```
ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE SDIR
OBJECT MODULE PLACED IN MAIN86
COMPILER INVOKED BY:  :F0: MAIN86.PLM DEBUG OBJECT(MAIN86) OPTIMIZE(3) XREF
```



```
                $title ('SDIR 8086 - Main Module')
  1             sdir:
                do;

                $include (copyrt.lit)
=
=               /*
=                 Copyright (C) 1982
=                 Digital Research
=                 P.O. Box 579
=                 Pacific Grove, CA 93950
=               */
=

                /*      commands used to generate       */
                /*
                asm86 scd.a86
                plm86 main86.plm debug object(main86) optimize(3) 'p2' 'p3' 'p4'
                plm86 scan.plm debug object(scan) optimize(3) 'p2' 'p3' 'p4'
                plm86 search.plm debug object(search) optimize(3) 'p2' 'p3' 'p4'
                plm86 sort.plm debug object(sort) optimize(3) 'p2' 'p3' 'p4'
                plm86 disp.plm debug object(disp) optimize(3) 'p2' 'p3' 'p4'
                plm86 dpb86.plm debug object(dpb86) optimize(3) 'p2' 'p3' 'p4'
                plm86 util.plm debug object(util) optimize(3) 'p2' 'p3' 'p4'
                plm86 timest.plm debug object(timest) optimize(3) 'p2' 'p3' 'p4'
                link86 scd.obj,main86,scan,search,sort,disp,util,dpb86,timest to sdir86.lnk
                loc86 sdir86.lnk od(sm(code,dats,data,const,stack)) -
                   ad(sm(code(0),dats(10000h))) ss(stack(+32))
                h86 sdir86

                   (on a micro)
                    wax sdir86.h86 $fans
                    gencmd sdir86 data(b1000 m3c5 x800)

                    * constants are last to force hex generation.
                    * a minimum data of 3c5h paragraphs is 12K plus the data space
                    * of SDIR, enough for 512 directory entries
                    * the max is lowered from 0fffh to 800h
                        (Aug 12, 1982 for CCP/M-86 IBM PC)
                */

                $include (main.plm)
=
=                 /* C P / M - M P / M    D I R E C T O R Y    C O M M O N    (SDIR)  */
=
=                  /* B E G I N N I N G    O F    C O M M O N    M A I N    M O D U L E */
=
=
=                     /* This module is included in main80.plm or main86.plm. */
=                     /* The differences between 8080 and 8086 versions are   */
=                     /* contained in the modules main80.plm, main86.plm and  */
```

```
        =           /* dpb80.plm, dpb86.plm and the submit files showing    */
        =           /* the different link and location addresses.           */
        =
        =   /* REVISION history:
        =   /*  Nov 82  Bill Fitler: convert from CP/M Plus to Concurrent CP/M-86 */
        =   /*       Feb 83 F.Borda: Took out paging and breaking to allow type-ahead. */
        =
        =   $include (comlit.lit)
        =1
 2   1  =1  declare
        =1          lit                 literally       'literally',
        =1          dcl                 lit             'declare',
        =1          true                lit             '0ffh',
        =1          false               lit             '0',
        =1          boolean             lit             'byte',
        =1          forever             lit             'while true',
        =1          cr                  lit             '13',
        =1          lf                  lit             '10',
        =1          tab                 lit             '9',
        =1          ctrlc               lit             '3',
        =1          ff                  lit             '12',
        =1          page$len$offset     lit             '1ch',
        =1          nopage$mode$offset  lit             '2Ch',
        =1          sectorlen           lit             '128';
        =   $include (mon.plm)
        =1
        =1                  /* definitions for assembly interface module        */
 3   1  =1  declare
        =1      fcb (33) byte external,         /* default file control block   */
        =1      maxb address external,          /* top of memory                */
        =1      buff(128)byte external;         /* default buffer               */
        =1
 4   1  =1  mon1: procedure(f,a) external;
 5   2  =1      declare f byte, a address;
 6   2  =1      end mon1;
        =1
 7   1  =1  mon2: procedure(f,a) byte external;
 8   2  =1      declare f byte, a address;
 9   2  =1      end mon2;
        =1
10   1  =1  mon3: procedure(f,a) address external;
11   2  =1      declare f byte, a address;
12   2  =1      end mon3;
        =1
        =
        =
        =
        =   /* Scanner Entry Points in scan.plm */
        =
13   1  =   scan: procedure(pcb$adr) external;
14   2  =       declare pcb$adr address;
15   2  =   end scan;
        =
16   1  =   scan$init: procedure(pcb$adr) external;
17   2  =       declare pcb$adr address;
18   2  =   end scan$init;
        =
```

```
                =   /* -------- Routines in other modules -------- */
                =
 19    1        =   search$init: procedure external;   /* initialization of search.plm */
 20    2        =   end search$init;
                =
 21    1        =   get$files: procedure external;     /* entry to search.plm */
 22    2        =   end get$files;
                =
 23    1        =   sort: procedure external;          /* entry to sort.plm */
 24    2        =   end sort;
                =
 25    1        =   mult23: procedure (num) address external;    /* in sort.plm */
 26    2        =   dcl num address;
 27    2        =   end mult23;
                =
 28    1        =   display$files: procedure external;    /* entry to disp.plm */
 29    2        =   end display$files;
                =
                =   /* -------- Routines in util.plm -------- */
                =
 30    1        =   printb: procedure external;
 31    2        =   end printb;
                =
 32    1        =   print$char: procedure(c) external;
 33    2        =   dcl c byte;
 34    2        =   end print$char;
                =
 35    1        =   print: procedure(string$adr) external;
 36    2        =   dcl string$adr address;
 37    2        =   end print;
                =
 38    1        =   crlf: procedure external;
 39    2        =   end crlf;
                =
 40    1        =   p$decimal: procedure(value,fieldsize,zsup) external;
 41    2        =       dcl value address,
                =           fieldsize address,
                =           zsup boolean;
 42    2        =   end p$decimal;
                =
                =
                =   /* ------------------------------------ */
                =
 43    1        =   dcl debug boolean public initial (false);
                =
                =   /* -------- version information -------- */
                =
 44    1        =   dcl plmstart label public;
                =
 45    1        =   dcl (os,bdos) byte public;
                =   $include (vers.lit)
 46    1        =1  declare
                =1      bdos20 lit '20h',
                =1      bdos22 lit '22h',
                =1      bdos30 lit '30h',
                =1      mpm    lit '01h',
                =1      cpm86  lit '10h',
```

```
        =1      mpm86  lit '11h',
        =1      ccpm86 lit '14h';
        =
        =   $include (fcb.lit)
        =1
47   1  =1  declare
        =1      f$drvusr        lit '0',        /* drive/user byte              */
        =1      f$name          lit '1',        /* file name                    */
        =1      f$namelen       lit '8',        /* file name length             */
        =1      f$type          lit '9',        /* file type field              */
        =1      f$typelen       lit '3',        /* type length                  */
        =1      f$rw            lit '9',        /* high bit is R/W attribute    */
        =1      f$dirsys        lit '10',       /* high bit is dir/sys attribute */
        =1      f$arc           lit '11',       /* high bit is archive attribute */
        =1      f$ex            lit '12',       /* extent                       */
        =1      f$s1            lit '13',       /* module byte                  */
        =1      f$rc            lit '15',       /* record count                 */
        =1      f$diskmap       lit '16',       /* file disk map                */
        =1      diskmaplen      lit '16',       /* disk map length              */
        =1      f$drvusr2       lit '16',       /* fcb2                         */
        =1      f$name2         lit '17',
        =1      f$type2         lit '25',
        =1      f$rrec          lit '33',       /* random record                */
        =1      f$rreco         lit '35';       /*   "      "    overflow        */
        =1
        =
        =   $include(search.lit)
        =1
48   1  =1  declare                       /* what kind of file user wants to find       */
        =1      find$structure lit 'structure (
        =1      dir byte,
        =1      sys byte,
        =1      ro  byte,
        =1      rw  byte,
        =1      pass byte,
        =1      xfcb byte,
        =1      nonxfcb byte,
        =1      exclude byte)';
        =1
49   1  =1  declare
        =1      max$search$files literally '10';
        =1
50   1  =1  declare
        =1      search$structure lit 'structure(
        =1      drv byte,
        =1      name(8) byte,
        =1      type(3) byte,
        =1      anyfile boolean)';      /* match on any drive if true */
        =1
        =
51   1  =   dcl find find$structure public initial
        =       (false,false,false,false,  false,false,false,false);
        =
52   1  =   dcl
        =       num$search$files byte public initial(0),
        =       no$page$mode byte public initial(0),
        =       search (max$search$files) search$structure public;
```

```
      =
53    1 =   dcl first$f$i$adr address external;
54    1 =   dcl get$all$dir$entries boolean public;
55    1 =   dcl first$pass boolean public;
      =
56    1 =   dcl usr$vector address public initial(0),   /* bits for user #s to scan */
      =         active$usr$vector address public,       /* active users on curdrv   */
      =         drv$vector address initial (0);         /* bits for drives to scan  */
      =
      =   $include (format.lit)
      =1
57    1 =1  dcl form$short lit "0",       /* format values for SDIR */
      =1        form$size lit "1",
      =1        form$full lit "2";
      =1
      =
58    1 =   dcl format byte public initial (form$full),
      =         page$len address public initial (0ffffh),
      =          /* lines on a page before printing new headers, 0 forces initial hdrs  */
      =         message boolean public initial(false),/* show titles when no files found*/
      =         formfeeds boolean public initial(false),/* use form feeds              */
      =         date$opt boolean public initial(false), /* dates display */
      =         display$attributes boolean public initial(false); /* attributes display */
      =
59    1 =   dcl file$displayed boolean external;
      =                             /* true if 1 or more files displayed by dsh.plm    */
      =
60    1 =   dcl sort$op boolean initial (true);              /* default is to do sorting */
61    1 =   dcl sorted boolean external;                     /* if successful sort       */
      =
      =
62    1 =   dcl cur$usr byte public,         /* current user being searched              */
      =         cur$drv byte public;         /* current drive   "     "                  */
      =
      =   /* -------- BDOS calls --------- */
      =
63    1 =   get$version: procedure address; /* returns current version information       */
64    2 =       return mon3(12,0);
65    2 =   end get$version;
      =
66    1 =   select$drive: procedure(d);
67    2 =       declare d byte;
68    2 =       call mon1(14,d);
69    2 =   end select$drive;
      =
70    1 =   search$first: procedure(d) byte external;
71    2 =   dcl d address;
72    2 =   end search$first;
      =
73    1 =   search$next: procedure byte external;
74    2 =   end search$next;
      =
75    1 =   get$cur$drv: procedure byte;          /* return current drive number          */
76    2 =       return mon2(25,0);
77    2 =   end get$cur$drv;
      =
78    1 =   getlogin: procedure address;          /* get the login vector                 */
```

```
 79   2  =       return mon3(24,0);
 80   2  =   end getlogin;
          =
 81   1  =   getusr: procedure byte;            /* return current user number          */
 82   2  =       return mon2(32,0ffh);
 83   2  =   end getusr;
          =
          =   /************************************************** commented out whf
          =       getscbbyte: procedure (offset) byte;
          =         declare offset byte;
          =         declare scbpb structure
          =           (offset byte,
          =            set    byte,
          =            value  address);
          =         scbpb.offset = offset;
          =         scbpb.set = 0;
          =         return mon2(49,.scbpb);
          =       end getscbbyte;
          =   **************************************************************/
          =
 84   1  =   set$console$mode: procedure;
          =     /* set console mode to control-c only */
          =   /*********  call mon1(109,1);  ********whf*************/
 85   2  =     ;
 86   2  =   end set$console$mode;
          =
 87   1  =   terminate: procedure public;
 88   2  =       call mon1 (0,0);
 89   2  =   end terminate;
          =
          =
          =   /* -------- Utility routines -------- */
          =
 90   1  =   number: procedure (char) boolean;
 91   2  =       dcl char byte;
 92   2  =       return(char >= '0' and char <= '9');
 93   2  =   end number;
          =
 94   1  =   make$numeric: procedure(char$adr,len,val$adr) boolean;
 95   2  =       dcl (char$adr, val$adr, place) address,
          =           chars based char$adr (1) byte,
          =           value based val$adr address,
          =           (i,len) byte;
          =
 96   2  =       value = 0;
 97   2  =       place = 1;
 98   2  =       do i = 1 to len;
 99   3  =           if not number(chars(len - i)) then
100   3  =               return(false);
101   3  =           value = value + (chars(len - i) - '0') * place;
102   3  =           place = place * 10;
103   3  =       end;
104   2  =       return(true);
105   2  =   end make$numeric;
          =
106   1  =   set$vec: procedure(v$adr,num) public;
107   2  =       dcl v$adr address,              /* set bit number given by num */
```

```
          =           vector based v$adr address,  /* 0 <= num <= 15              */
          =           num byte;
  108   2 =       if num = 0 then
  109   2 =          vector = vector or 1;
          =       else
  110   2 =          vector = vector or shl(double(1),num);
  111   2 =   end set$vec;
          =
  112   1 =   bit$loc: procedure(vector) byte;
          =                                 /* return location of right most on bit vector */
  113   2 =       dcl vector address,       /* 0 - 15                                      */
          =           i byte;
  114   2 =       i = 0;
  115   2 =       do while i < 16 and (vector and double(1)) = 0;
  116   3 =           vector = shr(vector,1);
  117   3 =           i = i + 1;
  118   3 =       end;
  119   2 =       return(i);
  120   2 =   end bit$loc;
          =
  121   1 =   get$nxt: procedure(vector$adr) byte;
  122   2 =       dcl i byte,
          =           (vector$adr,mask) address,
          =           vector based vector$adr address;
          =       /* if debug then
          =           do;  call print(.(cr,lf,'getnxt: vector = $'));
          =           call pdecimal(vector,10000,false);
          =           end; */
          =
  123   2 =       if (i := bit$loc(vector)) > 15 then
  124   2 =           return(0ffh);
  125   2 =       mask = 1;
  126   2 =       if i > 0 then
  127   2 =           mask = shl(mask,i);
  128   2 =       vector = vector xor mask;                 /* turn off bit        */
          =       /* if debug then
          =           do;  call print(.(cr,lf,'getnxt: vector, i, mask $'));
          =           call pdecimal(vector,10000,false);
          =           call printb;
          =           call pdecimal(i,10000,false);
          =           call printb;
          =           call pdecimal(mask,10000,false);
          =           end; */
  129   2 =       return(i);
  130   2 =   end get$nxt;               /* too bad plm rotates only work on byte values */
          =
          =   /* help: procedure;       COMMENTED OUT - HELP PROGRAM REPLACE DISPLAY
          =
          =   call print(.(cr,lf,
          =   tab,tab,tab,'DIR EXAMPLES',cr,lf,lf,
          =   'dir file.one',tab,tab,tab,
          =   '(find a file on current user and default drive)',cr,lf,
          =   'dir *.com d:*.pli',tab,tab,'(find matching files on default and d: drive)',
          =   cr,lf,
          =   'dir [rw]',tab,tab,tab,'(find files that are read/write)',cr,lf,
          =   'dir [ro dir sys]',tab,tab,'(same for read/only, directory, system)',cr,lf,
          =   'dir [xfcb]',tab,tab,tab,'(find files with XFCB''s)',cr,lf,
```

```
        =   'dir [nonxfcb]',tab,tab,tab,'(find files without XFCB''s)',cr,lf,
        =   'dir [exclude] *.com',tab,tab,'(find files that don''t end in ''com'')',cr,lf,
        =   'dir [nosort]',tab,tab,tab,'(don''t sort the files)',cr,lf,
