
        =        dir$pd.flag = dir$pd.flag and not double(pf$ctlC);
        =        /* Could be on from last DIR */
        =        /* set defaults same as calling process's, have both PDs so will poke */
        =        /* and not call O.S. */
        =        cpd$ptr.offset = cpd$offset;
        =        call mon1(m$setcns, calling$pd.cns);
        =        call mon1(m$setusr, (user := calling$pd.user));
        =        call mon1(m$select, calling$pd.dsk);
        =   $endif
255   2 =        new$user = true;
256   2 =        sys$exists, incl$sys = false;
257   2 =        tail$len = findb(@tbuff, 0, 128);
        =
        =        /* scan for options - all are global */
        =
258   2 =        if not parse$options then
259   2 =          goto done;                   /* option error */
        =
        =        /* do command line */
        =
260   2 =        shown$nothing = true;
        =   $if rsp
        =        parse$fn.buff$adr = .tbuff;
        =   $else
261   2 =        parse$fn.buff$adr = (.tbuff) + 1;/* Skip # of bytes in buffer */
        =   $endif
262   2 =        parse$fn.fcb$adr = .fcb;
263   2 =        do while parse$file;           /* false when no more files, sets */
264   3 =          if not directory then        /* shown$nothing=false if parsing error */
265   3 =            goto done;                 /* directory = false if console inpute */
266   3 =        end;
        =
267   2 =        if shown$nothing then         /* no files specified on command line */
268   2 =        do;
```

```
269   3  =          call setb('?', .fcb(1), 11);
270   3  =          if not directory then
271   3  =            goto done;                    /* false on console input */
272   3  =        end;
273   2  =    done:
          =    $if rsp
          =        call mon1(m$detach, 0);
          =        end; /* do forever */
          =    $else
          =        call terminate;
          =    $endif
          =
274   2  =    end plmstart;
          =
275   1  =    end dir;
```

CROSS-REFERENCE LISTING

| DEFN | ADDR | SIZE | NAME, ATTRIBUTES, AND REFERENCES |
|---|---|---|---|
| 2 | | | BOOLEAN. . . . . LITERALLY 115 166 204 206 250 |
| 168 | 0000H | 1 | BUF. . . . . BYTE BASED(PARSEFN.BUFFADR) ARRAY(1) 174 175 177 181 |
| 90 | 0000H | 2 | BUFFADR. . . . . WORD MEMBER(PARSEFN) 90 168 170 174 175 177 181 183 184 188 189 216 219 237 261 |
| 61 | 0004H | 2 | BUFFERADDRESS. . . WORD PARAMETER AUTOMATIC 62 63 |
| 57 | 0004H | 1 | CHAR . . . . . BYTE PARAMETER AUTOMATIC 58 59 |
| 65 | 007AH | 7 | CHECKCTRLC . . . . PROCEDURE BYTE STACK=0002H 150 |
| 100 | 002AH | 1 | CNT. . . . . BYTE 123 136 158 |
| 2 | | | CR . . . . . LITERALLY 199 244 |
| 95 | 00F0H | 17 | CRLF . . . . . PROCEDURE STACK=000EH 138 |
| 2 | | | DCL. . . . . LITERALLY |
| 100 | 0029H | 1 | DCNT . . . . . BYTE 130 131 132 156 |
| 90 | 0000H | 1 | DELIMITER. . . . . BYTE BASED(PARSEFN.BUFFADR) 188 237 |
| 1 | 0002H | | DIR. . . . . PROCEDURE STACK=0000H |
| 102 | 0031H | 128 | DIRBUF . . . . . BYTE ARRAY(128) 133 134 149 208 221 223 225 226 227 228 254 |
| 115 | 012DH | 339 | DIRECTORY. . . . . PROCEDURE BYTE STACK=0012H 264 270 |
| 99 | 0010H | 23 | DIRTITLE . . . . . BYTE ARRAY(23) INITIAL 106 107 110 111 113 |
| 76 | 0004H | 2 | DMA. . . . . WORD PARAMETER AUTOMATIC 77 78 |
| 273 | 04E9H | | DONE . . . . . LABEL 259 265 271 |
| 206 | 00B3H | 1 | ERROR. . . . . BYTE 209 211 218 235 242 |
| 2 | | | FALSE. . . . . LITERALLY 66 116 120 122 151 171 200 201 209 213 238 245 256 |
| 55 | 0000H | 1 | FCB. . . . . BYTE ARRAY(1) EXTERNAL(4) 124 125 126 128 130 139 156 172 186 192 194 253 262 269 |
| 56 | 0000H | 1 | FCB16. . . . . BYTE ARRAY(1) EXTERNAL(5) |
| 72 | 0004H | 2 | FCBADDRESS . . . . WORD PARAMETER AUTOMATIC 73 74 |
| 68 | 0004H | 2 | FCBADDRESS . . . . WORD PARAMETER AUTOMATIC 69 70 |
| 90 | 0002H | 2 | FCBADR . . . . . WORD MEMBER(PARSEFN) 208 262 |
| | | | FINDB. . . . . BUILTIN 175 212 257 |
| 2 | | | FOREVER. . . . . LITERALLY |
| 33 | 0000H | 1 | FUNC . . . . . BYTE PARAMETER 34 |
| 25 | 0000H | 1 | FUNC . . . . . BYTE PARAMETER 26 |
| 29 | 0000H | 1 | FUNC . . . . . BYTE PARAMETER 30 |
| 21 | 0000H | 1 | FUNC . . . . . BYTE PARAMETER 22 |
| 80 | 00B1H | 15 | GETUSERCODE. . . . PROCEDURE BYTE STACK=0008H 252 |
| 103 | 0101H | 44 | HEADING. . . . . PROCEDURE STACK=000EH 119 |
| 38 | 0000H | 2 | I. . . . . WORD 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 |
| 167 | 0008H | 2 | I. . . . . WORD 172 174 175 177 178 181 |
| 101 | 002CH | 1 | I. . . . . BYTE 127 128 146 147 149 |
| 205 | 000EH | 2 | I. . . . . WORD 207 212 215 216 |
| 206 | 00B2H | 1 | INBRACKETS . . . . BYTE 217 218 238 |
| 101 | 002FH | 1 | INCLSYS. . . . . BYTE 134 222 253 256 |
| 25 | 0000H | 2 | INFO . . . . . WORD PARAMETER 27 |
| 33 | 0000H | 2 | INFO . . . . . WORD PARAMETER 35 |
| 29 | 0000H | 2 | INFO . . . . . WORD PARAMETER 31 |
| 21 | 0000H | 2 | INFO . . . . . WORD PARAMETER 23 |
| 2 | | | LF . . . . . LITERALLY 199 244 |