        =   'dir [full]',tab,tab,tab,'(show all file information)',cr,lf,
        =   'dir [size]',tab,tab,tab,'(show name and size in kilobytes)',cr,lf,
        =   'dir [short]',tab,tab,tab,'(show just the file names)',cr,lf,
        =   'dir [drive = all]',tab,tab,'(search all logged in drives)',cr,lf,
        =   'dir [drive = (a,b,p)]',tab,tab,
        =   '(search specified drives, ''disk'' is synonym)',cr,lf,
        =   'dir [user = all]',tab,tab,'(find files with any user number)',cr,lf,
        =   'dir [user = (0,1,15), G12]',tab,'(find files with specified user number)',
        =   cr,lf,
        =   'dir [length = nn]',tab,tab,'(print headers every n lines)',cr,lf,
        =   'dir [ff]',tab,tab,tab,'(print form feeds between headers)',cr,lf,
        =   'dir [message user=all]',tab,tab,'(show user/drive areas with no files)',
        =   cr,lf,
        =   'dir [help]',tab,tab,tab,'(show this message)',cr,lf,
        =   'dir [dir sys rw ro sort xfcb nonxfcb full] d:*.*',tab,'(defaults)$'));
        =
        =   call terminate;
        =   end help; */
        =
        =
        =   /* -------- Scanner Info -------- */
        =
        =   $include (scan.lit)
        =1
131   1 =1  declare
        =1      pcb$structure literally 'structure (
        =1              state address,
        =1              scan$adr address,
        =1              token$adr address,
        =1              tok$typ byte,
        =1              token$len byte,
        =1              p$level byte,
        =1              nxt$token byte)';
        =1
132   1 =1  declare
        =1      t$null lit '0',
        =1      t$param  lit '1',
        =1      t$op lit '2',
        =1      t$mod lit '4',
        =1      t$identifier lit '8',
        =1      t$string lit '16',
        =1      t$numeric lit '32',
        =1      t$filespec lit '64',
        =1      t$error lit '128';
        =1
        =
133   1 =   dcl pcb pcb$structure
        =       initial (0,.buff(0),.fcb,0,0,0,0) ;
        =
134   1 =   dcl token based pcb.token$adr (12) byte;
135   1 =   dcl got$options boolean;
        =
136   1 =   get$options: procedure;
137   2 =       dcl temp byte;
```

```
                =
138    2        =       do while pcb.scan$adr <> 0ffffh and ((pcb.tok$typ and t$op) <> 0);
139    3        =         if pcb.nxt$token <> t$mod then
140    3        =         do;  /* options with no modifiers */
141    4        =           if token(1) = 'A' then
142    4        =               display$attributes = true;
143    4        =           else if token(1) = ')' and token(2) = 'I' then
144    4        =               find.dir = true;
145    4        =           else if token(1) = 'D' and token(2) = 'A' then do;
147    5        =               format = form$full;
148    5        =               date$opt = true;
149    5        =             end;
                =           /* else if token(1) = 'D' and token(2) = 'E' then
                =               debug = true; */
150    4        =           else if token(1) = 'E' then
151    4        =               find.exclude = true;
152    4        =           else if token(1) = 'F'then
153    4        =               if token(2) = 'F' then
154    4        =                   formfeeds = true;
155    4        =               else if token(2) = 'U' then
156    4        =                   format = form$full;
157    4        =               else goto op$err;
158    4        =           else if token(1) = 'G' then
159    4        =           do;
160    5        =               if pcb.token$len < 3 then
161    5        =                   temp = token(2) - '0';
                =               else
162    5        =                   temp = (token(2) - '0') * 10 + (token(3) - '0');
163    5        =               if temp >= 0 and temp <= 15 then
164    5        =                   call set$vec(.usr$vector,temp);
165    5        =               else goto op$err;
166    5        =           end;
                =           /* else if token(1) = 'H' then
                =               call help; */
167    4        =           else if token(1) = 'M' then
168    4        =               message = true;
169    4        =           else if token(1) = 'N' then
170    4        =               if token(4) = 'X' then
171    4        =                   find.nonxfcb = true;
172    4        =               else if token(3) = 'P' then
173    4        =                   no$page$mode = 0FFh;
174    4        =               else if token(3) = 'S' then
175    4        =                   sort$op = false;
176    4        =               else goto op$err;
                =           /* else if token(1) = 'P' then
                =               find.pass = true; */
177    4        =           else if token(1) = 'R' and token(2) = 'O' then
178    4        =               find.ro = true;
179    4        =           else if token(1) = 'R' and token(2) = 'W' then
180    4        =               find.rw = true;
181    4        =           else if token(1) = 'S' then
182    4        =               if token(2) = 'Y' then
183    4        =                   find.sys = true;
184    4        =               else if token(2) = 'I' then
185    4        =                   format = form$size;
186    4        =               else if token(2) = 'O' then
187    4        =                   sort$op = true;
```

```
188   4  =              else goto op$err;
189   4  =            else if token(1) = 'X' then
190   4  =              find.xfcb = true;
191   4  =            else goto op$err;
192   4  =            call scan(.pcb);
193   4  =          end;
      =          else
194   3  =          do; /* options with modifiers */
195   4  =            if token(1) = 'L' then
196   4  =            do;
197   5  =              call scan(.pcb);
198   5  =              if (pcb.tok$typ and t$numeric) <> 0 then
199   5  =                if make$numeric(.token(1),pcb.token$len,.page$len) then
200   5  =                  if page$len < 5 then
201   5  =                    goto op$err;
202   5  =                  else call scan(.pcb);
203   5  =                else goto op$err;
204   5  =              else goto op$err;
205   5  =            end;
206   4  =            else if token(1) = 'U' then
207   4  =            do;
      =              /* if debug then
      =                  call print(.(cr,lf,'In User option$')); */
208   5  =              call scan(.pcb);
209   5  =              if ((pcb.tok$typ and t$mod) = 0) or bdos < bdos20 then
210   5  =                  goto op$err;
211   5  =              do while (pcb.tok$typ and t$mod) <> 0 and
      =                pcb.scan$adr <> 0ffffh;
212   6  =                if token(1) = 'A' and token(2) = 'L' then
213   6  =                  usr$vector = 0ffffh;
214   6  =                else if (pcb.tok$typ and t$numeric) <> 0 and pcb.token$len < 3 then
215   6  =                  do;
216   7  =                  if pcb.token$len = 1 then
217   7  =                    temp = token(1) - '0';
      =                  else
218   7  =                    temp = (token(1) - '0') * 10 + (token(2) - '0');
219   7  =                  if temp >= 0 and temp <= 15 then
220   7  =                    call set$vec(.usr$vector,temp);
221   7  =                  else goto op$err;
222   7  =                  end;
223   6  =                else goto op$err;
224   6  =                call scan(.pcb);
225   6  =              end;
226   5  =            end;
227   4  =            else if token(1) = 'D' and (token(2) = 'R' or token(2) = 'I') then
228   4  =            do;                          /* allow DRIVE or DISK */
229   5  =              call scan(.pcb);
230   5  =              if (pcb.tok$typ and t$mod) = 0  then
231   5  =                  goto op$err;
232   5  =              do while (pcb.tok$typ and t$mod ) <> 0 and
      =                pcb.scan$adr <> 0ffffh;
233   6  =                if token(1) = 'A' and token(2) = 'L' then
234   6  =                do;
235   7  =                  drv$vector = 0ffffh;
236   7  =                  drv$vector = drv$vector and get$login;
237   7  =                end;
238   6  =                else if token(1) >= 'A' and token(1) <= 'P' then
```

```
239   6  =                  call set$vec(.drv$vector,token(1) - 'A');
240   6  =               else goto op$err;
241   6  =               call scan(.pcb);
242   6  =             end;
243   5  =           end; /* drive option */
244   4  =         else goto op$err;
245   4  =         end; /* options with modifiers */
246   3  =       end; /* do while */
          =
247   2  =       got$options = true;
248   2  =       return;
          =
249   2  =       op$err:
          =           call print(.('ERROR: Illegal Option or Modifier.',
          =                       cr,lf,'$'));
250   2  =           call terminate;
251   2  =   end get$options;
          =
252   1  =   get$file$spec: procedure;
253   2  =       dcl i byte;
254   2  =       if num$search$files < max$search$files then
255   2  =       do;
256   3  =           call move(f$namelen + f$typelen,.token(1),
          =              .search(num$search$files).name(0));
          =
257   3  =           if search(num$search$files).name(f$name - 1) = ' ' and
          =              search(num$search$files).name(f$type - 1) = ' ' then
258   3  =              search(num$search$files).anyfile = true;   /* match on any file */
259   3  =           else search(num$search$files).anyfile = false;/* speedier compare  */
          =
260   3  =           if token(0) = 0 then
261   3  =               search(num$search$files).drv = 0ffh;  /* no drive letter with   */
          =           else                                     /* file spec              */
262   3  =               search(num$search$files).drv = token(0) - 1;
          =           /* 0ffh in drv field indicates to look on all drives that will be   */
          =           /* scanned as set by the "drive =" option, see "match:" proc in     */
          =           /* search.plm module          */
          =
263   3  =           num$search$files = num$search$files + 1;
264   3  =       end;
          =       else
265   2  =       do; call print(.('File Spec Limit is $'));
267   3  =           call p$decimal(max$search$files,100,true);
268   3  =           call crlf;
269   3  =       end;
270   2  =       call scan(.pcb);
271   2  =   end get$file$spec;
          =
272   1  =   set$defaults: procedure;
          =       /* set defaults if not explicitly set by user */
273   2  =       if not (find.dir or find.sys) then
274   2  =           find.dir, find.sys = true;
275   2  =       if not(find.ro or find.rw) then
276   2  =           find.rw, find.ro = true;
          =
277   2  =       if find.xfcb or find.nonxfcb then
278   2  =          do; if format = form$short then
```

```
280   3  =              format = form$full;
281   3  =           end;
      =        else            /* both xfcb and nonxfcb are off */
282   2  =           find.nonxfcb, find.xfcb = true;
      =
283   2  =        if num$search$files = 0 then
284   2  =        do;
285   3  =            search(num$search$files).anyfile = true;
286   3  =            search(num$search$files).drv = 0ffh;
287   3  =            num$search$files = 1;
288   3  =        end;
      =
289   2  =        if drv$vector = 0 then
290   2  =            do i = 0 to num$search$files - 1;
291   3  =                if search(i).drv = 0ffh then search(i).drv = cur$drv;
293   3  =                call set$vec(.drv$vector,search(i).drv);
294   3  =            end;
      =        else                            /* a "Edrive =" option was found */
295   2  =            do i = 0 to num$search$files - 1;
296   3  =                if search(i).drv <> 0ffh and search(i).drv <> cur$drv then
297   3  =                do; call print(.('ERROR: Illegal Global/Local ',
      =                               'Drive Spec Mixing.',cr,lf,'$'));
299   4  =                    call terminate;
300   4  =                end;
301   3  =            end;
302   2  =        if usr$vector = 0 then
303   2  =           call set$vec(.usr$vector,get$usr);
      =
      =        /* set up default page size for display */
      =        /**** page$len = 23;                    /* number lines per screen page */
      =
304   2  =    end set$defaults;
      =
      =
305   1  =    dcl (save$uvec,temp) address;
306   1  =    dcl i byte;
307   1  =    declare last$dseg$byte byte
      =      initial (0);
      =
308   1  =    plmstart:
      =      do;
309   2  =        os = high(get$version);
310   2  =        bdos = low(get$version);
      =
311   2  =        if bdos < bdos22  /* or os <> ccpm86 */
      =           then do;
      =          /*call print(.('Requires Concurrent CP/M-86',cr,lf,'$'));*/
313   3  =          call print(.('Requires BDOS 2.2 or greater.',cr,lf,'$'));
314   3  =          call terminate;   /* check to make sure function call is valid */
315   3  =          end;
      =        else
316   2  =          call set$console$mode;
      =
      =        /* note - initialized declarations set defaults */
317   2  =        cur$drv = get$cur$drv;
318   2  =        call scan$init(.pcb);
319   2  =        call scan(.pcb);
```

```
320   2  =       no$page$mode = false; /******** getscbyte(no$page$mode$offset); ***shf***/
321   2  =       got$options = false;
322   2  =       do while pcb.scan$adr <> 0ffffh;
323   3  =           if (pcb.tok$typ and t$op) <> 0 then
324   3  =               if got$options = false then
325   3  =                   call get$options;
      =               else
326   3  =               do;
327   4  =                   call print(.('ERROR: Options not grouped together.',
      =                               cr,lf,'$'));
328   4  =                   call terminate;
329   4  =               end;
330   3  =           else if (pcb.tok$typ and t$filespec) <> 0 then
331   3  =               call get$file$spec;
      =           else
332   3  =           do;
333   4  =               call print(.('ERROR: Illegal command tail.',cr,lf,'$'));
334   4  =               call terminate;
335   4  =           end;
336   3  =       end;
      =
337   2  =       call set$defaults;
      =
      =       /* main control loop */
      =
338   2  =       call search$init;    /* set up memory pointers for subsequent storage */
      =
339   2  =       do while (cur$drv := get$nxt(.drv$vector)) <> 0ffh;
340   3  =           call select$drive(cur$drv);
341   3  =           save$uvec = usr$vector;       /* user numbers to search on each drive */
342   3  =           active$usr$vector = 0;            /* users active on cur$drv       */
343   3  =           cur$usr = get$nxt(.usr$vector);  /* get first user num and mask   */
344   3  =           get$all$dir$entries = false;     /* off it off                    */
345   3  =           if usr$vector <> 0 and format <> form$short then
      =                                            /* find high water mark if       */
346   3  =           do;                              /* more than one user requested */
347   4  =               fcb(f$drvusr) = '?';
348   4  =               i = search$first(.fcb);       /* get first directory entry    */
349   4  =               temp = 0;
350   4  =               do while i <> 255;
351   5  =                   temp = temp + 1;
352   5  =                   i = search$next;
353   5  =               end;                         /* is there enough space in the */
      =                                            /* worst case ?                  */
354   4  =               if maxb > mult23(temp) + shl(temp,1) then
355   4  =                   get$all$dir$entries = true;  /* location of last possible   */
356   4  =           end;                             /* file info record and add    */
357   3  =           first$pass = true;               /* room for sort indices       */
358   3  =           active$usr$vector = 0ffffh;
359   3  =           do while cur$usr <> 0ffh;
      =               /* if debug then
      =                   call print(.(cr,lf,'in user loop $')); */
360   4  =               call set$vec(.temp,cur$usr);
361   4  =               if (temp and active$usr$vector) <> 0 then
362   4  =               do;
363   5  =                   if format <> form$short and
      =                       (first$pass or not get$all$dir$entries)  then
```

```
364   5  =                    do;
365   6  =                        call getfiles;       /* collect files in memory and  */
366   6  =                        first$pass = false; /* build the active usr vector  */
367   6  =                        sorted = false;      /* sort module will set sorted  */
368   6  =                        if sort$op then      /* to true, if successful sort  */
369   6  =                            call sort;
370   6  =                    end;
371   5  =                    call display$files;
372   5  =                end;
373   4  =                cur$usr = get$nxt(.usr$vector);
374   4  =            end;
375   3  =            usr$vector = save$uvec;              /* restore user vector for nxt  */
376   3  =        end; /* do while drv$usr                    drive scan                   */
               =
               =
377   2  =        if  not file$displayed and not message then
378   2  =            call print(.(cr,lf,cr,lf,'No File',cr,lf,'$'));
379   2  =        call terminate;
               =
380   2  =     end;
381   1  =   end sdir;
```

CROSS-REFERENCE LISTING



| DEFN | ADDR | SIZE | NAME, ATTRIBUTES, AND REFERENCES | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 0000H | 2 | A. . . . . | WORD PARAMETER | 8 | | | | | | | | | |
| 4 | 0000H | 2 | A. . . . . | WORD PARAMETER | 5 | | | | | | | | | |
| 10 | 0000H | 2 | A. . . . . | WORD PARAMETER | 11 | | | | | | | | | |
| 56 | 000AH | 2 | ACTIVEUSRVECTOR. . | WORD PUBLIC | 342 | 358 | 361 | | | | | | | |
| 52 | 000CH | 1 | ANYFILE. . . . . | BYTE MEMBER(SEARCH) | | 258 | 259 | 285 | | | | | | |
| 45 | 0024H | 1 | BDOS . . . . . | BYTE PUBLIC | 209 | 310 | 311 | | | | | | | |
| 46 | | | BDOS20 . . . . . | LITERALLY | 209 | | | | | | | | | |
| 46 | | | BDOS22 . . . . . | LITERALLY | 311 | | | | | | | | | |
| 46 | | | BDOS30 . . . . . | LITERALLY | | | | | | | | | | |
| 112 | 02A6H | 56 | BITLOC . . . . . | PROCEDURE BYTE STACK=0006H | | | | 123 | | | | | | |
| 2 | | | BOOLEAN. . . . . | LITERALLY | 41 | 43 | 52 | 54 | 55 | 58 | 59 | 60 | 61 | 90 | 94 |
| | | | | 135 | | | | | | | | | | | |
| 3 | 0000H | 128 | BUFF . . . . . | BYTE ARRAY(128) EXTERNAL(2) | | | | 133 | | | | | | |
| 32 | 0000H | 1 | C. . . . . | BYTE PARAMETER | 33 | | | | | | | | | |
| 46 | | | CCPM86 . . . . . | LITERALLY | | | | | | | | | | |
| 90 | 0004H | 1 | CHAR . . . . . | BYTE PARAMETER AUTOMATIC | | | | 91 | 92 | | | | | |
| 94 | 0008H | 2 | CHARADR. . . . . | WORD PARAMETER AUTOMATIC | | | | 95 | 99 | 101 | | | | |
| 95 | 0000H | 1 | CHARS. . . . . | BYTE BASED(CHARADR) ARRAY(1) | | | | | 99 | 101 | | | | |
| 46 | | | CPM86. . . . . | LITERALLY | | | | | | | | | | |
| 2 | | | CR . . . . . | LITERALLY | 249 | 298 | 313 | 327 | 333 | 378 | | | | |
| 38 | 0000H | | CRLF . . . . . | PROCEDURE EXTERNAL(16) STACK=0000H | | | | | | 268 | | | | |
| 2 | | | CTRLC. . . . . | LITERALLY | | | | | | | | | | |
| 62 | 0082H | 1 | CURDRV . . . . . | BYTE PUBLIC | 292 | 296 | 317 | 339 | 340 | | | | | |
| 62 | 0081H | 1 | CURUSR . . . . . | BYTE PUBLIC | 343 | 359 | 360 | 373 | | | | | | |
| 70 | 0000H | 2 | D. . . . . | WORD PARAMETER | 71 | | | | | | | | | |
| 66 | 0004H | 1 | D. . . . . | BYTE PARAMETER AUTOMATIC | | | | 67 | 68 | | | | | |
| 58 | 00AEH | 1 | DATEOPT. . . . . | BYTE PUBLIC INITIAL | | 148 | | | | | | | | |
| 2 | | | DCL. . . . . | LITERALLY | | | | | | | | | | |
| 43 | 0022H | 1 | DEBUG. . . . . | BYTE PUBLIC INITIAL | | | | | | | | | | |
| 51 | 0000H | 1 | DIR. . . . . | BYTE MEMBER(FIND) | | 144 | 273 | 274 | | | | | | |
| 47 | | | DISKMAPLEN . . . . | LITERALLY | | | | | | | | | | |
| 58 | 00AFH | 1 | DISPLAYATTRIBUTES. | BYTE PUBLIC INITIAL | | 142 | | | | | | | | |
| 28 | 0000H | | DISPLAYFILES . . . | PROCEDURE EXTERNAL(12) STACK=0000H | | | | | | 371 | | | | |
| | | | DOUBLE . . . . . | BUILTIN | 110 | 115 | | | | | | | | |
| 52 | 0000H | 1 | DRV. . . . . | BYTE MEMBER(SEARCH) | | 261 | 262 | 286 | 291 | 292 | 293 | 296 | | |
| 56 | 000CH | 2 | DRVVECTOR. . . . . | WORD INITIAL | 235 | 236 | 239 | 289 | 293 | 339 | | | | |
| 51 | 0007H | 1 | EXCLUDE. . . . . | BYTE MEMBER(FIND) | | 151 | | | | | | | | |
| 7 | 0000H | 1 | F. . . . . | BYTE PARAMETER | 8 | | | | | | | | | |
| 10 | 0000H | 1 | F. . . . . | BYTE PARAMETER | 11 | | | | | | | | | |
| 4 | 0000H | 1 | F. . . . . | BYTE PARAMETER | 5 | | | | | | | | | |
| 2 | | | FALSE. . . . . | LITERALLY | 43 | 51 | 58 | 100 | 175 | 259 | 320 | 321 | 324 | 344 | 366 |
| | | | | 367 | | | | | | | | | | | |
| 47 | | | FARC . . . . . | LITERALLY | | | | | | | | | | |
| 3 | 0000H | 33 | FCB. . . . . | BYTE ARRAY(33) EXTERNAL(0) | | | | 133 | 347 | 348 | | | | |
| 47 | | | FDIRSYS. . . . . | LITERALLY | | | | | | | | | | |
| 47 | | | FDISKMAP . . . . . | LITERALLY | | | | | | | | | | |
| 47 | | | FDRVUSR. . . . . | LITERALLY | 347 | | | | | | | | | |
| 47 | | | FDRVUSR2 . . . . . | LITERALLY | | | | | | | | | | |
| 47 | | | FEX. . . . . | LITERALLY | | | | | | | | | | |

```
  2                 FF . . . . . .   LITERALLY
 40  0000H     2    FILESIZE. . . .   WORD PARAMETER          41
 59  0000H     1    FILEDISPLAYED. .  BYTE EXTERNAL(19)              377
 51  0000H     8    FIND . . . . . .  STRUCTURE PUBLIC INITIAL          144  151  171  178  180  183  190  273
                                       274  275  276  277  282
 48                 FINDSTRUCTURE. .  LITERALLY          51
 53  0000H     2    FIRSTFADR . . . .  WORD EXTERNAL(18)
 55  00AAH     1    FIRSTPASS. . . .  BYTE PUBLIC              357  363  366
 47                 FNAME. . . . . .  LITERALLY         257
 47                 FNAME2 . . . . .  LITERALLY
 47                 FNAMELEN . . . .  LITERALLY         256
  2                 FOREVER. . . . .  LITERALLY
 58  00ABH     1    FORMAT . . . . .  BYTE PUBLIC INITIAL          147  156  185  279  280  345  363
 58  00ADH     1    FORMFEEDS. . . .  BYTE PUBLIC INITIAL          154
 57                 FORMFULL . . . .  LITERALLY           58  147  156  280
 57                 FORMSHORT. . . .  LITERALLY          279  345  363
 57                 FORMSIZE . . . .  LITERALLY         185
 47                 FRC. . . . . .   LITERALLY
 47                 FRREC. . . . . .  LITERALLY
 47                 FRRECO . . . . .  LITERALLY
 47                 FRW. . . . . .   LITERALLY
 47                 FS1. . . . . .   LITERALLY
 47                 FTYPE. . . . . .  LITERALLY         257
 47                 FTYPE2 . . . . .  LITERALLY
 47                 FTYPELEN . . . .  LITERALLY         256
 54  00A9H     1    GETALLDIRENTRIES  BYTE PUBLIC              344  355  363
 75  01BFH    15    GETCURDRV. . . .  PROCEDURE BYTE STACK=0008H          317
 21  0000H          GETFILES . . . .  PROCEDURE EXTERNAL(9) STACK=0000H          365
252  067EH   178    GETFILESPEC. . .  PROCEDURE STACK=000AH               331
 78  01CEH    15    GETLOGIN . . . .  PROCEDURE WORD STACK=0008H          236
121  02E3H    58    GETNXT . . . . .  PROCEDURE BYTE STACK=000CH          339  343  373
136  031DH   865    GETOPTIONS . . .  PROCEDURE STACK=0014H               325
 81  01DDH    15    GETUSR . . . . .  PROCEDURE BYTE STACK=0008H          303
 63  019DH    15    GETVERSION . . .  PROCEDURE WORD STACK=0008H          309  310
135  00B6H     1    GOTOPTIONS . . .  BYTE        247  321  324
                    HIGH . . . . . .  BUILTIN           309
253  00B8H     1    I. . . . . .     BYTE
113  00B4H     1    I. . . . . .     BYTE        114  115  117  119
 95  00B3H     1    I. . . . . .     BYTE         98   99  101
306  00B9H     1    I. . . . . .     BYTE        290  291  292  293  295  296  348  350  352
122  00B5H     1    I. . . . . .     BYTE        123  126  127  129
307  00BAH     1    LASTDSEGBYTE . .  BYTE INITIAL
 94  0006H     1    LEN. . . . . .   BYTE PARAMETER AUTOMATIC          95   98   99  101
  2                 LF . . . . . .   LITERALLY        249  298  313  327  333  378
  2                 LIT. . . . . .   LITERALLY          2   46   47   48   50   57  132
                    LOW. . . . . .   BUILTIN          310
 94  021CH   108    MAKENUMERIC. . .  PROCEDURE BYTE STACK=0010H          199
122  0012H     2    MASK . . . . . .  WORD        125  127  128
  3  0000H     2    MAXB . . . . . .  WORD EXTERNAL(1)              354
 49                 MAXSEARCHFILES . .  LITERALLY          52  254  267
 58  00ACH     1    MESSAGE. . . . .  BYTE PUBLIC INITIAL          168  377
  4  0000H          MON1 . . . . . .  PROCEDURE EXTERNAL(3) STACK=0000H          68   88
  7  0000H          MON2 . . . . . .  PROCEDURE BYTE EXTERNAL(4) STACK=0000H          76   82
 10  0000H          MON3 . . . . . .  PROCEDURE WORD EXTERNAL(5) STACK=0000H          64   79
                    MOVE . . . . . .  BUILTIN          256
 46                 MPM. . . . . .   LITERALLY
 46                 MPM86. . . . . .  LITERALLY
```

```
  25  0000H        MULT23 . . . . . .  PROCEDURE WORD EXTERNAL(11) STACK=0000H        354
  52  0001H     8  NAME . . . . . .  BYTE ARRAY(8) MEMBER(SEARCH)          256  257
  51  0006H     1  NOMXFCB. . . . . .  BYTE MEMBER(FIND)          171  277  282
  52  0026H     1  NOPAGEMODE . . . .  BYTE PUBLIC INITIAL        173  320
   2              NOPAGEMODEOFFSET .  LITERALLY
 106  0004H     1  NUM. . . . . .  BYTE PARAMETER AUTOMATIC        107  108  110
  25  0000H     2  NUM. . . . . .  WORD PARAMETER          26
  90  01FFH    29  NUMBER . . . . . .  PROCEDURE BYTE STACK=0006H            99
  52  0025H     1  NUMSEARCHFILES . .  BYTE PUBLIC INITIAL        254  256  257  258  259  261  262  263  283
                                       285  286  287  290  295
 135  0009H     1  NXTTOKEN . . . . .  BYTE MEMBER(PCB)          139
 249  0672H        OPERR. . . . . .  LABEL        157  165  176  188  191  201  203  204  210  221  223  231
                                       240  244
  45  0023H     1  OS . . . . . .  BYTE PUBLIC          309
  58  000EH     2  PAGELEN. . . . . .  WORD PUBLIC INITIAL        199  200
   2              PAGELENOFFSET. . .  LITERALLY
  51  0004H     1  PASS . . . . . .  BYTE MEMBER(FIND)
 133  0014H    10  PCB. . . . . .  STRUCTURE INITIAL          134  138  139  141  143  145  150  152  153
                                       155  158  160  161  162  167  169  170  172  174  177  179  181  182
                                       184  186  189  192  195  197  198  199  202  206  208  209  211  212
                                       214  216  217  218  224  227  229  230  232  233  238  239  241  256
                                       260  262  270  318  319  322  323  330
  16  0000H     2  PCBADR . . . . . .  WORD PARAMETER          17
  13  0000H     2  PCBADR . . . . . .  WORD PARAMETER          14
 131              PCBSTRUCTURE . . .  LITERALLY        133
  40  0000H        PDECIMAL . . . . .  PROCEDURE EXTERNAL(17) STACK=0000H        267
  95  0010H     2  PLACE. . . . . .  WORD          97  101  102
 133  0008H     1  PLEVEL . . . . . .  BYTE MEMBER(PCB)
  44  0010H        PLMSTART . . . . .  LABEL PUBLIC          308
  35  0000H        PRINT. . . . . .  PROCEDURE EXTERNAL(15) STACK=0000H        249  266  298  313  327  333
                                       378
  30  0000H        PRINTB . . . . . .  PROCEDURE EXTERNAL(13) STACK=0000H
  32  0000H        PRINTCHAR. . . . .  PROCEDURE EXTERNAL(14) STACK=0000H
  51  0002H     1  RO . . . . . .  BYTE MEMBER(FIND)          178  275  276
  51  0003H     1  RW . . . . . .  BYTE MEMBER(FIND)          180  275  276
 305  001EH     2  SAVEUVEC . . . . .  WORD          341  375
  13  0000H        SCAN . . . . . .  PROCEDURE EXTERNAL(6) STACK=0000H         192  197  202  208  224  229
                                       241  270  319
 133  0002H     2  SCANADR. . . . . .  WORD MEMBER(PCB)          138  211  232  322
  16  0000H        SCANINIT . . . . .  PROCEDURE EXTERNAL(7) STACK=0000H         318
   1  0002H   411  SDIR . . . . . .  PROCEDURE STACK=0016H
  52  0027H   130  SEARCH . . . . . .  STRUCTURE ARRAY(10) PUBLIC          256  257  258  259  261  262  285
                                       286  291  292  293  296
  70  0000H        SEARCHFIRST. . . .  PROCEDURE BYTE EXTERNAL(21) STACK=0000H        348
  19  0000H        SEARCHINIT . . . .  PROCEDURE EXTERNAL(8) STACK=0000H         338
  73  0000H        SEARCHNEXT . . . .  PROCEDURE BYTE EXTERNAL(22) STACK=0000H        352
  50              SEARCHSTRUCTURE. .  LITERALLY         52
   2              SECTORLEN. . . . .  LITERALLY
  66  01ACH    19  SELECTDRIVE. . . .  PROCEDURE STACK=000AH          340
  84  01ECH     5  SETCONSOLEMODE . .  PROCEDURE STACK=0002H          316
 272  0730H   269  SETDEFAULTS. . . .  PROCEDURE STACK=000EH          337
 106  0288H    35  SETVEC . . . . . .  PROCEDURE PUBLIC STACK=0006H        164  220  239  293  303  36
                   SHL. . . . . .  BUILTIN          110  127  354
                   SHR. . . . . .  BUILTIN          116
  23  0000H        SORT . . . . . .  PROCEDURE EXTERNAL(10) STACK=0000H        369
  61  0000H     1  SORTED . . . . . .  BYTE EXTERNAL(20)          367
  60  00B0H     1  SORTOP . . . . . .  BYTE INITIAL          175  187  368
```

```
  133  0000H     2  STATE. . . . . .  WORD MEMBER(PCB)
   35  0000H     2  STRINGADR. . . .  WORD PARAMETER          36
   51  0001H     1  SYS. . . . . .    BYTE MEMBER(FIND)          183  273  274
    2             TAB. . . . . .     LITERALLY
  305  0020H     2  TEMP . . . . . .  WORD         349  351  354  360  361
  137  0087H     1  TEMP . . . . . .  BYTE         161  162  163  164  217  218  219  220
   87  01F1H    14  TERMINATE. . . .  PROCEDURE PUBLIC STACK=0008H        250  299  314  328  334  379
  132             TERROR . . . . .   LITERALLY
  132             TFILESPEC. . . .   LITERALLY          330
  132             TIDENTIFIER. . .   LITERALLY
  132             TMOD . . . . . .   LITERALLY          133  209  211  230  232
  132             TNULL. . . . . .   LITERALLY
  132             TNUMERIC . . . .   LITERALLY          138  214
  134  0000H    12  TOKEN. . . . . .  BYTE BASED(PCB.TOKENADR) ARRAY(12)        141  143  145  150  152  153
                                       155  158  161  162  167  169  170  172  174  177  179  181  182  184
                                       186  189  195  199  206  212  217  218  227  233  238  239  256  260
                                       262
  133  0004H     2  TOKENADR . . . .  WORD MEMBER(PCB)            134  141  143  145  150  152  153  155  158
                                       161  162  167  169  170  172  174  177  179  181  182  184  186  189
                                       195  199  206  212  217  218  227  233  238  239  255  260  262
  133  0007H     1  TOKENLEN . . . .  BYTE MEMBER(PCB)            160  199  214  216
  133  0006H     1  TOKTYP . . . . .  BYTE MEMBER(PCB)            138  198  209  211  214  230  232  323  330
  132             TOP. . . . . .     LITERALLY          138  323
  132             TPARAM . . . . .   LITERALLY
    2             TRUE . . . . . .   LITERALLY           60  104  142  144  148  151  154  168  171  178  180
                                       183  187  190  247  258  267  274  276  282  285  355  357
  132             TSTRING. . . . .   LITERALLY
   52  0009H     3  TYPE . . . . . .  BYTE ARRAY(3) MEMBER(SEARCH)
   56  0008H     2  USRVECTOR. . . .  WORD PUBLIC INITIAL          164  213  220  302  303  341  343  345  373
                                       375
  106  0006H     2  VADR . . . . . .  WORD PARAMETER AUTOMATIC      107  109  110
   94  0004H     2  VALADR . . . . .  WORD PARAMETER AUTOMATIC       95   96  101
   40  0000H     2  VALUE. . . . . .  WORD PARAMETER          41
   95  0000H     2  VALUE. . . . . .  WORD BASED(VALADR)          96  101
  122  0000H     2  VECTOR . . . . .  WORD BASED(VECTORADR)       123  128
  112  0004H     2  VECTOR . . . . .  WORD PARAMETER AUTOMATIC    113  115  116
  107  0000H     2  VECTOR . . . . .  WORD BASED(VADR)          109  110
  121  0004H     2  VECTORADR. . . .  WORD PARAMETER AUTOMATIC    122  123  128
   51  0005H     1  XFCB . . . . . .  BYTE MEMBER(FIND)          190  277  282
   40  0000H     1  ZSUP . . . . . .  BYTE PARAMETER          41
```



MODULE INFORMATION:

```
  CODE AREA SIZE     = 083DH    2109D
  CONSTANT AREA SIZE = 00DEH     222D
  VARIABLE AREA SIZE = 00BBH     187D
  MAXIMUM STACK SIZE = 0016H      22D
  757 LINES READ
  0 PROGRAM ERROR(S)
```