```
   2                LIT. . . . . .   LITERALLY          2    3    4    8    9   10   11   12   13   14   15
                                       16   17   18   19   20
   3                MABORT . . . . .  LITERALLY
   3                MATTACH. . . . .  LITERALLY
   3                MCLOSEF. . . . .  LITERALLY
   3                MCREADQ. . . . .  LITERALLY
   3                MCURDSK. . . . .  LITERALLY        125
   3                MCWRITEQ . . . .  LITERALLY
   3                MDELAY . . . . .  LITERALLY
   3                MDELETEF . . . .  LITERALLY
   3                MDELETEQ . . . .  LITERALLY
   3                MDETACH. . . . .  LITERALLY
   3                MDIRBIOS . . . .  LITERALLY
   3                MDISPATCH. . . .  LITERALLY
   3                MERRMODE . . . .  LITERALLY
   3                MGETCNS. . . . .  LITERALLY
   3                MGETLOGIN. . . .  LITERALLY
   3                MGETPD . . . . .  LITERALLY
   4                MISLEEP. . . . .  LITERALLY
   4                MIWAKEUP . . . .  LITERALLY
   3                MMAKEF . . . . .  LITERALLY
   3                MMAKEQ . . . . .  LITERALLY
  21  0000H         MON1 . . . . . .  PROCEDURE EXTERNAL(0) STACK=0000H          59   63   78   85   88  231
  25  0000H         MON2 . . . . . .  PROCEDURE BYTE EXTERNAL(1) STACK=0000H          70   74   81  125
  29  0000H         MON3 . . . . . .  PROCEDURE WORD EXTERNAL(2) STACK=0000H          93
  33  0000H         MON4 . . . . . .  PROCEDURE POINTER EXTERNAL(3) STACK=0000H
   3                MOPENF . . . . .  LITERALLY
   3                MOPENQ . . . . .  LITERALLY
   3                MPARSE . . . . .  LITERALLY
   3                MPRTBUF. . . . .  LITERALLY
   3                MREADF . . . . .  LITERALLY
   3                MREADQ . . . . .  LITERALLY
   3                MREADRF. . . . .  LITERALLY
   3                MRESETDRV. . . .  LITERALLY
   3                MSELECT. . . . .  LITERALLY
   3                MSETATT. . . . .  LITERALLY
   3                MSETCNS. . . . .  LITERALLY
   3                MSETDMA. . . . .  LITERALLY
   3                MSETPRIOR. . . .  LITERALLY
   3                MSETUSR. . . . .  LITERALLY        231
   3                MSYSDAT. . . . .  LITERALLY
   3                MWRITEF. . . . .  LITERALLY
   3                MWRITEQ. . . . .  LITERALLY
   3                MWRITERF . . . .  LITERALLY
 205  000CH     2   N. . . . . . . .  WORD         212  215
 101  002DH     1   NEWUSER. . . . .  BYTE         117  120  232  255
   2                NO . . . . . . .  LITERALLY
 101  0030H     1   OPTION . . . . .  BYTE
 206  00B1H     1   OPTIONS. . . . .  BYTE         210  211  213
  92  00E1H    15   PARSE. . . . . .  PROCEDURE WORD STACK=0008H              173  179  219
 166  0280H   213   PARSEFILE. . . .  PROCEDURE BYTE STACK=000EH              263
  90  0002H     4   PARSEFN. . . . .  STRUCTURE          90   93  168  170  174  175  177  181  183  184  188
                                       189  208  216  219  237  261  262
 204  0355H   269   PARSEOPTIONS . .  PROCEDURE BYTE STACK=000EH              258
 169  000AH     2   PARSERET . . . .  WORD         170  173  174  179  183
  37  0002H    85   PATCH. . . . . .  PROCEDURE PUBLIC STACK=0002H
  10                PCM11. . . . . .  LITERALLY
```

CCP/M-86 2.0 V.4
COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CC-104

```
  10                PCMCTLC. . . . . LITERALLY
  10                PCMCTLO. . . . . LITERALLY
  10                PCMCTLS. . . . . LITERALLY
  10                PCMROUT. . . . . LITERALLY
  10                PCMRSX . . . . . LITERALLY
   6                PDHDR. . . . . . LITERALLY
   7                PDSTRUCTURE. . . . LITERALLY
   9                PFACTIVE . . . . . LITERALLY
   9                PFCHILDABORT . . . LITERALLY
   9                PFCTLC . . . . . LITERALLY
   9                PFCTLD . . . . . LITERALLY
   9                PFKEEP . . . . . LITERALLY
   9                PFKERNAL . . . . . LITERALLY
   9                PFNOCTLS . . . . . LITERALLY
   9                PFPURE . . . . . LITERALLY
   9                PFRAW. . . . . LITERALLY
   9                PFRESOURCE . . . . LITERALLY
   9                PFSYS. . . . . LITERALLY
   9                PFTABLE. . . . . LITERALLY
   9                PFTEMPKEEP . . . . LITERALLY
 251  0462H   140   PLMSTART . . . . . PROCEDURE PUBLIC STACK=0016H
   5                PNAMSIZ. . . . . LITERALLY
  61  006AH    16   PRINTBUF . . . . . PROCEDURE STACK=000AH          113  160  163  199  244
   8                PSCIOWAIT. . . . . LITERALLY
   8                PSDELAY. . . . . LITERALLY
   8                PSDQ . . . . . LITERALLY
   8                PSFLAGWAIT . . . . LITERALLY
   8                PSNQ . . . . . LITERALLY
   8                PSPOLL . . . . . LITERALLY
   8                PSRUN. . . . . LITERALLY
   8                PSSLEEP. . . . . LITERALLY
   8                PSSWAP . . . . . LITERALLY
   8                PSTERM . . . . . LITERALLY
  12                QDSTRUCTURE. . . . LITERALLY
  19                QFDEV. . . . . LITERALLY
  15                QFHIDE . . . . . LITERALLY
  14                QFKEEP . . . . . LITERALLY
  13                QFMX . . . . . LITERALLY
  18                QFRPL. . . . . LITERALLY
  16                QFRSP. . . . . LITERALLY
  17                QFTABLE. . . . . LITERALLY
  11                QNAMSIZ. . . . . LITERALLY
  20                QPBSTRUCTURE . . . LITERALLY
  68  0081H    16   SEARCHFIRST. . . . PROCEDURE BYTE STACK=000AH          130
  72  0091H    16   SEARCHNEXT . . . . PROCEDURE BYTE STACK=000AH          156
                    SETB . . . . . BUILTIN          194  269
  76  00A1H    16   SETDMA . . . . . PROCEDURE STACK=000AH          254
  83  00C0H    19   SETUSERCODE. . . . PROCEDURE STACK=000AH
                    SHL. . . . . . . BUILTIN        132
 250  0084H     1   SHOWNNOTHING . . . BYTE          116  200  260  267
 100  0027H     1   SYS. . . . . BYTE          133  134  154
 101  002EH     1   SYSEXTSTS. . . . . BYTE          122  155  162  256
   2                TAB. . . . . . . LITERALLY
  91  0006H     2   TAILLEN. . . . . WORD          175  212  257
 249  0000H   128   TBUFF. . . . . . BYTE ARRAY(128) EXTERNAL(6)          212  216  257  261
 100  0028H     1   TEMP . . . . . . BYTE          132  133  134  149  226  228  229  231
  87  00D3H    14   TERMINATE. . . . . PROCEDURE STACK=0008H          273
```