END OF PL/M-86 COMPILATION

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



```
                $title ('Utility Command Line Scanner')
  1             scanner:
                do;

                $include(comlit.lit)
                =
  2   1         =   declare
                =           lit                 literally           'literally',
                =           dcl                 lit                 'declare',
                =           true                lit                 '0ffh',
                =           false               lit                 '0',
                =           boolean             lit                 'byte',
                =           forever             lit                 'while true',
                =           cr                  lit                 '13',
                =           lf                  lit                 '10',
                =           tab                 lit                 '9',
                =           ctrlc               lit                 '3',
                =           ff                  lit                 '12',
                =           page$len$offset     lit                 '1ch',
                =           nopage$mode$offset  lit                 '2Ch',
                =           sectorlen           lit                 '128';
                $include(mon.plm)
                =
                =                   /* definitions for assembly interface module        */
  3   1         =   declare
                =       fcb (33) byte external,         /* default file control block   */
                =       maxb address external,          /* top of memory                */
                =       buff(128)byte external;         /* default buffer               */
                =
  4   1         =   mon1: procedure(f,a) external;
  5   2         =       declare f byte, a address;
  6   2         =       end mon1;
                =
  7   1         =   mon2: procedure(f,a) byte external;
  8   2         =       declare f byte, a address;
  9   2         =       end mon2;
                =
 10   1         =   mon3: procedure(f,a) address external;
 11   2         =       declare f byte, a address;
 12   2         =       end mon3;
                =


 13   1             dcl debug boolean initial (false);

 14   1             dcl eob lit '0';            /* end of buffer */

                $include(fcb.lit)
                =
 15   1         =   declare
                =       f$drvusr            lit '0',        /* drive/user byte          */
                =       f$name              lit '1',        /* file name                */
```

```
=       f$namelen       lit '8',        /* file name length                 */
=       f$type          lit '9',        /* file type field                  */
=       f$typelen       lit '3',        /* type length                      */
=       f$rw            lit '9',        /* high bit is R/W attribute        */
=       f$dirsys        lit '10',       /* high bit is dir/sys attribute    */
=       f$arc           lit '11',       /* high bit is archive attribute    */
=       f$ex            lit '12',       /* extent                           */
=       f$s1            lit '13',       /* module byte                      */
=       f$rc            lit '15',       /* record count                     */
=       f$diskmap       lit '16',       /* file disk map                    */
=       diskmaplen      lit '16',       /* disk map length                  */
=       f$drvusr2       lit '16',       /* fcb2                             */
=       f$name2         lit '17',
=       f$type2         lit '25',
=       f$rrec          lit '33',       /* random record                    */
=       f$rreco         lit '35';       /*   "      "    overflow            */
=

        /* -------- Some routines used for diagnostics if debug mode is on -------- */

16  1   printchar: procedure(char) external;
17  2       declare char byte;
18  2   end printchar;

19  1   printb: procedure external;
20  2   end printb;

21  1   crlf: procedure external;
22  2   end crlf;

23  1   pdecimal: procedure(v,prec,zerosup) external;
                            /* print value v, field size = (log10 prec) + 1  */
                            /* with leading zero suppression if zerosup = true */
24  2       declare v address,                      /* value to print           */
                prec address,                       /* precision                */
                zerosup boolean,                    /* zero suppression flag    */
                d byte;                             /* current decimal digit    */

25  2   end pdecimal;

        /*
        show$buf: procedure;
        dcl i byte;
        i = 1;
        call crlf;
        call mon1(9,.('buff = $'));
        do while buff(i) <> 0;
            i = i + 1;
        end;
        buff(i) = '$';
        call mon1(9,.buff(1));
        buff(i) = 0;
        end show$buf; */


        /* -------- -------- */
```

```
26   1      white$space: procedure (str$adr) byte;
27   2          dcl str$adr address,
                    str based str$adr (1) byte,
                    i byte;
28   2          i = 0;
29   2          do while (str(i) = ' ') or (str(i) = tab);
30   3              i = i + 1;
31   3          end;
32   2          return(i);
33   2      end white$space;

34   1      delimiter: procedure(char) boolean;
35   2          dcl char byte;
36   2          if char = '[' or char = ']' or char = '(' or char = ')' or
                    char = '=' or char = ',' or char = 0 then
37   2              return (true);
38   2          return(false);
39   2      end delimiter;

40   1      dcl string$marker lit '05ch';

41   1      deblank: procedure(buf$adr);
42   2          dcl (buf$adr,dest) address,
                    buf based buf$adr (128) byte,
                    (i,numspaces) byte,
                    string boolean;

43   2          string = false;
44   2          if (numspaces := white$space(.buf(1))) > 0 then
45   2              call move(buf(0) - numspaces + 1,.buf(numspaces+1),.buf(1));
46   2          i = 1;
47   2          do while buf(i) <> 0;

            /*      call show$buf;*/

48   3              do while ((numspaces := white$space(.buf(i))) = 0 and (buf(i) <> 0))
                        and not string;
                   /*   call mon1(9,.(cr,lf,'2numspaces = $'));
                        call pdecimal(numspaces,100,false);*/
            /*          call show$buf;*/
49   4                  if buf(i) = '"' then
50   4                  do;
51   5                      string = true;
52   5                      buf(i) = string$marker;
53   5                  end;
54   4                  i = i + 1;
55   4              end;

56   3              do while string and buf(i) <> 0;
57   4                  if buf(i) = '"' then
58   4                      if buf(i+1) = '"' then
59   4                          call move(buf(0) - i + 1,.buf(i+1), .buf(i));
                            else
60   4                      do;
61   5                          buf(i) = string$marker;
62   5                          string = false;
```

```
 63   5                    end;
 64   4                i = i + 1;
 65   4            end;

 66   3            if (numspaces := white$space(.buf(i))) > 0 then
 67   3            do;
          /*              call mon1(9,.(cr,lf,'1numspaces = $'));
                      call pdecimal(numspaces,100,false);*/
 68   4                buf(i) = ' ';
 69   4                dest = .buf(i+1);                    /* save space for ','    */
 70   4                if i > 1 then
 71   4                    if delimiter(buf(i-1)) or delimiter(buf(i+numspaces)) then
                                                           /* write over ' ' with   */
 72   4                        dest = dest - 1;             /* a = [ ] ( )           */

 73   4                call move(((buf(0)+1)-(i+numspaces-1)),
                          .buf(i+numspaces),dest);
 74   4                if buf(i) = '"' then
 75   4                    string = true;
 76   4                i = i + 1;
 77   4            end;

 78   3        end;
 79   2        if buf(i - 1) = ' ' then      /* no trailing blanks */
 80   2            buf(i - 1) = 0;
              /* if debug then
                  call show$buf; */
 81   2    end deblank;

 82   1    upper$case: procedure (buf$adr);
 83   2         dcl buf$adr address,
                  buf based buf$adr (1) byte,
                  i byte;

 84   2         i = 0;
 85   2         do while buf(i) <> eob;
 86   3             if buf(i) >= 'a' and buf(i) <= 'z' then
 87   3                 buf(i) = buf(i) - ('a' - 'A');
 88   3             i = i + 1;
 89   3         end;
 90   2    end upper$case;

 91   1    dcl option$max lit '11';
 92   1    dcl done$scan lit '0ffffh';
 93   1    dcl ident$max lit '11';
 94   1    dcl token$max lit '11';

 95   1    dcl t$null lit '0',
              t$param  lit '1',
              t$option lit '2',
              t$modifier lit '4',
              t$identifier lit '8',
              t$string lit '16',
              t$numeric lit '32',
              t$filespec lit '64',
              t$error lit '128';
```

```
 96   1      dcl pcb$base address;
 97   1      dcl pcb based pcb$base structure (
                    state address,
                    scan$adr address,
                    token$adr address,
                    token$type byte,
                    token$len byte,
                    p$level byte,
                    nxt$token byte);

 98   1      dcl     scan$adr address,
                    inbuf based scan$adr (1) byte,
                    in$ptr byte,
                    token$adr address,
                    token based token$adr (1) byte,
                    t$ptr byte,
                    (char, nxtchar, tcount) byte;

 99   1      digit: procedure (char) boolean;
100   2          dcl char byte;
101   2          return (char >= '0' and char <= '9');
102   2      end digit;

103   1      letter: procedure (char) boolean;
104   2          dcl char byte;
105   2          return (char >= 'A' and char <= 'Z');
106   2      end letter;

107   1          eat$char: procedure;
108   2              char = inbuf(in$ptr := inptr + 1);
109   2              nxtchar = inbuf(in$ptr + 1);
110   2          end eat$char;

111   1          put$char: procedure(charx);
112   2              dcl charx byte;
113   2              if pcb.token$adr <> 0ffffh then
114   2                  token(t$ptr := t$ptr + 1) = charx;
115   2          end put$char;

116   1          get$identifier: procedure (max) byte;
117   2              dcl max byte;

118   2              tcount = 0;
                 /* call mon1(9,.(cr,lf,'getindentifier$'));*/
119   2              if not letter(char) and char <> '$' then
120   2                  return(tcount);
121   2              do while (letter(char) or digit(char) or char = '_' or
                        char = '$' ) and tcount <= max;
122   3                  call put$char(char);
123   3                  call eat$char;
124   3                  tcount = tcount + 1;
125   3              end;
126   2              do while letter(char) or digit(char) or char = '_'
                        or char = '$' ;
127   3                  call eat$char;
128   3                  tcount = tcount + 1;
129   3              end;
```

```
130   2             pcb.token$type = t$identifier;
              /*     call mon1(9,.(cr,lf,'end of getident$'));  */
131   2             pcb.token$len = tcount;
132   2             return(tcount);
133   2         end get$identifier;

134   1         file$char: procedure (x) boolean;
135   2             dcl x byte;
136   2             return(letter(x) or digit(x) or x = '*' or x = '?'
                        or x = '_' or x = '$');
137   2         end file$char;

138   1         expand$wild$cards: procedure(field$size) boolean;
139   2             dcl (i,leftover,field$size) byte,
                      save$inptr address;

140   2             field$size = field$size + t$ptr;
141   2             do while filechar(char) and t$ptr < field$size;
142   3                if char = '*' then
143   3                do; leftover = t$ptr;
145   4                    save$inptr = inptr;
146   4                    call eatchar;
147   4                    do while filechar(char);
148   5                        leftover = leftover + 1;
149   5                        call eatchar;
150   5                    end;
151   4                    if leftover >= field$size then   /* too many chars */
152   4                    do; inptr = save$inptr;
154   5                        return(false);
155   5                    end;
156   4                    do i = 1 to field$size - leftover;
157   5                        call putchar('?');
158   5                    end;
159   4                    inptr = save$inptr;
160   4                end;
                         else
161   3                    call putchar(char);
162   3                call eatchar;
163   3             end;
164   2             return(true);
165   2         end expand$wild$cards;

166   1         get$file$spec: procedure boolean;
167   2             dcl i byte;
168   2             do i = 1 to f$name$len + f$type$len;
169   3                token(i) = ' ';
170   3             end;
171   2             if nxtchar = ':' then
172   2                if char >= 'A' and char <= 'P' then
173   2                do;
174   3                    call putchar(char - 'A' + 1);
175   3                    call eat$char;        /* skip ':'              */
176   3                    call eat$char;        /* 1st char of file name */
177   3                end;
                         else
178   2                    return(false);
                     else
```

```
179  2              call putchar(0);                    /* use default drive */

180  2          if not (letter(char) or char = '$' or char = '_'
                    or char = '*' or char = '?' ) then  /* no leading numerics */
181  2              if token(0) = 0 then        /* ambiguous with numeric token */
182  2                  return(false);

183  2          if not expand$wild$cards(f$namelen) then
184  2              return(false);                      /* blank name is illegal */
185  2          if char = '.' then
186  2              do; call eat$char;
188  3              if filechar(char) then
189  3                  do; t$ptr = f$namelen;
191  4                  if not expand$wild$cards(f$typelen) then
192  4                      return(false);
193  4                  end;
194  3              end;

195  2          pcb.token$len = f$name$len + f$type$len + 1;
196  2          pcb.token$type = t$file$spec;
197  2          return(true);
198  2      end get$file$spec;

199  1      get$numeric: procedure(max) boolean;
200  2          dcl max byte;
201  2          if not digit(char) then
202  2               return(false);
203  2          do while digit(char) and pcb.token$len <= max and
                    char <> eob;
204  3              call putchar(char);
205  3              call eat$char;
206  3              pcb.token$len = pcb.token$len + 1;
207  3          end;
208  2          if char = 'H' or char = 'D' or char = 'B' then
209  2              if pcb.token$len < max then
210  2              do;
211  3                   call putchar(char);
212  3                   call eat$char;
213  3                   pcb.token$len = pcb.token$len + 1;
214  3              end;
                    else
215  2                   return(false);
216  2          pcb.token$type = t$numeric;
217  2          return(true);
218  2      end get$numeric;

219  1      get$string: procedure(max) boolean;
220  2          dcl max byte;
221  2          if char <> string$marker then
222  2              return(false);
223  2          call eatchar;
224  2          do while char <> string$marker and char <> eob
                    and pcb.token$len < token$max;
225  3              call putchar(char);
226  3              call eatchar;
227  3              pcb.token$len = pcb.token$len + 1;
228  3          end;
```

```
229   2             do while char <> string$marker and char <> eob;
230   3                 call eat$char;
231   3             end;
232   2             if char <> string$marker then
233   2                 return(false);
234   2             pcb.token$type = t$string;
235   2             call eat$char;
236   2             return(true);
237   2         end get$string;

238   1         get$token$all: procedure boolean;
239   2             dcl save$inptr byte;

              /*      call mon1(9,.(cr,lf,'gettokenall$'));*/

240   2             save$inptr = in$ptr;
241   2         if get$file$spec then
242   2                 return(true);

              /*        call mon1(9,.(cr,lf,'gettokenall - no file$')); */
243   2             in$ptr = save$inptr - 1; /* need to re-scan, reset buffer pointers */
244   2             call eat$char;
245   2             t$ptr = 255;
246   2             call putchar(0);              /* zero drive byte */

247   2             if get$identifier(token$max) = 0 then
248   2                 if not get$string(token$max) then
249   2                     if not get$numeric(token$max) then
250   2                         return(false);
             /*       call mon1(9,.(cr,lf,'end gettokenall$'));*/
251   2             return(true);
252   2         end get$token$all;

253   1         get$modifier: procedure boolean;
254   2             if char = ',' or char = ')' or char = 0 then
255   2             do;
256   3                 pcb.token$type = t$modifier or t$null;
257   3                 return(true);
258   3             end;
259   2             if get$token$all then
260   2             do;
261   3                 pcb.token$type = pcb.token$type or t$modifier;
262   3                 return(true);
263   3             end;
264   2             return(false);
265   2         end get$modifier;

266   1         get$option: procedure boolean;
267   2             call putchar(0);
268   2             if get$identifier(token$max) > 0 then
269   2             do;
270   3                 pcb.token$type = pcb.token$type or t$option;
271   3                 if pcb.token$len > token$max then
272   3                     pcb.token$len = token$max;
273   3                 return(true);
274   3             end;
```

```
275   2             return(false);
276   2         end get$option;

277   1          get$param: procedure boolean;
278   2              if char = ',' or char = ')' or char = 0 then
279   2              do;
280   3                  pcb.token$type = t$param or t$null;
281   3                  return(true);
282   3              end;
283   2              if get$token$all then
284   2              do;
285   3                  pcb.token$type = pcb.token$type or t$param;
286   3                  return(true);
287   3              end;
288   2              return(false);
289   2          end get$param;

290   1          dcl gotatoken boolean;
291   1          dcl parens byte initial (0);

292   1          end$state: procedure boolean;
293   2              if gotatoken then
294   2              do;
295   3                  pcb.state = .end$state;
296   3                  return(true);
297   3              end;
298   2              pcb.token$type = t$null;
299   2              pcb.scan$adr = 0ffffh;
300   2              return(true);
301   2          end end$state;

302   1          state8: procedure boolean reentrant;
303   2              if debug then do;
305   3              call mon1(9,.(cr,lf,'state8, char = $'));
306   3              call printchar(char); end;
308   2              if char = 0 then
309   2                  return(end$state);
310   2              if char = ']' then
311   2              do;
312   3                  call eatchar;
313   3                  if char = ',' or nxtchar = '(' or nxtchar = ')' then
314   3                      return(state2);
315   3                  else if char = 0 then
316   3                      return(end$state);
                        else
317   3                      return(state1);
318   3              end;
319   2              else if char = ' ' or char = ',' then
320   2              do;
321   3                  call eatchar;
322   3                  return(state3);
323   3              end;
                    return(state3);
325   2          end state8;

326   1          state7:procedure boolean reentrant;
327   2              if debug then do;
```

```
329   3             call mon1(9,.(cr,lf,'state7, char = $'));
330   3             call printchar(char); end;
332   2             if char = 0 then
333   2                 return(end$state);
334   2             if char = ' ' or char = ',' then
335   2             do;
336   3                 call eat$char;
337   3                 return(state6);
338   3             end;
              
              else
339   2                 if char = ')' then
340   2                 do;
341   3                    call eat$char;
342   3                    return(state8);
343   3                 end;
              return(false);
345   2         end state7;

346   1         state6: procedure boolean reentrant;
347   2             if debug then do;
349   3             call mon1(9,.(cr,lf,'state6, char = $'));
350   3             call printchar(char); end;
352   2             if gotatoken then
353   2             do;
354   3                 pcb.state = .state6;
355   3                 pcb.nxt$token = t$modifier;
356   3                 return(true);
357   3             end;
358   2             if (gotatoken := get$modifier) then
359   2                 return(state7);
360   2             return(false);
361   2         end state6;

362   1         state5:procedure boolean reentrant;
363   2             if debug then do;
365   3             call mon1(9,.(cr,lf,'state5, nxtchar = $'));
366   3             call printchar(nxtchar); end;
368   2             if char = '(' then
369   2             do;
370   3                 call eat$char;
371   3                 return(state6);
372   3             end;
373   2             if gotatoken then
374   2             do;
375   3                 pcb.state = .state5;
376   3                 pcb.nxt$token = t$modifier;
377   3                 return(true);
378   3             end;
379   2             if (gotatoken := get$modifier) then
380   2                     return(state8);
381   2             return(false);
382   2         end state5;

383   1         state4: procedure boolean reentrant;
384   2             dcl temp byte;
385   2             if debug then do;
387   3             call mon1(9,.(cr,lf,'state4, char = $'));
```

```
388   3             call printchar(char); end;
390   2             if char = 0 then
391   2                 return(end$state);
392   2             temp = char;
393   2             call eatchar;
394   2             if temp = ',' or temp = ' ' then
395   2                 return(state3);
396   2             if temp = ']' then
397   2                 if char = '(' or char = ',' or char = ')' then
398   2                     return(state2);
399   2                 else if char = 0 then
400   2                     return(end$state);
                      else
401   2                     return(state1);
402   2             if temp = '=' then
403   2                 return(state5);
404   2             return(false);
405   2         end state4;

406   1         state3: procedure boolean reentrant;
407   2            if debug then do;
409   3            call mon1(9,.(cr,lf,'state3, char = $'));
410   3            call printchar(char); end;
412   2            if gotatoken then
413   2            do;
414   3                pcb.state = .state3;
415   3                pcb.nxt$token = t$option;
416   3                return(true);
417   3            end;
418   2            if (pcb.plevel := parens ) > 128 then
419   2                 return(false);
420   2            if (gotatoken := get$option) then
421   2                 return(state4);
422   2            return(false);
423   2         end state3;

424   1         state2: procedure boolean reentrant;
425   2             if debug then do;
427   3             call mon1(9,.(cr,lf,'state2, char = $'));
428   3             call printchar(char); end;
430   2             do while char = ')' or char = 0;
431   3                 if char = 0 then
432   3                     return(end$state);
433   3                 call eat$char;
434   3                 parens = parens - 1;
435   3             end;
436   2             if char = '[' then
437   2             do;
438   3                 call eat$char;
439   3                 return(state3);
440   3             end;
441   2             if char = ' ' or char = ',' or char = '(' then
442   2             do;
443   3                 if char = '(' then
444   3                     parens = parens + 1;
445   3                 call eat$char;
446   3                 return(state1);
```

```
447   3               end;
448   2               return(state1);
449   2           end state$2;

450   1           state1: procedure boolean reentrant;
451   2               if debug then do;
453   3               call mon1(9,.(cr,lf,'state1, char = $'));
454   3               call printchar(char); end;

456   2               if gotatoken then
457   2               do;
458   3                   pcb.nxt$token = t$param;
459   3                   pcb.state = .state1;
460   3                   return(true);
461   3               end;
462   2               do while char = '(' ;
463   3                   parens = parens + 1;
464   3                   call eat$char;
465   3               end;
466   2               if (pcb.plevel := parens) > 128 then
467   2                   return(false);
468   2               if (gotatoken := get$param) then
469   2                   return(state2);
470   2               return(false);
471   2            end state1;

472   1           start$state: procedure boolean;
473   2               if char = '@' then do;
475   3                  debug = true;
476   3                  call eat$char;
477   3                  call mon1(9,.(cr,lf,'startstate, char = $'));
478   3                  call printchar(char); end;

480   2               if char = 0 then
481   2                   return(end$state);
482   2               if char = ')' then
483   2                   return(false);
484   2               if char = '(' then
485   2               do;
486   3                  parens = parens + 1;
487   3                  call eat$char;
488   3                  return(state1);
489   3               end;
490   2               if char = '[' then
491   2               do;
492   3                   call eat$char;
493   3                   return(state3);
494   3               end;
495   2               if (gotatoken := get$param) then
496   2                   return(state2);
497   2               return(false);
498   2           end start$state;

            /* display$all: procedure;     /* called if debug set */

               /* call mon1(9,.(cr,lf,'scanadr=$'));
               call pdecimal(pcb.scanadr,10000,false);
```

```
        call mon1(9,.(', tadr=$'));
        call pdecimal(pcb.token$adr,10000, false);
        call mon1(9,.(', tlen=$'));
        call pdecimal(double(pcb.token$len),100, false);
        call mon1(9,.(', ttype=$'));
        call pdecimal(double(pcb.token$type),100,false);
        call mon1(9,.(', plevel=$'));
        call pdecimal(double(pcb.plevel),100,false);
        call mon1(9,.(', ntok=$'));
        call pdecimal(double(pcb.nxt$token),100,false);

        if (pcb.token$type and t$option) <> 0 then
            call mon1(9,.(cr,lf,'option  =$'));
        if (pcb.token$type and t$param) <> 0 then
            call mon1(9,.(cr,lf,'parm    =$'));
        if (pcb.token$type and t$modifier) <> 0 then
            call mon1(9,.(cr,lf,'modifier=$'));

        if (pcb.token$type and t$filespec) <> 0 then
        do;
            if fcb(0) = 0 then
                call print$char('0');
            else call print$char(fcb(0) + 'A' - 1);
            call print$char(':');
            fcb(12) = '$';
            call mon1(9,.fcb(1));
            call mon1(9,.('         (filespec)$'));
        end;
        if ((pcb.token$type and t$string) or (pcb.token$type and
            t$identifier) or (pcb.token$type and t$numeric)) <> 0 then
        do;
            fcb(pcb.token$len + 1) = '$';
            call mon1(9,.fcb(1));
        end;
        if pcb.token$type = t$error then
        do;
            call mon1(9,.(cr,lf,'scanner error$'));
            return;
        end;

        if (pcb.token$type and t$identifier) <> 0 then
             call mon1(9,.(' (identifier)$'));
        if (pcb.token$type and t$string) <> 0 then
             call mon1(9,.(' (string)$'));
        if (pcb.token$type and t$numeric) <> 0 then
             call mon1(9,.(' (numeric)$'));

        if (pcb.nxt$token and t$option) <> 0 then
            call mon1(9,.(cr,lf,'nxt tok = option  $'));
        if (pcb.nxt$token and t$param) <> 0 then
            call mon1(9,.(cr,lf,'nxt tok = parm    $'));
        if (pcb.nxt$token and t$modifier) <> 0 then
            call mon1(9,.(cr,lf,'nxt tok = modifier$'));
        call crlf;

    end display$all; */
```

```
499  1      scan: procedure (pcb$adr) public;

500  2          dcl status boolean,
                   pcb$adr address;

501  2          pcb$base = pcb$adr;
502  2          scan$adr = pcb.scan$adr;
503  2          token$adr = pcb.token$adr;

504  2          in$ptr, t$ptr = 255;
505  2          call eatchar;

506  2          gotatoken = false;
507  2          pcb.nxt$token = t$null;
508  2          pcb.token$len = 0;

509  2          if pcb.token$type = t$error then          /* after one error, return */
510  2              return;                               /* on any following calls  */
511  2          else if pcb.state = .start$state then
512  2              status = start$state;
513  2          else if pcb.state = .state$1 then
514  2              status = state$1;
515  2          else if pcb.state = .state$3 then
516  2              status = state$3;
517  2          else if pcb.state = .state$5 then
518  2              status = state$5;
519  2          else if pcb.state = .state$6 then
520  2              status = state$6;
521  2          else if pcb.state = .end$state then       /* repeated calls go here   */
522  2              status = end$state;                   /* after first end$state    */
                else
523  2              status = false;

524  2          if not status then
525  2              pcb.token$type = t$error;

526  2          if pcb.scan$adr <> 0ffffh then
527  2              pcb.scan$adr = pcb.scan$adr + inptr;
                /* if debug then
                    call display$all; */
528  2      end scan;

529  1      scan$init: procedure(pcb$adr) public;
530  2          dcl pcb$adr address;

531  2          pcb$base = pcb$adr;
532  2          call deblank(pcb.scan$adr);
533  2          call upper$case(pcb.scan$adr := pcb.scan$adr + 1);
534  2          pcb.state = .start$state;
535  2      end scan$init;

536  1      end scanner;
```

CROSS-REFERENCE LISTING



| DEFN | ADDR | SIZE | NAME, ATTRIBUTES, AND REFERENCES |
|---|---|---|---|
| 7 | 0000H | 2 | A. . . . . WORD PARAMETER 8 |
| 4 | 0000H | 2 | A. . . . . WORD PARAMETER 5 |
| 10 | 0000H | 2 | A. . . . . WORD PARAMETER 11 |
| 2 | | | BOOLEAN. . . . . LITERALLY 13 24 34 42 99 103 134 138 166 199 219 238 253 266 277 290 292 302 326 346 362 383 406 424 450 472 500 |
| 83 | 0000H | 1 | BUF. . . . . BYTE BASED(BUFADR) ARRAY(1) 85 86 87 |
| 42 | 0000H | 128 | BUF. . . . . BYTE BASED(BUFADR) ARRAY(128) 44 45 47 48 49 52 56 57 58 59 61 66 68 69 71 73 74 79 80 |
| 82 | 0004H | 2 | BUFADR . . . . . WORD PARAMETER AUTOMATIC 83 85 86 87 |
| 41 | 0004H | 2 | BUFADR . . . . . WORD PARAMETER AUTOMATIC 42 44 45 47 48 49 52 56 57 58 59 61 66 68 69 71 73 74 79 80 |
| 3 | 0000H | 128 | BUFF . . . . . BYTE ARRAY(128) EXTERNAL(2) |
| 34 | 0004H | 1 | CHAR . . . . . BYTE PARAMETER AUTOMATIC 35 36 |
| 98 | 0013H | 1 | CHAR . . . . . BYTE 108 119 121 122 126 141 142 147 161 172 174 180 185 188 201 203 204 208 211 221 224 225 229 232 254 278 306 308 310 313 315 319 330 332 334 339 350 368 388 390 392 397 399 410 428 430 431 436 441 443 454 462 473 478 480 482 484 490 |
| 16 | 0000H | 1 | CHAR . . . . . BYTE PARAMETER 17 |
| 99 | 0004H | 1 | CHAR . . . . . BYTE PARAMETER AUTOMATIC 100 101 |
| 103 | 0004H | 1 | CHAR . . . . . BYTE PARAMETER AUTOMATIC 104 105 |
| 111 | 0004H | 1 | CHARX. . . . . BYTE PARAMETER AUTOMATIC 112 114 |
| 2 | | | CR . . . . . LITERALLY 305 329 349 365 387 409 427 453 477 |
| 21 | 0000H | | CRLF . . . . . PROCEDURE EXTERNAL(8) STACK=0000H |
| 2 | | | CTRLC. . . . . LITERALLY |
| 24 | 000BH | 1 | D. . . . . BYTE |
| 2 | | | DCL. . . . . LITERALLY |
| 41 | 0066H | 427 | DEBLANK. . . . . PROCEDURE STACK=000CH 532 |
| 13 | 000AH | 1 | DEBUG. . . . . BYTE INITIAL 303 327 347 363 385 407 425 451 475 |
| 34 | 003AH | 44 | DELIMITER. . . . . PROCEDURE BYTE STACK=0004H 71 |
| 42 | 0000H | 2 | DEST . . . . . WORD 69 72 73 |
| 99 | 0240H | 29 | DIGIT. . . . . PROCEDURE BYTE STACK=0006H 121 126 136 201 203 |
| 15 | | | DISKMAPLEN . . . . LITERALLY |
| 92 | | | DONESCAN . . . . . LITERALLY |
| 107 | 027AH | 33 | EATCHAR. . . . . PROCEDURE STACK=0002H 123 127 146 149 162 175 176 187 205 212 223 226 230 235 244 312 321 336 341 370 393 433 438 445 464 476 487 492 505 |
| 292 | 0747H | 37 | ENDSTATE . . . . . PROCEDURE BYTE STACK=0002H 295 309 316 333 391 400 432 481 521 522 |
| 14 | | | EOB. . . . . LITERALLY 85 203 224 229 |
| 138 | 03D5H | 152 | EXPANDWILDCARDS. . PROCEDURE BYTE STACK=0014H 183 191 |
| 4 | 0000H | 1 | F. . . . . BYTE PARAMETER 5 |
| 7 | 0000H | 1 | F. . . . . BYTE PARAMETER 8 |
| 10 | 0000H | 1 | F. . . . . BYTE PARAMETER 11 |
| 2 | | | FALSE. . . . . LITERALLY 13 38 43 62 154 178 182 184 192 202 215 222 233 250 264 275 288 344 360 381 404 419 422 467 470 483 497 506 523 |
| 15 | | | FARC . . . . . LITERALLY |

```
  3  0000H    33  FCB. . . . . .   BYTE ARRAY(33) EXTERNAL(0)
 15              FDIRSYS. . . . .   LITERALLY
 15              FDISKMAP . . . .   LITERALLY
 15              FORVUSR. . . . .   LITERALLY
 15              FORVUSR2 . . . .   LITERALLY
 15              FEX. . . . . .     LITERALLY
  2              FF . . . . . .     LITERALLY
138  0004H     1  FIELDSIZE. . . .   BYTE PARAMETER AUTOMATIC        139  140  141  151  156
134  0389H    76  FILECHAR . . . .   PROCEDURE BYTE STACK=000EH           141  147  188
 15              FNAME. . . . . .   LITERALLY
 15              FNAME2 . . . . .   LITERALLY
 15              FNAMELEN . . . .   LITERALLY        168  183  190  195
  2              FOREVER. . . . .   LITERALLY
 15              FRC. . . . . .     LITERALLY
 15              FRREC. . . . . .   LITERALLY
 15              FRRECO . . . . .   LITERALLY
 15              FRW. . . . . .     LITERALLY
 15              FS1. . . . . .     LITERALLY
 15              FTYPE. . . . . .   LITERALLY
 15              FTYPE2 . . . . .   LITERALLY
 15              FTYPELEN . . . .   LITERALLY        168  191  195
166  046DH   222  GETFILESPEC. . .   PROCEDURE BYTE STACK=0018H           241
116  02C1H   200  GETIDENTIFIER. .   PROCEDURE BYTE STACK=000EH           247  268
253  06B4H    51  GETMODIFIER. . .   PROCEDURE BYTE STACK=0020H           358  379
199  054BH   143  GETNUMERIC . . .   PROCEDURE BYTE STACK=000CH           249
266  06E7H    45  GETOPTION. . . .   PROCEDURE BYTE STACK=0012H           420
277  0714H    51  GETPARAM . . . .   PROCEDURE BYTE STACK=0020H           468  495
219  05DAH   138  GETSTRING. . . .   PROCEDURE BYTE STACK=000AH           248
238  0664H    80  GETTOKENALL. . .   PROCEDURE BYTE STACK=001CH           259  283
290  001AH     1  GOTATOKEN. . . .   BYTE        293  352  358  373  379  412  420  456  468  495  506
 27  000CH     1  I. . . . . .       BYTE         28   29   30   32
167  0018H     1  I. . . . . .       BYTE        163  169
139  0016H     1  I. . . . . .       BYTE        156
 83  0010H     1  I. . . . . .       BYTE         84   85   86   87   88
 42  0000H     1  I. . . . . .       BYTE         46   47   48   49   52   54   56   57   58   59   61   64
                                       66   68   69   70   71   73   74   76   79   80
 93              IDENTMAX . . . .   LITERALLY
 98  0000H     1  INBUF. . . . . .   BYTE BASED(SCANADR) ARRAY(1)        108  109
 98  0011H     1  INPTR. . . . . .   BYTE        108  109  145  153  159  240  243  504  527
139  0017H     1  LEFTOVER . . . .   BYTE        144  148  151  156
103  025DH    29  LETTER . . . . .   PROCEDURE BYTE STACK=0006H           119  121  126  136  180
  2              LF . . . . . .     LITERALLY        305  329  349  365  387  409  427  453  477
  2              LIT. . . . . .     LITERALLY          2   14   15   40   91   92   93   94   95
219  0004H     1  MAX. . . . . .     BYTE PARAMETER AUTOMATIC        220
199  0004H     1  MAX. . . . . .     BYTE PARAMETER AUTOMATIC        200  203  209
116  0004H     1  MAX. . . . . .     BYTE PARAMETER AUTOMATIC        117  121
  3  0000H     2  MAXB . . . . . .   WORD EXTERNAL(1)
  4  0000H        MON1 . . . . . .   PROCEDURE EXTERNAL(3) STACK=0000H        305  329  349  365  387  409
                                       427  453  477
  7  0000H        MON2 . . . . . .   PROCEDURE BYTE EXTERNAL(4) STACK=0000H
 10  0000H        MON3 . . . . . .   PROCEDURE WORD EXTERNAL(5) STACK=0000H
              MOVE . . . . . .     BUILTIN          45   59   73
  2              NOPAGEMODEOFFSET .   LITERALLY
 42  000EH     1  NUMSPACES. . . .   BYTE         44   45   48   66   71   73
 98  0014H     1  NXTCHAR. . . . .   BYTE        109  171  313  366
 97  0009H     1  NXTTOKEN . . . .   BYTE MEMBER(PCB)          355  376  415  458  507
 91              OPTIONMAX. . . .   LITERALLY
```

```
  2                    PAGELENOFFSET. . .   LITERALLY
291  001BH     1       PARENS . . . . . .   BYTE INITIAL          418  434  444  453  466  485
 97  0000H    10       PCB. . . . . .       STRUCTURE BASED(PCBBASE)        113  130  131  195  196  203  206  209
                        213  216  224  227  254  256  261  270  271  272  280  285  295  298
                        299  354  355  375  376  414  415  418  458  459  466  502  503  507
                        508  509  511  513  515  517  519  521  525  526  527  532  533  534
499  0004H     2       PCBADR . . . . . .   WORD PARAMETER AUTOMATIC       500  501
529  0004H     2       PCBADR . . . . . .   WORD PARAMETER AUTOMATIC       530  531
 96  0002H     2       PCBBASE. . . . . .   WORD          97  113  130  131  195  196  203  206  209  213  216  224
                        227  234  256  261  270  271  272  280  285  295  298  299  354  355
                        375  376  414  415  418  458  459  466  501  502  503  507  508  509
                        511  513  515  517  519  521  525  526  527  531  532  533  534
 23  0000H             PDECIMAL . . . . .   PROCEDURE EXTERNAL(9) STACK=0000H
 97  0008H     1       PLEVEL . . . . . .   BYTE MEMBER(PCB)              418  466
 23  0000H     2       PREC . . . . . .     WORD PARAMETER          24
 19  0000H             PRINTB . . . . . .   PROCEDURE EXTERNAL(7) STACK=0000H
 16  0000H             PRINTCHAR. . . . .   PROCEDURE EXTERNAL(6) STACK=0000H        306  330  350  366  388  410
                        428  454  478
111  029BH    38       PUTCHAR. . . . . .   PROCEDURE STACK=0004H          122  157  161  174  179  204  211  225
                        246  267
239  0019H     1       SAVEINPTR. . . . .   BYTE          240  243
139  0008H     2       SAVEINPTR. . . . .   WORD          145  153  159
499  0AD5H   195       SCAN . . . . . .     PROCEDURE PUBLIC STACK=0044H
 98  0004H     2       SCANADR. . . . . .   WORD           98  108  109  502
 97  0002H     2       SCANADR. . . . . .   WORD MEMBER(PCB)              299  502  526  527  532  533
529  0B98H    44       SCANINIT . . . . .   PROCEDURE PUBLIC STACK=0012H
  1  0000H             SCANNER. . . . . .   PROCEDURE STACK=0000H
  2                    SECTORLEN. . . . .   LITERALLY
472  0A6AH   107       STARTSTATE . . . .   PROCEDURE BYTE STACK=003EH              511  512  534
 97  0000H     2       STATE. . . . . .     WORD MEMBER(PCB)              295  354  375  414  459  511  513  515  517
                        519  521  534
450  0A07H    99       STATE1 . . . . . .   PROCEDURE BYTE REENTRANT STACK=003AH             317  401  446  448  459
                        488  513  514
424  098DH   122       STATE2 . . . . . .   PROCEDURE BYTE REENTRANT STACK=0036H             314  398  469  496
406  093AH    83       STATE3 . . . . . .   PROCEDURE BYTE REENTRANT STACK=0032H             322  324  395  414  439
                        493  515  516
383  08C2H   120       STATE4 . . . . . .   PROCEDURE BYTE REENTRANT STACK=002EH             421
362  086EH    84       STATE5 . . . . . .   PROCEDURE BYTE REENTRANT STACK=0028H             375  403  517  518
346  0829H    69       STATE6 . . . . . .   PROCEDURE BYTE REENTRANT STACK=0024H             337  354  371  519  520
326  07D9H    80       STATE7 . . . . . .   PROCEDURE BYTE REENTRANT STACK=000CH             359
302  076CH   109       STATE8 . . . . . .   PROCEDURE BYTE REENTRANT STACK=0008H             342  380
500  001CH     1       STATUS . . . . . .   BYTE          512  514  516  518  520  522  523  524
 27  0000H     1       STR. . . . . .       BYTE BASED(STRADR) ARRAY(1)             29
 26  0004H     2       STRADR . . . . . .   WORD PARAMETER AUTOMATIC          27   29
 42  000FH     1       STRING . . . . . .   BYTE           43   48   51   56   62   75
 40                    STRINGMARKER . . .   LITERALLY          52   61  221  224  229  232
  2                    TAB. . . . . .       LITERALLY          29
 98  0015H     1       TCOUNT . . . . . .   BYTE         118  120  121  124  128  131  132
384  0002H     1       TEMP . . . . . .     BYTE AUTOMATIC         392  394  396  402
 95                    TERROR . . . . . .   LITERALLY          509  525
 95                    TFILESPEC. . . . .   LITERALLY          196
 95                    TIDENTIFIER. . . .   LITERALLY          130
 95                    TMODIFIER. . . . .   LITERALLY          256  261  355  376
 95                    TNULL. . . . . .     LITERALLY          256  280  298  507
 95                    TNUMERIC . . . . .   LITERALLY          216
 98  0000H     1       TOKEN. . . . . .     BYTE BASED(TOKENADR) ARRAY(1)           114  169  181
 98  0006H     2       TOKENADR . . . . .   WORD           98  114  169  181  503
```

```
  97  0004H     2  TOKENADR . . . . .  WORD MEMBER(PCB)          113  503
  97  0007H     1  TOKENLEN . . . . .  BYTE MEMBER(PCB)          131  195  203  206  209  213  224  227  271
                                        272  508
  94             TOKENMAX . . . . .  LITERALLY        224  247  248  249  268  271  272
  97  0006H     1  TOKENTYPE. . . . .  BYTE MEMBER(PCB)          130  196  215  234  256  261  270  280  285
                                        298  509  525
  95             TOPTION. . . . . .  LITERALLY        270  415
  95             TPARAM . . . . . .  LITERALLY        280  285  458
  98  0012H     1  TPTR . . . . . .  BYTE        114  140  141  144  190  245  504
   2             TRUE . . . . . .  LITERALLY         37   51   75  164  197  217  236  242  251  257  262
                                        273  281  285  296  300  356  377  416  460  475
  95             TSTRING. . . . . .  LITERALLY        234
  82  0211H    47  UPPERCASE. . . . .  PROCEDURE STACK=0004H          533
  23  0000H     2  V. . . . . .  WORD PARAMETER         24
  26  0000H    58  WHITESPACE . . . .  PROCEDURE BYTE STACK=0006H          44   48   66
 134  0004H     1  X. . . . . .  BYTE PARAMETER AUTOMATIC          135  136
  23  0000H     1  ZEROSUP. . . . . .  BYTE PARAMETER         24
```

MODULE INFORMATION:

```
  CODE AREA SIZE     = 0BC4H   3012D
  CONSTANT AREA SIZE = 00A9H    169D
  VARIABLE AREA SIZE = 001DH     29D
  MAXIMUM STACK SIZE = 0044H     68D
  787 LINES READ
  0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

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


                $title ('SDIR - Search For Files')
  1             search:
                do;
                        /* search module for extended dir */

                $include (comlit.lit)
                =
  2   1         =   declare
                =           lit                 literally           'literally',
                =           dcl                 lit                 'declare',
                =           true                lit                 '0ffh',
                =           false               lit                 '0',
                =           boolean             lit                 'byte',
                =           forever             lit                 'while true',
                =           cr                  lit                 '13',
                =           lf                  lit                 '10',
                =           tab                 lit                 '9',
                =           ctrlc               lit                 '3',
                =           ff                  lit                 '12',
                =           page$len$offset     lit                 '1ch',
                =           nopage$mode$offset  lit                 '2Ch',
                =           sectorlen           lit                 '128';
                $include (mon.plm)
                =
                =                       /* definitions for assembly interface module        */
  3   1         =   declare
                =       fcb (33) byte external,         /* default file control block    */
                =       maxb address external,          /* top of memory                 */
                =       buff(128)byte external;         /* default buffer                */
                =
  4   1         =   mon1: procedure(f,a) external;
  5   2         =       declare f byte, a address;
  6   2         =       end mon1;
                =
  7   1         =   mon2: procedure(f,a) byte external;
  8   2         =       declare f byte, a address;
  9   2         =       end mon2;
                =
 10   1         =   mon3: procedure(f,a) address external;
 11   2         =       declare f byte, a address;
 12   2         =       end mon3;
                =


 13   1             dcl debug boolean external;

 14   1             dcl first$pass boolean external;
 15   1             dcl get$all$dir$entries boolean external;
 16   1             dcl usr$vector address external;
 17   1             dcl active$usr$vector address external;
 18   1             dcl used$de address public;                 /* used directory entries        */
 19   1             dcl filesfound address public;              /* num files collected in memory */
```

```
                $include(fcb.lit)
                =
20    1         =   declare
                =       f$drvusr            lit '0',        /* drive/user byte                  */
                =       f$name              lit '1',        /* file name                        */
                =       f$namelen           lit '8',        /* file name length                 */
                =       f$type              lit '9',        /* file type field                  */
                =       f$typelen           lit '3',        /* type length                      */
                =       f$rw                lit '9',        /* high bit is R/W attribute        */
                =       f$dirsys            lit '10',       /* high bit is dir/sys attribute    */
                =       f$arc               lit '11',       /* high bit is archive attribute    */
                =       f$ex                lit '12',       /* extent                           */
                =       f$s1                lit '13',       /* module byte                      */
                =       f$rc                lit '15',       /* record count                     */
                =       f$diskmap           lit '16',       /* file disk map                    */
                =       diskmaplen          lit '16',       /* disk map length                  */
                =       f$drvusr2           lit '16',       /* fcb2                             */
                =       f$name2             lit '17',
                =       f$type2             lit '25',
                =       f$rrec              lit '33',       /* random record                    */
                =       f$rreco             lit '35';       /*   "       "     overflow         */
                =
                $include(xfcb.lit)
                =
21    1         =   declare                                 /* XFCB                             */
                =       xfcb$type           lit '10h',      /* identifier on disk               */
                =       xf$passmode         lit '12',       /* pass word protection mode        */
                =       xf$pass             lit '16',       /* XFCB password                    */
                =       passlen             lit '8',        /* password length                  */
                =       xf$create           lit '24',       /* creation/access time stamp       */
                =       xf$update           lit '28';       /* update time stamp                */
                =
22    1         =   declare                       /* directory label: special case of XFCB */
                =       dirlabeltype        lit '20h',      /* identifier on disk               */
                =       dl$password         lit '128',      /* masks on data byte               */
                =       dl$access           lit '64',
                =       dl$update           lit '32',
                =       dl$makexfcb         lit '16',
                =       dl$exists           lit '1';
                =
23    1         =   declare                                 /* password mode of xfcb            */
                =       pm$read             lit '80h',
                =       pm$write            lit '40h',
                =       pm$delete           lit '20h';
                =

24    1             declare
                      sfcb$type lit '21H',
                      deleted$type lit '0E5H';

                $include (search.lit)
                =
25    1         =   declare                       /* what kind of file user wants to find      */
                =       find$structure lit 'structure (
                =       dir byte,
                =       sys byte,
```

```
            =       ro  byte,
            =       rw  byte,
            =       pass byte,
            =       xfcb byte,
            =       nonxfcb byte,
            =       exclude byte)';
            =
 26    1    =   declare
            =       max$search$files literally '10';
            =
 27    1    =   declare
            =       search$structure lit 'structure(
            =       drv byte,
            =       name(8) byte,
            =       type(3) byte,
            =       anyfile boolean)';      /* match on any drive if true */
            =
 28    1        dcl find find$structure external;      /* what kind of files to look for */
 29    1        dcl num$search$files byte external;
 30    1        dcl search (max$search$files) search$structure external;
                                               /* file specs to match on         */

                    /* other globals        */

 31    1        dcl cur$usr byte external,
                    cur$drv byte external,       /* current drive   "     "          */
                    dir$label byte public;      /* directory label for BDOS 3.0      */


                /* -------- BDOS calls -------- */

 32    1        read$char: procedure byte;
 33    2            return mon2 (1,0);
 34    2        end read$char;


                /* -------- in sort.plm -------- */

 35    1        mult23: procedure(f$info$index) address external;
 36    2            dcl f$info$index address;
 37    2        end mult23;


                /* -------- in util.plm -------- */

 38    1        print: procedure(string$adr) external;
 39    2            dcl string$adr address;
 40    2        end print;

 41    1        print$char: procedure(char) external;
 42    2            dcl char byte;
 43    2        end print$char;

 44    1        pdecimal:procedure(val,prec,zsup) external;
 45    2            dcl (val, prec) address;
 46    2            dcl zsup boolean;
 47    2        end pdecimal;
```