```
     2                TRUE . . . . . .   LITERALLY        155  164  190  195  210  217  222  232  235  247  255
                                           260
   100  0028H     1  USER . . . . . .   BYTE        104  107  111  134  231  252
    83  0004H     1  USER . . . . . .   BYTE PARAMETER AUTOMATIC          84   85
    57  0057H    19  WRITECONSOLE . . .  PROCEDURE STACK=000AH           96   97  139  142  144  145  148  149
```

MODULE INFORMATION:

```
     CODE AREA SIZE     = 04EEH   1262D
     CONSTANT AREA SIZE = 0054H     84D
     VARIABLE AREA SIZE = 0085H    181D
     MAXIMUM STACK SIZE = 0016H     22D
     695 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
ISIS-II MCS-86 LOCATER, V1.2 INVOKED BY:
:F0: DIRCMD.LNK OD(SM(CODE,DATA,DATA,STACK,CONST)) AD(SM(CODE(0), DATA(10000H))) SS(STACK(+32)) TO DIRCMD.

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

BASE   OFFSET TYPE SYMBOL          BASE   OFFSET TYPE SYMBOL

0000H  0562H  PUB  PLMSTART       0000H  0102H  PUB  PATCH
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  MAXB           1000H  0050H  PUB  CHDRV
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