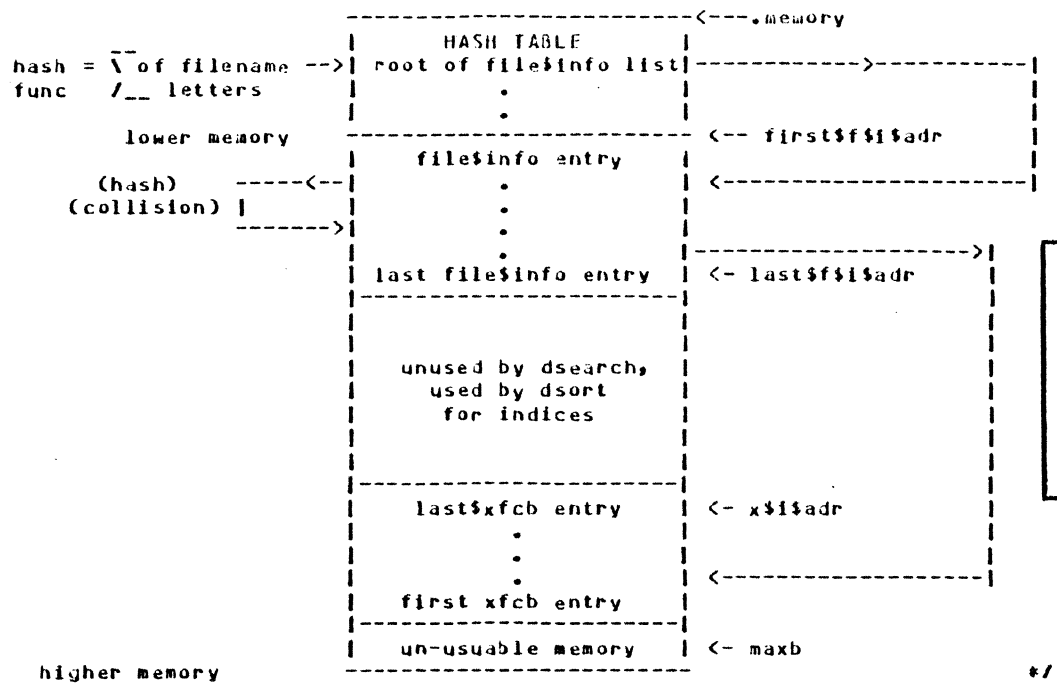
```
48   1      printfn: procedure(fnameadr) external;
49   2          dcl fnameadr address;
50   2      end printfn;

51   1      crlf: procedure external;   /* print carriage return, linefeed */
52   2      end crlf;

53   1      add3byte: procedure(byte3adr,num) external;
54   2            dcl (byte3adr,num) address;
55   2      end add3byte;

                 /* add three byte number to 3 byte accumulater */
56   1      add3byte3: procedure(totalb,numb) external;
57   2            dcl (totalb,numb) address;
58   2      end add3byte3;

                 /* divide 3 byte value by 8 */
59   1      shr3byte: procedure(byte3adr) external;
60   2            dcl byte3adr address;
61   2      end shr3byte;

            /* -------- In dpb86.plm -------- */

            $include(dpb.lit)
        =
        =   /* indices into disk parameter block, used as parameters to dpb procedure */
        =
62   1  =   dcl         spt$w       lit       '0',
        =               blkshf$b    lit       '2',
        =               blkmsk$b    lit       '3',
        =               extmsk$b    lit       '4',
        =               blkmax$w    lit       '5',
        =               dirmax$w    lit       '7',
        =               dirblk$w    lit       '9',
        =               chksiz      lit       '11',
        =               offset$w    lit       '13';
        =

63   1      dcl k$per$block byte external;        /* set in dpb module */

64   1      base$dpb: procedure external;
65   2      end base$dpb;

66   1      dpb$byte: procedure(param) byte external;
67   2          dcl param byte;
68   2      end dpb$byte;

69   1      dpb$word: procedure(param) address external;
70   2          dcl param byte;
71   2      end dpb$word;


            /* -------- Some Utility Routines -------- */

72   1      check$console$status: procedure byte;
73   2          return mon2 (11,0);
```

```
 74  2       end check$console$status;

 75  1       search$first: procedure (fcb$address) byte public;
 76  2           declare fcb$address address;          /* shared with disp.plm */
 77  2           return mon2 (17,fcb$address);         /* for short display    */
 78  2       end search$first;

 79  1       search$next: procedure byte public;       /* shared with disp.plm */
 80  2           return mon2 (18,0);
 81  2       end search$next;

 82  1       terminate: procedure external;            /* in main.plm */
 83  2       end terminate;

 84  1       set$vec: procedure(vector,value) external;   /* in main.plm */
 85  2       dcl vector address,
               value byte;
 86  2       end set$vec;

          /*break: procedure public;                     shared with disp.plm */
          /*    dcl x byte;
              if check$console$status then
              do;
                  x = read$char;
                  call terminate;
              end;
          end break;*/


          /* -------- file information record declaration -------- */

          $include(finfo.lit)
      =
      =   /* file info record for SDIR - note if this structure changes in size  */
      =   /* the multXX: routine in the sort.plm module must also change         */
      =
 87  1 =  declare
      =           f$info$structure lit 'structure(
      =               usr byte, name (8) byte, type (3) byte, onekblocks address,
      =               kbytes address, recs$lword address, recs$hbyte byte,
      =               hash$link address, x$i$adr address)';
 88  1 =  declare
      =           x$info$structure lit 'structure (
      =               create (4) byte,
      =               update (4) byte,
      =               passmode byte)';
      =


 89  1       declare
                  buf$fcb$adr address public,     /* index into directory buffer */
                  buf$fcb based buf$fcb$adr (32) byte,
                                                  /* fcb template for dir        */
                  (first$f$i$adr, f$i$adr, last$f$i$adr) address public,
                                                  /* indices into file$info array */
                  file$info based f$i$adr f$info$structure,
                  sfcb$adr address,
                  dir$type based sfcb$adr byte,
```

```
                sfcbs$present byte,
                x$i$adr address public,
                xfcb$info based x$i$adr x$info$structure;

 90   1     compare: procedure(length, str1$adr, str2$adr) boolean;
 91   2         dcl (length,i) byte,
                (str1$adr, str2$adr) address,
                str1 based str1$adr (1) byte,
                str2 based str2$adr (1) byte;
                    /* str2 is the possibly wildcarded filename we are looking for */

 92   2         do i = 0 to length - 1;
 93   3             if ((str1(i) and 7fh) <> (str2(i) and 7fh)) and str2(i) <> '?' then
 94   3                 return(false);
 95   3         end;
 96   2         return(true);
 97   2     end compare;

 98   1     match: procedure boolean public;
 99   2     dcl i byte,
                temp address;
100   2         if (i := (buf$fcb(f$drvusr) and 0fh)) <> cur$usr then
101   2             if not get$all$dir$entries then       /* Not looking for this user  */
102   2                 return(false);                    /* and not buffering all other*/
                  else                                  /* specified user files on    */
103   2                 do; temp = 0;                     /* this drive.                */
105   3                     call set$vec(.temp,i);
106   3                     if (temp and usr$vector) = 0 then /* Getting all dir entries,  */
107   3                         return(false);            /* with user number corresp'g */
108   3                 end;                              /* to a bit on in usr$vector  */

109   2         if usr$vector <> 0 and i <> 0 and first$pass <> 0 then
110   2             call set$vec(.active$usr$vector,i);   /* skip cur$usr files          */
                                                        /* build active usr vector for this drive */

111   2         do i = 0 to num$search$files - 1;
112   3             if search(i).drv = 0ffh or search(i).drv = cur$drv then
                            /* match on any drive if 0ffh */
113   3                 if search(i).anyfile = true then
114   3                     return(not find.exclude);     /* file found */
115   3                 else if compare(11,.buf$fcb(f$name),.search(i).name(0)) then
116   3                     return(not find.exclude);     /* file found */
              end;
118   2         return(find.exclude);      /* file not found */
119   2     end match;                     /* find.exclude = the exclude option value   */

120   1     dcl hash$table$size lit '128',                /* must be power of 2         */
              hash$table (hash$table$size) address at (.memory),
                                                        /* must be initialized on each*/
              hash$entry$adr address,                   /* disk scan                  */
              hash$entry based hash$entry$adr address;  /* where to put a new entry's */
                                                        /* address                    */

121   1     hash$look$up: procedure boolean;
122   2         dcl (i,found,hash$index) byte;
123   2         hash$index = 0;
124   2         do i = f$name to f$namelen + f$typelen;
```

```
125   3             hash$index = hash$index + (buf$fcb(i) and 1fh); /* attributes may   */
126   3         end;                                  /* only be set w/ 1st extent   */
127   2         hash$index = hash$index + cur$usr;
128   2         hash$index = hash$index and (hash$table$size - 1);
129   2         hash$entry$adr = .hash$table(hash$index); /* put new entry in table if  */
130   2         f$i$adr = hash$table(hash$index);          /* unused ( = 0)              */

131   2         found = false;
132   2         do while f$i$adr <> 0 and not found;
133   3             if file$info.usr = (buf$fcb(f$drvusr) and 0fh) and
                      compare(f$namelen + f$typelen,.file$info.name(0),.buf$fcb(f$name))
                      then
134   3                 found = true;
                  else                                  /* table entry used - collison */
135   3                 do; hash$entry$adr = .file$info.hash$link; /* resolve by linked  */
137   4                 f$i$adr = file$info.hash$link;             /* list               */
138   4                 end;
139   3         end;
140   2         if f$i$adr = 0 then
141   2             return(false);   /* didn't find it, used hash$entry to keep new info */
142   2         else return(true);   /* found it, file$info at matched entry             */
143   2     end hash$look$up;
```

```
          $eject
144   1   store$file$info: procedure boolean;
            /* Look for file name of last found fcb or xfcb in fileinfo      */
            /* array, if not found put name in fileinfo array.  Copy other   */
            /* info to fileinfo or xfcbinfo.  The lookup is hash coded with  */
            /* collisions handled by linking up file$info records through    */
            /* the hash$link field of the previous file$info record.         */
            /* The file$info array grows upward in memory and the xfcbinfo   */
            /* grows downward.                                               */
            /*
```




```
                                                                          */
```



```
145   2       dcl (i, j, d$map$cnt) byte,
                  temp address;

146   2       store$file: procedure;
147   3         call move(f$namelen + f$typelen, .buf$fcb(f$name),.file$info.name);
                            /* attributes are not in XFCBs to copy again in case    */
                            /* XFCB came first in directory                         */

148   3           file$info.name(f$arc-1) = file$info.name(f$arc-1) and buf$fcb(f$arc);
                                          /* 0 archive bit if it is 0 in any dir entry */
149   3           d$map$cnt = 0;          /* count kilobytes for current dir entry     */
150   3           i = 1;                            /* 1 or 2 byte block numbers ?     */
151   3           if dpb$word(blk$max$w) > 255 then
152   3               i = 2;
153   3           do j = f$diskmap to f$diskmap + diskmaplen - 1 by i;
```

```
154   4                 temp = buf$fcb(j);
155   4                 if i = 2 then                       /* word block numbers      */
156   4                     temp = temp or buf$fcb(j+1);
157   4                 if temp <> 0 then                   /* allocated               */
158   4                     d$map$cnt = d$map$cnt + 1;
159   4               end;
160   3               if d$map$cnt > 0 then
161   3               do;
162   4                 call add3byte
                            (.file$info.recs$lword,
                              d$map$cnt * (dpb$byte(blkmsk$b) + 1) -
                              (  (128 - buf$fcb(f$rc)) and dpb$byte(blkmsk$b)  )
                            );
163   4                 file$info.onekblocks = file$info.onekblocks +
                            d$map$cnt * k$per$block -
                            shr( (128 - buf$fcb(f$rc)) and dpb$byte(blkmsk$b), 3 );
                             /* treat each directory entry separately for sparse files */
                             /* if copied to single density diskette, the number of 1kblocks */
164   4                 file$info.kbytes = file$info.kbytes + d$map$cnt * k$per$block;
165   4               end;
166   3             end;

167   2           if buf$fcb(f$drvusr) <> sfcb$type then do; /* don't put SFCB's in table */
169   3             if not hash$look$up then            /* not in table already        */
                                                      /* hash$entry is where to put adr of new entry */
170   3               do;                               /* copy to new position in file info array     */
171   4                 if (temp := mult23(files$found + 1)) > x$i$adr then
172   4                     return(false);                /* out of memory           */
173   4                 if (temp < first$f$i$adr) then
174   4                     return(false);              /* wrap around - out of memory */
175   4                 f$i$adr = (last$f$i$adr := last$f$i$adr + size(file$info));
176   4                 filesfound = filesfound + 1;
177   4                 call move(f$namelen + f$typelen, .buf$fcb(f$name),.file$info.name);
178   4                 file$info.usr = buf$fcb(f$drvusr) and 0fh;
179   4                 file$info.onekblocks,file$info.kbytes,file$info.recs$lword,
                              file$info.recs$hbyte, file$info.x$i$adr,file$info.hash$link = 0;
180   4                 hash$entry = f$i$adr;           /* save the address of file$info    */
181   4               end;                              /* zero totals for the new file     */
182   3           end;

                  /* else hash$lookup has set f$i$adr to the file entry already in the    */
                  /* hash table                        */

183   2           if sfcbs$present then do;           /* save sfcb,xfcb or fcb type info    */
185   3             if (buf$fcb(f$drvusr) and xfcb$type) = 0 then do;
187   4               if buf$fcb(f$drvusr) <> sfcb$type then do;
189   5                 if buf$fcb(f$ex) <= dpb$byte(extmsk$b) then do;
                            /* first extent? then store sfcb info into xfcb table */
191   6                   if last$f$i$adr + size(file$info) > x$i$adr - size(xfcb$info) then
192   6                     return(false);    /* out of memory */
193   6                   x$i$adr = x$i$adr - size(xfcb$info);
194   6                   call move(9,sfcb$adr,.xfcb$info.create);
195   6                   file$info.x$i$adr = x$i$adr;
196   6               end;
197   5                 call store$file;
198   5               end;
199   4             end;
```

```
200   3          end;
201   2          else do;                              /* no SFCB's present */
202   3            if (buf$fcb(f$drvusr) and xfcb$type) <> 0 then do;          /* XFCB */
204   4              if last$f$i$adr + size(file$info) > x$i$adr - size(xfcb$info) then
205   4                  return(false);                /* out of memory                */
206   4              x$i$adr = x$i$adr - size(xfcb$info);
207   4              call move(8,.buf$fcb(xf$create),.xfcb$info.create);
208   4              xfcb$info.passmode = buf$fcb(xf$passmode);
209   4              file$info.x$i$adr = x$i$adr;
210   4            end;
211   3            else call store$file;         /* must be a regular fcb then */
212   3          end;
213   2        return(true);                             /* success              */
214   2      end store$file$info;


                            /* Module Entry Point */

215   1      get$files: procedure public;          /* with one scan through directory get   */
216   2          dcl dcnt byte;                    /* files from currently selected drive   */

217   2          last$f$i$adr = first$f$i$adr - size(file$info);
                                                  /* after hash table                      */
            /* last$f$i$adr is the address of the highest file info record    */
            /* in memory                                                      */

218   2          do dcnt = 0 to hash$table$size - 1;   /* init hash table               */
219   3              hash$table(dcnt) = 0;
220   3          end;

221   2          x$i$adr = maxb;                  /* top of mem, put xfcb info here        */
222   2          call base$dpb;
223   2          dir$label,filesfound, used$de = 0;

224   2          fcb(f$drvusr) = '?';                          /* match all dir entries */
225   2          dcnt = search$first(.fcb);
226   2          sfcb$adr = 96 + .buff;            /* determine if SFCB's are present */
227   2          if dir$type = sfcb$type then
228   2          do;
229   3            sfcbs$present = true;
230   3            used$de = shr(1+dpb$word(dirmax$w),2);       /* count all sfcb's once */
231   3          end;
            else
232   2            sfcbs$present = false;
233   2          do while dcnt <> 255;
234   3              buf$fcb$adr = shl(dcnt and 11b,5)+.buff;  /* dcnt mod 4 * 32        */
235   3              if sfcbs$present then
236   3                sfcb$adr = 97 + (dcnt * 10) + .buff;  /* SFCB time & date stamp adr */
237   3              if (buf$fcb(f$drvusr) <> deleted$type) then
238   3              do;
239   4                if (buf$fcb(f$drvusr) <> sfcb$type) then
240   4                  used$de = used$de + 1;
241   4                if buf$fcb(f$drvusr) = dirlabel$type then   /* dir label ?        */
242   4                    dir$label = buf$fcb(f$ex);            /* save label info    */
243   4                else if match then
244   4                do;
245   5                    if not store$file$info then         /* store fcb or xfcb info */
```

```
246   5                 do;                                   /* out of space          */
247   6                     call print (.('Out of Memory',cr,lf,'$'));
248   6                     return;
249   6                 end;
250   5               end;
                    end;
                    /*call break;*/
252   3               dcnt = search$next;                     /* to next entry in directory */
253   3           end; /* of do while dcnt <> 255 */

254   2       end get$files;

255   1       search$init: procedure public;                  /* called once from main.plm  */
256   2           if (first$f$i$adr := (.hash$table + size(hash$table))) + size(file$info)
                    > maxb then
257   2           do;
258   3               call print(.('Not Enough Memory',cr,lf,'$'));
259   3               call terminate;
260   3           end;
261   2       end search$init;

262   1       end search;
```