DIR:  SYMBOLS AND LINES
1000H  0250H  SYM  MEMORY         0000H  0102H  SYM  PATCH
1000H  0108H  SYM  I              0000H  0157H  SYM  WRITECONSOLE
STACK  0004H  SYM  CHAR           0000H  016AH  SYM  PRINTBUF
STACK  0004H  SYM  BUFFERADDRESS  0000H  017AH  SYM  CHECKCTRLC
0000H  0181H  SYM  SEARCHFIRST    STACK  0004H  SYM  FCBADDRESS
0000H  0191H  SYM  SEARCHNEXT     STACK  0004H  SYM  FCBADDRESS
0000H  01A1H  SYM  SETDMA         STACK  0004H  SYM  DMA
0000H  01B1H  SYM  GETUSERCODE    0000H  01C0H  SYM  SETUSERCODE
STACK  0004H  SYM  USER           0000H  01D3H  SYM  TERMINATE
1000H  010AH  SYM  PARSEFN        1000H  0108H  BAS  DELIMITER
1000H  010EH  SYM  TAILLEN        0000H  01E1H  SYM  PARSE
0000H  01F0H  SYM  CRLF           1000H  0118H  SYM  DIRTITLE
1000H  012FH  SYM  SYS            1000H  0130H  SYM  TEMP
1000H  0131H  SYM  DCNT           1000H  0132H  SYM  CNT
1000H  0133H  SYM  USER           1000H  0134H  SYM  I
1000H  0135H  SYM  NEWUSER        1000H  0136H  SYM  SYSEXISTS
1000H  0137H  SYM  INCLSYS        1000H  0138H  SYM  OPTION
1000H  0139H  SYM  DIRBUF         0000H  0201H  SYM  HEADING
0000H  022DH  SYM  DIRECTORY      0000H  0380H  SYM  PARSEFILE
1000H  0110H  SYM  I              1000H  0108H  BAS  BUF
1000H  0112H  SYM  PARSERET       0000H  0455H  SYM  PARSEOPTIONS
1000H  0114H  SYM  N              1000H  0116H  SYM  I
1000H  0189H  SYM  OPTIONS        1000H  018AH  SYM  INBRACKETS
1000H  018BH  SYM  ERROR          1000H  018CH  SYM  SHOWNNOTHING
0000H  0562H  SYM  PLMSTART       0000H  05E9H  SYM  DONE
0000H  0102H  LIN     37          0000H  0105H  LIN     39
0000H  0110H  LIN     40          0000H  0115H  LIN     41
0000H  011AH  LIN     42          0000H  011FH  LIN     43
0000H  0124H  LIN     44          0000H  0129H  LIN     45
0000H  012EH  LIN     46          0000H  0133H  LIN     47
0000H  0138H  LIN     48          0000H  013DH  LIN     49
0000H  0142H  LIN     50          0000H  0147H  LIN     51
0000H  014CH  LIN     52          0000H  0151H  LIN     53
0000H  0155H  LIN     54          0000H  0157H  LIN     57
0000H  015AH  LIN     59          0000H  0166H  LIN     60
```

```
0000H  016AH  LIN    61      0000H  0169H  LIN    63
0000H  0176H  LIN    64      0000H  017AH  LIN    65
0000H  017DH  LIN    66      0000H  0181H  LIN    67
0000H  0181H  LIN    68      0000H  0184H  LIN    70
0000H  0191H  LIN    71      0000H  0191H  LIN    72
0000H  0194H  LIN    74      0000H  01A1H  LIN    75
0000H  01A1H  LIN    76      0000H  01A4H  LIN    78
0000H  01ADH  LIN    79      0000H  01B1H  LIN    80
0000H  01B4H  LIN    81      0000H  01C0H  LIN    82
0000H  01C0H  LIN    83      0000H  01C3H  LIN    85
0000H  01CFH  LIN    86      0000H  01D3H  LIN    87
0000H  01D6H  LIN    88      0000H  01DFH  LIN    89
0000H  01E1H  LIN    92      0000H  01E4H  LIN    93
0000H  01F0H  LIN    94      0000H  01F0H  LIN    95
0000H  01F3H  LIN    96      0000H  01F9H  LIN    97
0000H  01FFH  LIN    98      0000H  0201H  LIN   103
0000H  0204H  LIN   104      0000H  020BH  LIN   106
0000H  0210H  LIN   107      0000H  0217H  LIN   108
0000H  0217H  LIN   110      0000H  021CH  LIN   111
0000H  0224H  LIN   113      0000H  022BH  LIN   114
0000H  022DH  LIN   115      0000H  0230H  LIN   116
0000H  0235H  LIN   117      0000H  023CH  LIN   119
0000H  023FH  LIN   120      0000H  0244H  LIN   122
0000H  0249H  LIN   123      0000H  024EH  LIN   124
0000H  0255H  LIN   125      0000H  0264H  LIN   126
0000H  0268H  LIN   127      0000H  0277H  LIN   128
0000H  0282H  LIN   129      0000H  0288H  LIN   130
0000H  0296H  LIN   131      0000H  02A0H  LIN   132
0000H  02AAH  LIN   133      0000H  02C0H  LIN   134
0000H  02D4H  LIN   136      0000H  02E9H  LIN   138
0000H  02ECH  LIN   139      0000H  02F5H  LIN   140
0000H  02F5H  LIN   142      0000H  02FBH  LIN   144
0000H  0301H  LIN   145      0000H  0307H  LIN   146
0000H  0313H  LIN   147      0000H  031AH  LIN   148
0000H  0320H  LIN   149      0000H  0334H  LIN   150
0000H  033BH  LIN   151      0000H  033FH  LIN   152
0000H  0345H  LIN   153      0000H  0347H  LIN   154
0000H  034EH  LIN   155      0000H  0353H  LIN   156
0000H  035DH  LIN   157      0000H  0360H  LIN   158
0000H  0367H  LIN   160      0000H  036EH  LIN   162
0000H  0375H  LIN   163      0000H  037CH  LIN   164
0000H  0380H  LIN   165      0000H  0380H  LIN   166
0000H  0383H  LIN   170      0000H  0390H  LIN   172
0000H  039BH  LIN   173      0000H  03A1H  LIN   174
0000H  03C3H  LIN   175      0000H  03E4H  LIN   177
0000H  03ECH  LIN   178      0000H  03EEH  LIN   180
0000H  03EEH  LIN   181      0000H  03F9H  LIN   182
0000H  03FBH  LIN   183      0000H  0401H  LIN   184
0000H  0406H  LIN   186      0000H  040DH  LIN   188
0000H  041DH  LIN   189      0000H  0421H  LIN   190
0000H  0423H  LIN   191      0000H  0423H  LIN   192
0000H  0431H  LIN   194      0000H  043FH  LIN   195
0000H  0443H  LIN   197      0000H  0443H  LIN   199
0000H  044AH  LIN   200      0000H  044FH  LIN   201
0000H  0453H  LIN   203      0000H  0455H  LIN   204
0000H  0458H  LIN   207      0000H  045EH  LIN   208
0000H  0464H  LIN   209      0000H  0469H  LIN   210
0000H  046EH  LIN   211      0000H  047EH  LIN   212
0000H  04A7H  LIN   213      0000H  04AEH  LIN   215
0000H  04B9H  LIN   216      0000H  04C2H  LIN   217
```