CROSS-REFERENCE LISTING

| DEFN | ADDR | SIZE | NAME, ATTRIBUTES, AND REFERENCES |
|---|---|---|---|
| 4 | 0000H | 2 | A. . . . . WORD PARAMETER 5 |
| 10 | 0000H | 2 | A. . . . . WORD PARAMETER 11 |
| 7 | 0000H | 2 | A. . . . . WORD PARAMETER 8 |
| 17 | 0000H | 2 | ACTIVEUSRVECTOR. . WORD EXTERNAL(10) 110 |
| 53 | 0000H | | ADD3BYTE . . . . . PROCEDURE EXTERNAL(22) STACK=0000H 162 |
| 56 | 0000H | | ADD3BYTE3. . . . . PROCEDURE EXTERNAL(23) STACK=0000H |
| 30 | 000CH | 1 | ANYFILE. . . . . BYTE MEMBER(SEARCH) 113 |
| 64 | 0000H | | BASEDPB. . . . . PROCEDURE EXTERNAL(26) STACK=0000H 222 |
| 62 | | | BLKMAXW. . . . . LITERALLY 151 |
| 62 | | | BLKMSKB. . . . . LITERALLY 162 163 |
| 62 | | | BLKSHFB. . . . . LITERALLY |
| 2 | | | BOOLEAN. . . . . LITERALLY 13 14 15 30 46 90 98 121 144 |
| 3 | 0000H | 128 | BUFF . . . . . BYTE ARRAY(128) EXTERNAL(2) 226 234 236 |
| 89 | 0000H | 32 | BUFFCB . . . . . BYTE BASED(BUFFCBADR) ARRAY(32) 100 115 125 133 147 148 154 156 162 163 167 177 178 185 187 189 202 207 208 237 239 241 242 |
| 89 | 0004H | 2 | BUFFCBADR. . . . . WORD PUBLIC 89 100 115 125 133 147 148 154 156 162 163 167 177 178 185 187 189 202 207 208 234 237 239 241 242 |
| 59 | 0000H | 2 | BYTE3ADR . . . . . WORD PARAMETER 60 |
| 53 | 0000H | 2 | BYTE3ADR . . . . . WORD PARAMETER 54 |
| 41 | 0000H | 1 | CHAR . . . . . BYTE PARAMETER 42 |
| 72 | 000FH | 15 | CHECKCONSOLESTATUS PROCEDURE BYTE STACK=0008H |
| 62 | | | CHKSIZ . . . . . LITERALLY |
| 90 | 003DH | 65 | COMPARE. . . . . PROCEDURE BYTE STACK=0008H 115 133 |
| 2 | | | CR . . . . . LITERALLY 247 258 |
| 89 | 0000H | 4 | CREATE . . . . . BYTE ARRAY(4) MEMBER(XFCBINFO) 194 207 |
| 51 | 0000H | | CRLF . . . . . PROCEDURE EXTERNAL(21) STACK=0000H |
| 2 | | | CTRLC. . . . . LITERALLY |
| 31 | 0000H | 1 | CURDRV . . . . . BYTE EXTERNAL(15) 112 |
| 31 | 0000H | 1 | CURUSR . . . . . BYTE EXTERNAL(14) 100 127 |
| 2 | | | DCL. . . . . LITERALLY |
| 216 | 0020H | 1 | DCNT . . . . . BYTE 218 219 225 233 234 236 252 |
| 13 | 0000H | 1 | DEBUG. . . . . BYTE EXTERNAL(6) |
| 24 | | | DELETEDTYPE. . . . LITERALLY 237 |
| 28 | 0000H | 1 | DIR. . . . . BYTE MEMBER(FIND) |
| 62 | | | DIRBLKW. . . . . LITERALLY |
| 31 | 0016H | 1 | DIRLABEL . . . . . BYTE PUBLIC 223 242 |
| 22 | | | DIRLABELTYPE . . . LITERALLY 241 |
| 62 | | | DIRMAXW. . . . . LITERALLY 230 |
| 89 | 0000H | 1 | DIRTYPE. . . . . BYTE BASED(SFCBADR) 227 |
| 20 | | | DISKMAPLEN . . . . LITERALLY 153 |
| 22 | | | DLACCESS . . . . . LITERALLY |
| 22 | | | DLEXISTS . . . . . LITERALLY |
| 22 | | | DLMAKEXFCB . . . . LITERALLY |
| 22 | | | DLPASSWORD . . . . LITERALLY |
| 22 | | | DLUPDATE . . . . . LITERALLY |
| 145 | 001FH | 1 | DMAPCNT. . . . . BYTE 149 158 160 162 163 164 |
| 66 | 0000H | | DPBBYTE. . . . . PROCEDURE BYTE EXTERNAL(27) STACK=0000H 162 163 189 |

```
 69  0000H        OPBWORD. . . . . PROCEDURE WORD EXTERNAL(28) STACK=0000H          151  230
 30  0000H     1  ORV. . . . . . . BYTE MEMBER(SEARCH)          112
 28  0007H     1  EXCLUDE. . . . . BYTE MEMBER(FIND)            114  116  118
 62             EXTMSKB. . . . . LITERALLY        189
  4  0000H     1  F. . . . . . . . BYTE PARAMETER              5
  7  0000H     1  F. . . . . . . . BYTE PARAMETER              8
 10  0000H     1  F. . . . . . . . BYTE PARAMETER             11
  2             FALSE. . . . . . LITERALLY         94  102  107  131  141  172  174  192  205  232
 20             FARC . . . . . . LITERALLY        148
  3  0000H    33  FCB. . . . . . . BYTE ARRAY(33) EXTERNAL(0)          224  225
 75  0004H     2  FCBADDRESS . . . . WORD PARAMETER AUTOMATIC          76   77
 20             FDIRSYS. . . . . LITERALLY
 20             FDISKMAP . . . . . LITERALLY        153
 20             FDRVUSR. . . . . LITERALLY        100  133  167  178  185  187  202  224  237  239  241
 20             FDRVUSR2 . . . . . LITERALLY
 20             FEX. . . . . . . . LITERALLY        139  242
  2             FF . . . . . . . . LITERALLY
 89  0008H     2  FIADR. . . . . . WORD PUBLIC             89  130  132  133  136  137  140  147  148  162
                                    163  164  175  177  178  179  180  195  209
 89  0000H    23  FILEINFO . . . . . STRUCTURE BASED(FIADR)          133  136  137  147  148  162  163  164
                                    175  177  178  179  191  195  204  209  217  256
 19  0002H     2  FILESFOUND . . . . WORD PUBLIC             171  176  223
 28  0000H     8  FIND . . . . . . STRUCTURE EXTERNAL(11)          114  116  118
 25             FINDSTRUCTURE. . . LITERALLY         28
 35  0000H     2  FINFOINDEX . . . . WORD PARAMETER             36
 87             FINFOSTRUCTURE . . LITERALLY         89
 89  0006H     2  FIRSTFIADR . . . . WORD PUBLIC             173  217  256
 14  0000H     1  FIRSTPASS. . . . . BYTE EXTERNAL(7)           109
 20             FNAME. . . . . . LITERALLY        115  124  133  147  177
 20             FNAME2 . . . . . LITERALLY
 48  0000H     2  FNAMEADR . . . . . WORD PARAMETER             49
 20             FNAMELEN . . . . . LITERALLY        124  133  147  177
  2             FOREVER. . . . . LITERALLY
122  001BH     1  FOUND. . . . . . BYTE         131  132  134
 20             FRC. . . . . . . LITERALLY        162  163
 20             FRREC. . . . . . LITERALLY
 20             FRRECO . . . . . LITERALLY
 20             FRW. . . . . . . LITERALLY
 20             FS1. . . . . . . LITERALLY
 20             FTYPE. . . . . . LITERALLY
 20             FTYPE2 . . . . . LITERALLY
 20             FTYPELEN . . . . . LITERALLY        124  133  147  177
 15  0000H     1  GETALLDIRENTRIES . BYTE EXTERNAL(8)           101
215  043AH   234  GETFILES . . . . . PROCEDURE PUBLIC STACK=0016H
120  0000H     2  HASHENTRY. . . . . WORD BASED(HASHENTRYADR)          180
120  0012H     2  HASHENTRYADR . . . WORD         120  129  136  180
122  001CH     1  HASHINDEX. . . . . BYTE         123  125  127  128  129  130
 89  0013H     2  HASHLINK . . . . . WORD MEMBER(FILEINFO)             136  137  179
121  0146H   189  HASHLOOKUP . . . . PROCEDURE BYTE STACK=000EH        169
120  0000H   256  HASHTABLE. . . . . WORD ARRAY(128) AT          129  130  219  256
120             HASHTABLESIZE. . . LITERALLY        120  128  218
122  001AH     1  I. . . . . . . . . BYTE         124  125
 99  0019H     1  I. . . . . . . . . BYTE         100  105  109  110  111  112  113  115
 91  0018H     1  I. . . . . . . . . BYTE          92   93
145  001DH     1  I. . . . . . . . . BYTE         150  152  153  155
145  001EH     1  J. . . . . . . . . BYTE         153  154  156
 89  000EH     2  KBYTES . . . . . . WORD MEMBER(FILEINFO)             164  179
```

```
 63  0000H     1  KPERBLOCK. . . . .  BYTE EXTERNAL(25)          163  164
 89  000AH     2  LASTFIADR. . . . .  WORD PUBLIC          175  191  204  217
 90  0008H     1  LENGTH . . . . . .  BYTE PARAMETER AUTOMATIC          91   92
  2              LF . . . . . . . .  LITERALLY       247  258
  2              LIT. . . . . . . .  LITERALLY         2   20   21   22   23   24   25   27   62   87   88
                                      120
 98  007EH   200  MATCH. . . . . . .  PROCEDURE BYTE PUBLIC STACK=000CH          243
  3  0000H     2  MAXB . . . . . . .  WORD EXTERNAL(1)          221  256
 26              MAXSEARCHFILES . .  LITERALLY        30
     0000H        MEMORY . . . . . .  BYTE ARRAY(0)         120
  4  0000H        MON1 . . . . . . .  PROCEDURE EXTERNAL(3) STACK=0000H
  7  0000H        MON2 . . . . . . .  PROCEDURE BYTE EXTERNAL(4) STACK=0000H          33   73   77   80
 10  0000H        MON3 . . . . . . .  PROCEDURE WORD EXTERNAL(5) STACK=0000H
                  MOVE . . . . . . .  BUILTIN         147  177  194  207
 35  0000H        MULT23 . . . . . .  PROCEDURE WORD EXTERNAL(16) STACK=0000H          171
 89  0001H     8  NAME . . . . . . .  BYTE ARRAY(8) MEMBER(FILEINFO)          133  147  148  177
 30  0001H     8  NAME . . . . . . .  BYTE ARRAY(8) MEMBER(SEARCH)          115
 28  0006H     1  NONXFCB. . . . . .  BYTE MEMBER(FIND)
  2              NOPAGEMODEOFFSET .  LITERALLY
 53  0000H     2  NUM. . . . . . . .  WORD PARAMETER         54
 56  0000H     2  NUMB . . . . . . .  WORD PARAMETER         57
 29  0000H     1  NUMSEARCHFILES . .  BYTE EXTERNAL(12)          111
 62              OFFSETW. . . . . .  LITERALLY
 89  000CH     2  ONEKBLOCKS . . . .  WORD MEMBER(FILEINFO)          163  179
  2              PAGELENOFFSET. . .  LITERALLY
 69  0000H     1  PARAM. . . . . . .  BYTE PARAMETER         70
 66  0000H     1  PARAM. . . . . . .  BYTE PARAMETER         67
 28  0004H     1  PASS . . . . . . .  BYTE MEMBER(FIND)
 21              PASSLEN. . . . . .  LITERALLY
 89  0008H     1  PASSMODE . . . . .  BYTE MEMBER(XFCBINFO)          208
 44  0000H        PDECIMAL . . . . .  PROCEDURE EXTERNAL(19) STACK=0000H
 23              PMDELETE . . . . .  LITERALLY
 23              PMREAD . . . . . .  LITERALLY
 23              PMWRITE. . . . . .  LITERALLY
 44  0000H     2  PREC . . . . . . .  WORD PARAMETER         45
 38  0000H        PRINT. . . . . . .  PROCEDURE EXTERNAL(17) STACK=0000H          247  258
 41  0000H        PRINTCHAR. . . . .  PROCEDURE EXTERNAL(18) STACK=0000H
 48  0000H        PRINTFN. . . . . .  PROCEDURE EXTERNAL(20) STACK=0000H
 32  0000H    15  READCHAR . . . . .  PROCEDURE BYTE STACK=0008H
 89  0012H     1  RECSHBYTE. . . . .  BYTE MEMBER(FILEINFO)          179
 89  0010H     2  RECSLWORD. . . . .  WORD MEMBER(FILEINFO)          162  179
 28  0002H     1  RO . . . . . . . .  BYTE MEMBER(FIND)
 28  0003H     1  RW . . . . . . . .  BYTE MEMBER(FIND)
  1  0000H        SEARCH . . . . . .  PROCEDURE STACK=0000H
 30  0000H   130  SEARCH . . . . . .  STRUCTURE ARRAY(10) EXTERNAL(13)          112  113  115
 75  001EH    16  SEARCHFIRST. . . .  PROCEDURE BYTE PUBLIC STACK=000AH          225
255  0524H    30  SEARCHINIT . . . .  PROCEDURE PUBLIC STACK=0006H
 79  002EH    15  SEARCHNEXT . . . .  PROCEDURE BYTE PUBLIC STACK=0008H          252
 27              SEARCHSTRUCTURE. .  LITERALLY        30
  2              SECTORLEN. . . . .  LITERALLY
 84  0000H        SETVEC . . . . . .  PROCEDURE EXTERNAL(30) STACK=0000H          105  110
 89  000CH     2  SFCBADR. . . . . .  WORD          89  194  226  227  236
 89  0017H     1  SFCBSPRESENT . . .  BYTE         183  229  232  235
 24              SFCBTYPE . . . . .  LITERALLY       167  187  227  239
                 SHL. . . . . . . .  BUILTIN         234
                 SHR. . . . . . . .  BUILTIN         163  230
 59  0000H        SHR3BYTE . . . . .  PROCEDURE EXTERNAL(24) STACK=0000H
```

```
                    SIZE . . . . . .    BUILTIN          175  191  193  204  206  217  256
    62                  SPTW . . . . . .    LITERALLY
   146  0338H   258     STORFILE. . . . .    PROCEDURE STACK=000CH            197  211
   144  0203H   309     STOREFILEINFO. . .   PROCEDURE BYTE STACK=0012H              245
    91  0000H     1     STR1 . . . . . .    BYTE BASED(STR1ADR) ARRAY(1)             93
    90  0006H     2     STR1ADR. . . . . .  WORD PARAMETER AUTOMATIC          91    93
    91  0000H     1     STR2 . . . . . .    BYTE BASED(STR2ADR) ARRAY(1)             93
    90  0004H     2     STR2ADR. . . . . .  WORD PARAMETER AUTOMATIC          91    93
    38  0000H     2     STRINGADR. . . . .  WORD PARAMETER          39
    28  0001H     1     SYS. . . . . .      BYTE MEMBER(FIND)
     2                  TAB. . . . . .      LITERALLY
   145  0014H     2     TEMP . . . . . .    WORD         154  156  157  171  173
    99  0010H     2     TEMP . . . . . .    WORD         104  105  106
    82  0000H           TERMINATE. . . . .  PROCEDURE EXTERNAL(29) STACK=0000H        259
    56  0000H     2     TOTALB . . . . . .  WORD PARAMETER          57
     2                  TRUE . . . . . .    LITERALLY          96  113  134  142  213  229
    89  0009H     3     TYPE . . . . . .    BYTE ARRAY(3) MEMBER(FILEINFO)
    30  0009H     3     TYPE . . . . . .    BYTE ARRAY(3) MEMBER(SEARCH)
    89  0004H     4     UPDATE . . . . . .  BYTE ARRAY(4) MEMBER(XFCBINFO)
    18  0000H     2     USEDDE . . . . . .  WORD PUBLIC          223  230  240
    89  0000H     1     USR. . . . . .      BYTE MEMBER(FILEINFO)            133  178
    16  0000H     2     USRVECTOR. . . . .  WORD EXTERNAL(9)           106  109
    44  0000H     2     VAL. . . . . .      WORD PARAMETER         45
    84  0000H     1     VALUE. . . . . .    BYTE PARAMETER         85
    84  0000H     2     VECTOR . . . . . .  WORD PARAMETER         85
    28  0005H     1     XFCB . . . . . .    BYTE MEMBER(FIND)
    89  0000H     9     XFCBINFO . . . . .  STRUCTURE BASED(XIADR)           191  193  194  204  206  207  208
    21                  XFCBTYPE . . . . .  LITERALLY        185  202
    21                  XFCREATE . . . . .  LITERALLY        207
    21                  XFPASS . . . . . .  LITERALLY
    21                  XFPASSMODE . . . .  LITERALLY        208
    21                  XFUPDATE . . . . .  LITERALLY
    89  000EH     2     XIADR. . . . . .    WORD PUBLIC          89  171  191  193  194  195  204  206  207  208
                                             209  221
    89  0015H     2     XIADR. . . . . .    WORD MEMBER(FILEINFO)            179  195  209
    88                  XINFOSTRUCTURE . .  LITERALLY        89
    44  0000H     1     ZSUP . . . . . .    BYTE PARAMETER         46
```



MODULE INFORMATION:

```
  CODE AREA SIZE     = 0542H   1346D
  CONSTANT AREA SIZE = 0024H     36D
  VARIABLE AREA SIZE = 0021H     33D
  MAXIMUM STACK SIZE = 0016H     22D
  552 LINES READ
  0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

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


                $title ('SDIR - Sort Module')
  1             sort:
                do;
                                /* sort module for extended dir */

                $include(comlit.lit)
        =
  2   1     =   declare
        =               lit                 literally           'literally',
        =               dcl                 lit                 'declare',
        =               true                lit                 '0ffh',
        =               false               lit                 '0',
        =               boolean             lit                 'byte',
        =               forever             lit                 'while true',
        =               cr                  lit                 '13',
        =               lf                  lit                 '10',
        =               tab                 lit                 '9',
        =               ctrlc               lit                 '3',
        =               ff                  lit                 '12',
        =               page$len$offset     lit                 '1ch',
        =               nopage$mode$offset  lit                 '20h',
        =               sectorlen           lit                 '128';

  3   1         print: procedure(str$adr) external;  /* in util.plm */
  4   2         dcl str$adr address;
  5   2         end print;

  6   1         dcl sorted boolean public;          /* set by this module if successful sort */

                $include(finfo.lit)
        =
        =       /* file info record for SDIR - note if this structure changes in size  */
        =       /* the multXX: routine in the sort.plm module must also change         */
        =
  7   1     =   declare
        =               f$info$structure lit 'structure(
        =                   usr byte, name (8) byte, type (3) byte, onekblocks address,
        =                   kbytes address, recs$lword address, recs$hbyte byte,
        =                   hash$link address, x$i$adr address)';
  8   1     =   declare
        =               x$info$structure lit 'structure (
        =                   create (4) byte,
        =                   update (4) byte,
        =                   passmode byte)';
        =

  9   1         declare
                        buf$fcb$adr address external,     /* index into directory buffer */
                        buf$fcb based buf$fcb$adr (32) byte,
                                                  /* fcb template for dir        */
```

```
                (f$i$adr, first$f$i$adr, last$f$i$adr, x$i$adr, filesfound)
                    address external,
                                        /* indices into file$info array    */
                file$info based f$i$adr f$info$structure;

                mid$adr address,
                mid$file$info based mid$adr f$info$structure;


10   1      mult23: procedure(index) address public;
11   2          dcl index address;   /* return address of file$info numbered by index */
12   2          return shl(index, 4) + shl(index,2) + shl(index,1) + index + first$f$i$adr;
                /* index * size(file$info) + base of file$info array */
13   2      end mult23;

14   1      lessthan: procedure( str1$adr, str2$adr) boolean;
15   2          dcl (i,c1,c2) byte,                 /* true if str1 < str2          */
                (str1$adr, str2$adr) address,       /* sorting on name and type field */
                str1 based str1$adr (1) byte,       /* only, assumed to be first in   */
                str2 based str2$adr (1) byte;       /* file$info record */
16   2          do i = 1 to 11;
17   3              if (c1:=(str1(i) and 7fh)) <> (c2:=(str2(i) and 7fh)) then
18   3                  return(c1 < c2);
19   3          end;
20   2          return(false);
21   2      end lessthan;

22   1      dcl f$i$indices$base address public,
                f$i$indices based f$i$indices$base (1) address;

23   1      qsort: procedure(l,r);      /* no recursive quick sort, sorting largest    */
24   2      dcl (l,r,i,j,temp) address,/* partition first                             */
                stacksiz lit '14',      /* should always be able to sort 2 ** stacksiz */
                stack (stack$siz) structure (l address, r address),
                sp byte;

25   2          sp = 0; stack(0).l = l; stack(0).r = r;

28   2          do while sp < stack$siz - 1;
29   3              l = stack(sp).l; r = stack(sp).r; sp = sp - 1;
32   3              do while l < r;
33   4                  i = l; j = r;
35   4                  mid$adr = mult23(f$i$indices(shr(l+r,1)));
36   4                  do while i <= j;
37   5                      f$i$adr = mult23(f$i$indices(i));
38   5                      do while lessthan(f$i$adr,mid$adr);
39   6                          i = i + 1;
40   6                          f$i$adr = mult23(f$i$indices(i));
41   6                      end;
42   5                      f$i$adr = mult23(f$i$indices(j));
43   5                      do while lessthan(mid$adr,f$i$adr);
44   6                          j = j - 1;
45   6                          f$i$adr = mult23(f$i$indices(j));
46   6                      end;
47   5                      if i <= j then
48   5                      do; temp = f$i$indices(i); f$i$indices(i) = f$i$indices(j);
51   6                          f$i$indices(j) = temp;
```

```
52   6                      i = i + 1;
53   6                      if j > 0 then j = j - 1;
55   6                  end;
56   5              end;  /* while i <= j     */
57   4              if j - l < r - i then        /* which partition is larger */
58   4              do; if i < r then
60   5                  do; sp = sp + 1; stack(sp).l = i; stack(sp).r = r;
64   6                  end;
65   5                  r = j;      /* continue sorting left partition */
66   5              end;
                    else
67   4              do; if l < j then
69   5                  do; sp = sp + 1; stack(sp).l = l; stack(sp).r = j;
73   6                  end;
74   5                  l = i;      /* continue sorting right partition */
75   5              end;
76   4          end;       /* while l < r               */
77   3      end;           /* while sp < stack$siz - 1 */
78   2      if sp <> 255 then
79   2          call print(.(cr,lf,lf,'Sort Stack Overflow',cr,lf,'$'));
80   2      else sorted = true;
81   2  end qsort;

82   1  sort: procedure public;
83   2      dcl i address;
84   2      f$i$indices$base = last$f$i$adr + size(file$info);
85   2      if filesfound < 2 then
86   2          return;
87   2      if shr((x$i$adr - f$i$indices$base),1) < filesfound then
88   2      do;
89   3          call print(.('Not Enough Memory for Sort',cr,lf,'$'));
90   3          return;
91   3      end;
92   2      do i = 0 to filesfound - 1;
93   3          f$i$indices(i) = i;                    /* initialize f$i$indices */
94   3      end;
95   2      call qsort(0,filesfound - 1);
96   2      sorted = true;
97   2  end sort;

98   1  end sort;
```