```
0000H  04C7H  LIN    218      0000H  04D4H  LIN    219
0000H  04DFH  LIN    221      0000H  04E6H  LIN    222
0000H  04EDH  LIN    223      0000H  04F4H  LIN    225
0000H  04F8H  LIN    226      0000H  0504H  LIN    227
0000H  0508H  LIN    228      0000H  0513H  LIN    229
0000H  051AH  LIN    231      0000H  0526H  LIN    232
0000H  052BH  LIN    234      0000H  052DH  LIN    235
0000H  0532H  LIN    237      0000H  0544H  LIN    238
0000H  0549H  LIN    239      0000H  054CH  LIN    241
0000H  054CH  LIN    242      0000H  0553H  LIN    244
0000H  055AH  LIN    245      0000H  055EH  LIN    247
0000H  0562H  LIN    248      0000H  0562H  LIN    251
0000H  0565H  LIN    252      0000H  056BH  LIN    253
0000H  0578H  LIN    254      0000H  057FH  LIN    255
0000H  0584H  LIN    256      0000H  058CH  LIN    257
0000H  05A5H  LIN    258      0000H  05AEH  LIN    260
0000H  05B3H  LIN    261      0000H  05B9H  LIN    262
0000H  05BFH  LIN    263      0000H  05C6H  LIN    264
0000H  05CFH  LIN    266      0000H  05D1H  LIN    267
0000H  05D8H  LIN    269      0000H  05E6H  LIN    270
0000H  05E9H  LIN    273      0000H  05ECH  LIN    274
0000H  0102H  LIN    275
```



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

SEGMENT MAP

START     STOP      LENGTH ALIGN NAME              CLASS

00000H    005EDH    05EEH   G    CODE              CODE
10000H    10107H    0108H   G    DATS              DATA
10108H    1018CH    0085H   W    DATA              DATA
1018EH    101F3H    0036H   W    STACK             STACK
101F4H    10247H    0054H   W    CONST             CONST
10250H    10250H    0000H   G    ??SEG
10250H    10250H    0000H   W    MEMORY            MEMORY


GROUP MAP

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