CROSS-REFERENCE LISTING



```
 DEFN  ADDR    SIZE  NAME, ATTRIBUTES, AND REFERENCES
-----  ------  ----  ---------------------------------

    2                BOOLEAN. . . . . LITERALLY          6    14
    9  0000H     32  BUFFCB . . . . . BYTE BASED(BUFFCBADR) ARRAY(32)
    9  0000H      2  BUFFCBADR. . . . . WORD EXTERNAL(1)                9
   15  0046H      1  C1 . . . . . . . . BYTE          17    18
   15  0047H      1  C2 . . . . . . . . BYTE          17    18
    2                CR . . . . . . . . LITERALLY          79    89
    2                CTRLC. . . . . . . LITERALLY
    2                DCL. . . . . . . . LITERALLY
    2                FALSE. . . . . . . LITERALLY          20
    2                FF . . . . . . . . LITERALLY
    9  0000H      2  FIADR. . . . . . . WORD EXTERNAL(2)                9    37    38    40    42    43    45
   22  0000H      2  FIINDICES. . . . . WORD BASED(FIINDICESBASE) ARRAY(1)      35    37    40    42    45    49
                                          50    51    93
   22  0002H      2  FIINDICESBASE. . . WORD PUBLIC              22    35    37    40    42    45    49    50    51    84
                                          87    93
    9  0000H     23  FILEINFO . . . . . STRUCTURE BASED(FIADR)                84
    9  0000H      2  FILESFOUND . . . . WORD EXTERNAL(6)                85    87    92    95
    7                FINFOSTRUCTURE . . LITERALLY          9
    9  0000H      2  FIRSTFIADR . . . . WORD EXTERNAL(3)                12
    2                FOREVER. . . . . . LITERALLY
    9  0013H      2  HASHLINK . . . . . WORD MEMBER(MIDFILEINFO)
    9  0013H      2  HASHLINK . . . . . WORD MEMBER(FILEINFO)
   15  0045H      1  I. . . . . . . . . BYTE          16    17
   24  0004H      2  I. . . . . . . . . WORD          33    36    37    39    40    47    49    50    52    57    59    62
                                          74
   83  0042H      2  I. . . . . . . . . WORD          92    93
   10  0004H      2  INDEX. . . . . . . WORD PARAMETER AUTOMATIC              11    12
   24  0006H      2  J. . . . . . . . . WORD          34    36    42    44    45    47    50    51    53    54    57    65
                                          68    72
    9  000EH      2  KBYTES . . . . . . WORD MEMBER(MIDFILEINFO)
    9  000EH      2  KBYTES . . . . . . WORD MEMBER(FILEINFO)
   23  0006H      2  L. . . . . . . . . WORD PARAMETER AUTOMATIC              24    26    29    32    33    35    57    68
                                          71    74
   24  0000H      2  L. . . . . . . . . WORD MEMBER(STACK)              26    29    62    71
    9  0000H      2  LASTFIADR. . . . . WORD EXTERNAL(4)                84
   14  0025H     66  LESSTHAN . . . . . PROCEDURE BYTE STACK=0006H                38    43
    2                LF . . . . . . . . LITERALLY          79    89
    2                LIT. . . . . . . . LITERALLY           2     7     8    24
    9  0000H      2  MIDADR . . . . . . WORD            9    35    38    43
    9  0000H     23  MIDFILEINFO. . . . STRUCTURE BASED(MIDADR)
   10  0000H     37  MULT23 . . . . . . PROCEDURE WORD PUBLIC STACK=0004H          35    37    40    42    45
    9  0001H      8  NAME . . . . . . . BYTE ARRAY(8) MEMBER(MIDFILEINFO)
    9  0001H      8  NAME . . . . . . . BYTE ARRAY(8) MEMBER(FILEINFO)
    2                NOPAGEMODEOFFSET . LITERALLY
    9  000CH      2  ONEKBLOCKS . . . . WORD MEMBER(MIDFILEINFO)
    9  000CH      2  ONEKBLOCKS . . . . WORD MEMBER(FILEINFO)
    2                PAGELENOFFSET. . . LITERALLY
    3  0000H         PRINT. . . . . . . PROCEDURE EXTERNAL(0) STACK=0000H          79    89
   23  0067H    373  QSORT. . . . . . . PROCEDURE STACK=000EH                95
```

```
 23  0004H     2  R. . . . . .    WORD PARAMETER AUTOMATIC        24   27   30   32   34   35   57   59
                                    63   65
 24  0002H     2  R. . . . . .    WORD MEMBER(STACK)         27   30   63   72
  9  0012H     1  RECSHBYTE. . . . .  BYTE MEMBER(MIDFILEINFO)
  9  0012H     1  RECSHBYTE. . . . .  BYTE MEMBER(FILEINFO)
  9  0010H     2  RECSLWORD. . . . .  WORD MEMBER(MIDFILEINFO)
  9  0010H     2  RECSLWORD. . . . .  WORD MEMBER(FILEINFO)
  2             SECTORLEN. . . . .  LITERALLY
                SHL. . . . . .    BUILTIN          12
                SHR. . . . . .    BUILTIN          35   87
                SIZE . . . . . .  BUILTIN          84
  1  0000H      SORT . . . . . .  PROCEDURE STACK=0000H
 82  01DCH   102  SORT . . . . . .  PROCEDURE PUBLIC STACK=0012H
  6  0044H     1  SORTED . . . . . .  BYTE PUBLIC            80   96
 24  0048H     1  SP . . . . . .    BYTE        25   28   29   30   31   61   62   63   70   71   72   78
 24  000AH    56  STACK. . . . . .  STRUCTURE ARRAY(14)       26   27   29   30   62   63   71   72
 24             STACKSIZ . . . . .  LITERALLY         24   28
 15  0000H     1  STR1 . . . . . .  BYTE BASED(STR1ADR) ARRAY(1)        17
 14  0006H     2  STR1ADR. . . . . .  WORD PARAMETER AUTOMATIC        15   17
 15  0000H     1  STR2 . . . . . .  BYTE BASED(STR2ADR) ARRAY(1)        17
 14  0004H     2  STR2ADR. . . . . .  WORD PARAMETER AUTOMATIC        15   17
  3  0000H     2  STRADR . . . . . .  WORD PARAMETER          4
  2             TAB. . . . . .    LITERALLY
 24  0008H     2  TEMP . . . . . .  WORD          49   51
  2             TRUE . . . . . .  LITERALLY         80   96
  9  0009H     3  TYPE . . . . . .  BYTE ARRAY(3) MEMBER(MIDFILEINFO)
  9  0009H     3  TYPE . . . . . .  BYTE ARRAY(3) MEMBER(FILEINFO)
  9  0000H     1  USR. . . . . .    BYTE MEMBER(MIDFILEINFO)
  9  0000H     1  USR. . . . . .    BYTE MEMBER(FILEINFO)
  9  0015H     2  XIADR. . . . . .  WORD MEMBER(MIDFILEINFO)
  9  0015H     2  XIADR. . . . . .  WORD MEMBER(FILEINFO)
  9  0000H     2  XIADR. . . . . .  WORD EXTERNAL(5)          87
  8             XINFOSTRUCTURE . .  LITERALLY
```



MODULE INFORMATION:

```
  CODE AREA SIZE      = 0242H    5780
  CONSTANT AREA SIZE  = 0036H      54D
  VARIABLE AREA SIZE  = 0049H      73D
  MAXIMUM STACK SIZE  = 0012H      18D
  149 LINES READ
  0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

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



```
                $title ('SDIR - Display Files')
  1             display:
                do;
                                /*  Display Module for SDIR */

                $include(comlit.lit)
                =
  2     1       =   declare
                =           lit                 literally           'literally',
                =           dcl                 lit                 'declare',
                =           true                lit                 '0ffh',
                =           false               lit                 '0',
                =           boolean             lit                 'byte',
                =           forever             lit                 'while true',
                =           cr                  lit                 '13',
                =           lf                  lit                 '10',
                =           tab                 lit                 '9',
                =           ctrlc               lit                 '3',
                =           ff                  lit                 '12',
                =           page$len$offset     lit                 '1ch',
                =           nopage$mode$offset  lit                 '20h',
                =           sectorlen           lit                 '128';

                $include(mon.plm)
                =
                =                       /* definitions for assembly interface module        */
  3     1       =   declare
                =       fcb (33) byte external,         /* default file control block    */
                =       maxb address external,          /* top of memory                 */
                =       buff(128)byte external;         /* default buffer                */
                =
  4     1       =   mon1: procedure(f,a) external;
  5     2       =       declare f byte, a address;
  6     2       =       end mon1;
                =
  7     1       =   mon2: procedure(f,a) byte external;
  8     2       =       declare f byte, a address;
  9     2       =       end mon2;
                =
 10     1       =   mon3: procedure(f,a) address external;
 11     2       =       declare f byte, a address;
 12     2       =       end mon3;
                =

 13     1           dcl (cur$drv, cur$usr) byte external;

 14     1           dcl (os,bdos) byte external;
                    $include(vers.lit)
 15     1       =   declare
                =      bdos20 lit '20h',
                =      bdos22 lit '22h',
```

```
        =       bdos30 lit '30h',
        =       mpm    lit '01h',
        =       cpm86  lit '10h',
        =       mpm86  lit '11h',
        =       ccpm86 lit '14h';

16  1       dcl used$de address external;       /* number of used directory entries */
17  1       dcl date$opt boolean external;      /* date option flag */
18  1       dcl display$attributes boolean external;    /* attributes display flag */
19  1       dcl sorted boolean external;
20  1       dcl filesfound address external;

            $include (search.lit)
        =
21  1   =   declare                       /* what kind of file user wants to find      */
        =       find$structure lit 'structure (
        =       dir byte,
        =       sys byte,
        =       ro  byte,
        =       rw  byte,
        =       pass byte,
        =       xfcb byte,
        =       nonxfcb byte,
        =       exclude byte)';
        =
22  1   =   declare
        =       max$search$files literally '10';
        =
23  1   =   declare
        =       search$structure lit 'structure(
        =       drv byte,
        =       name(8) byte,
        =       type(3) byte,
        =       anyfile boolean)';          /* match on any drive if true */
        =
24  1       dcl find find$structure external;

25  1       dcl format byte external,       /* format is one of the following */
                page$len address external,  /* page size before printing new headers */
                message boolean external,   /* print titles and msg when no file found */
                formfeeds boolean external; /* use form feeds to separate headers */

            $include(format.lit)
        =
26  1   =   dcl form$short lit '0',         /* format values for SDIR */
        =       form$size lit '1',
        =       form$full lit '2';
        =

27  1       dcl file$displayed boolean public initial (false);
                        /* true if we ever display a file, from any drive or user */
                        /* used by main.plm for file not found message            */

28  1       dcl dir$label byte external;

            $include(fcb.lit)
        =
```

```
29  1  =  declare
       =      f$drvusr          lit '0',        /* drive/user byte                */
       =      f$name            lit '1',        /* file name                      */
       =      f$namelen         lit '8',        /* file name length               */
       =      f$type            lit '9',        /* file type field                */
       =      f$typelen         lit '3',        /* type length                    */
       =      f$rw              lit '9',        /* high bit is R/W attribute      */
       =      f$dirsys          lit '10',       /* high bit is dir/sys attribute  */
       =      f$arc             lit '11',       /* high bit is archive attribute  */
       =      f$ex              lit '12',       /* extent                         */
       =      f$sl              lit '13',       /* module byte                    */
       =      f$rc              lit '15',       /* record count                   */
       =      f$diskmap         lit '16',       /* file disk map                  */
       =      diskmaplen        lit '16',       /* disk map length                */
       =      f$drvusr2         lit '16',       /* fcb2                           */
       =      f$name2           lit '17',
       =      f$type2           lit '25',
       =      f$rrec            lit '33',       /* random record                  */
       =      f$rreco           lit '35';       /*    "       "    overflow       */
       =
       =  $include(xfcb.lit)
       =
30  1  =  declare                                /* XFCB                           */
       =      xfcb$type         lit '10h',      /* identifier on disk             */
       =      xf$passmode       lit '12',       /* pass word protection mode      */
       =      xf$pass           lit '16',       /* XFCB password                  */
       =      passlen           lit '8',        /* password length                */
       =      xf$create         lit '24',       /* creation/access time stamp     */
       =      xf$update         lit '28';       /* update time stamp              */
       =
31  1  =  declare                    /* directory label: special case of XFCB */
       =      dirlabeltype      lit '20h',      /* identifier on disk             */
       =      dl$password       lit '128',      /* masks on data byte             */
       =      dl$access         lit '64',
       =      dl$update         lit '32',
       =      dl$makexfcb       lit '16',
       =      dl$exists         lit '1';
       =
32  1  =  declare                                /* password mode of xfcb          */
       =      pm$read           lit '80h',
       =      pm$write          lit '40h',
       =      pm$delete         lit '20h';
       =

33  1     dcl
             buf$fcb$adr address external,      /* index into directory buffer  */
             buf$fcb based buf$fcb$adr (32) byte,
                                                /* fcb template for dir         */

             (f$i$adr,last$f$i$adr,first$f$i$adr) address external,
             cur$file address;                  /* number of file currently     */
                                                /* being displayed              */

       =  $include(finfo.lit)
       =
       =  /* file info record for SDIR - note if this structure changes in size  */
       =  /* the multXX: routine in the sort.plm module must also change         */
```

```
        =
 34   1 =   declare
        =           f$info$structure lit 'structure(
        =               usr byte, name (8) byte, type (3) byte, onekblocks address,
        =               kbytes address, recs$lword address, recs$hbyte byte,
        =               hash$link address, x$i$adr address)';
 35   1 =   declare
        =           x$info$structure lit 'structure (
        =               create (4) byte,
        =               update (4) byte,
        =               passmode byte)';
        =
                                                    /* structure of file info       */
 36   1     dcl     file$info based f$i$adr f$info$structure;

 37   1     dcl     x$i$adr address external,
                    xfcb$info based x$i$adr x$info$structure;

 38   1     dcl     f$i$indices$base address external,  /* if sorted then f$i$indices  */
                    f$i$indices based f$i$indices$base (1) address; /* are here       */


              /* -------- Routines in util.plm -------- */

 39   1     printchar: procedure (char) external;
 40   2         dcl char byte;
 41   2     end printchar;

 42   1     print: procedure (string$adr) external;       /* BDOS call # 9            */
 43   2         dcl string$adr address;
 44   2     end print;

 45   1     printb: procedure external;
 46   2     end printb;

 47   1     crlf: procedure external;
 48   2     end crlf;

 49   1     printfn: procedure(fname$adr) external;
 50   2         dcl fname$adr address;
 51   2     end printfn;

 52   1     pdecimal: procedure(v,prec,zerosup) external;
                              /* print value val, field size = (log10 prec) + 1  */
                              /* with leading zero suppression if zerosup = true */
 53   2         declare v address,                        /* value to print        */
                       prec address,                     /* precision             */
                       zerosup boolean;                  /* zero suppression flag */
 54   2     end pdecimal;

 55   1     p3byte: procedure(byte3adr,prec)external;
                                    /* print 3 byte value with 0 suppression */
 56   2       dcl (byte3adr,prec) address; /* assume high order bit is < 10         */
 57   2     end p3byte;

 58   1     add3byte: procedure (byte3$adr,word$amt) external;
 59   2         dcl (byte3$adr, word$amt) address;
```

```
60   2      end add3byte;              /* add word to 3 byte structure */

61   1      add3byte3: procedure (byte3$adr,byte3) external;
62   2          dcl (byte3$adr, byte3) address;
63   2      end add3byte3;              /* add 3 byte quantity to 3 byte total */

64   1      shr3byte: procedure (byte3$adr) external;
65   2          dcl byte3$adr address;
66   2      end shr3byte;


            /* -------- Routines in search.plm -------- */

67   1      search$first: procedure(fcb$adr) byte external;
68   2          dcl fcb$adr address;
69   2      end search$first;

70   1      search$next: procedure byte external;
71   2      end search$next;

            /*break: procedure external;
            end break;*/

72   1      match: procedure boolean external;
73   2          dcl fcb$adr address;
74   2      end match;


            /* -------- Other external routines -------- */

75   1      display$time$stamp: procedure (ts$adr) external;      /* in dts.plm */
76   2          dcl ts$adr address;
77   2      end display$time$stamp;

78   1      terminate: procedure external;                        /* in main.plm */
79   2      end terminate;

80   1      mult23: procedure(index) address external;            /* in sort.plm */
81   2          dcl index address;
82   2      end mult23;


            /* -------- From dpb86.plm or dpb80.plm -------- */

            $include(dpb.lit)
        =
        =   /* indices into disk parameter block, used as parameters to dpb procedure */
        =
83   1  =   dcl         spt$w       lit         '0',
        =               blkshf$b    lit         '2',
        =               blkmsk$b    lit         '3',
        =               extmsk$b    lit         '4',
        =               blkmax$w    lit         '5',
        =               dirmax$w    lit         '7',
        =               dirblk$w    lit         '9',
        =               chksiz      lit         '11',
        =               offset$w    lit         '13';
```

```
 84   1      dpb$byte: procedure (dpb$index) byte external;
 85   2          dcl dpb$index byte;
 86   2      end dpb$byte;

 87   1      dpb$word: procedure (dpb$index) address external;
 88   2          dcl dpb$index byte;
 89   2      end dpb$word;


              /* -------- routines and data structures local to this module -------- */

 90   1      direct$console$io: procedure byte;
 91   2        return mon2(6,0ffh);  /* ff to stay downward compatable */
 92   2      end direct$console$io;

 93   1      wait$keypress: procedure;
 94   2        declare char byte;
 95   2        char = direct$console$io;
 96   2        do while char = 0;
 97   3          char = direct$console$io;
 98   3        end;
 99   2        if char = ctrlc then
100   2          call terminate;
101   2      end wait$keypress;

102   1      declare global$line$count byte initial(1);

103   1      dcl     total$kbytes structure (    /* grand total k bytes of files matched */
                        lword address,
                        hbyte byte),
                      total$recs structure (      /* grand total records of files matched */
                        lword address,
                        hbyte byte),
                      total$1k$blocks structure(  /* how many 1k blocks are allocated     */
                        lword address,
                        hbyte byte);

104   1      add$totals: procedure;

105   2          call add3byte(.total$kbytes,file$info.kbytes);
106   2          call add3byte3(.total$recs,.file$info.recs$lword); /* records in file */
107   2          call add3byte(.total$1k$blocks,file$info.onekblocks);

108   2      end add$totals;

109   1      dcl files$per$line byte;
110   1      dcl cur$line address;

111   1      dcl hdr (*) byte data      ('    Name     Bytes   Recs   Attributes $');
112   1      dcl hdr$bars (*) byte data ('------------ ------ ------ ------------$');
113   1      dcl hdr$pu (*) byte data       ('  Prot      Update    $');
114   1      dcl hdr$xfcb$bars (*) byte data (' ------ --------------  --------------$');
115   1      dcl hdr$access (*) byte data                          ('      Access    $');
116   1      dcl hdr$create (*) byte data                          ('      Create    $');
                                        /* example date        04/02/55 00:34  */
```

```
117   1     display$file$info: procedure;
                                                        /* print filename.typ   */
118   2         call printfn(.file$info.name(0));
119   2         call printb;
120   2         call pdecimal(file$info.kbytes,10000,true);
121   2         call printchar('k');                      /* up to 32 Meg - Bytes  */
                                                        /* or 32,000k            */
122   2         call printb;
123   2         call p3byte(.file$info.recs$lword,1);     /* records              */
124   2         call printb;
125   2         if rol(file$info.name(f$dirsys-1),1) then /* Type                 */
126   2            call print(.('Sys$'));
127   2         else call print(.('Dir$'));
128   2         call printb;
129   2         if rol(file$info.name(f$rw-1),1) then
130   2             call print(.('RO$'));
131   2         else call print(.('RW$'));
132   2         call printb;
133   2         if not display$attributes then do;
135   3           if rol(file$info.name(f$arc-1),1) then
136   3             call print(.('Arcv $'));
                  else
137   3             call print(.('     $'));
138   3         end;
139   2         else do;
140   3           if  rol(file$info.name(f$arc-1),1) then   /* arc bit was on in all */
141   3               call print$char('A');                 /* dir entries          */
142   3           else call printb;
143   3           if rol(file$info.name(0),1) then
144   3               call print$char('1');
145   3           else call printb;
146   3           if rol(file$info.name(1),1) then
147   3               call print$char('2');
148   3           else call printb;
149   3           if rol(file$info.name(2),1) then
150   3               call print$char('3');
151   3           else call printb;
152   3           if rol(file$info.name(3),1) then
153   3               call print$char('4');
154   3           else call printb;
155   3         end;
156   2     end display$file$info;

157   1     display$xfcb$info: procedure;
158   2             if file$info.x$i$adr <> 0 then
159   2             do;
160   3                 call printb;
161   3                 x$i$adr = file$info.x$i$adr;
162   3                 if (xfcb$info.passmode and pm$read) <> 0 then
163   3                     call print(.('Read  $'));
164   3                 else if (xfcb$info.passmode and pm$write) <> 0 then
165   3                     call print(.('Write $'));
166   3                 else if (xfcb$info.passmode and pm$delete) <> 0 then
167   3                     call print(.('Delete$'));
                          else
168   3                     call print(.('None  $'));
```

```
169   3                 call printb;
170   3                 if (xfcb$info.update(0) <> 0 or xfcb$info.update(1) <> 0) then
171   3                     call display$timestamp(.xfcb$info.update);
172   3                 else call print(.('                $'));
173   3                     call printb; call printb;
175   3                 if (xfcb$info.create(0) <> 0 or xfcb$info.create(1) <> 0) then
176   3                      call display$timestamp(.xfcb$info.create(0));
                                                    /* Create/Access */
177   3             end;
178   2       end display$xfcb$info;

179   1       dcl first$title boolean initial (true);

180   1       display$title: procedure;

181   2           if formfeeds then
182   2               call print$char(ff);
183   2           else if not first$title then
184   2               call crlf;
            call print(.('Directory For Drive $'));
186   2          call printchar('A'+ cur$drv); call printchar(':');
188   2          if bdos >= bdos20 then
189   2          do;
190   3              call print(.('  User $'));
191   3              call pdecimal(cur$usr,10,true);
192   3          end;
193   2          call crlf;
194   2          cur$line = 2;
195   2          first$title = false;
196   2       end display$title;

197   1       short$display: procedure (fname$adr);
198   2           dcl fname$adr address;
199   2           if cur$file mod files$per$line = 0 then
200   2               do;
201   3                   if cur$line mod page$len = 0 then
202   3                   do;
203   4                       call crlf;
204   4                       call display$title;
205   4                       call crlf;
206   4                   end;
                        else
207   3                       call crlf;
208   3                   cur$line = cur$line + 1;
209   3                   call printchar(cur$drv + 'A');
210   3               end;
211   2           else call printb;
212   2           call print(.(': $'));
213   2           call printfn(fname$adr);
214   2           cur$file = cur$file + 1;
215   2       end short$display;

216   1       test$att: procedure(char,off,on) boolean;
217   2           dcl (char,off,on) byte;
218   2           if (80h and char) <> 80h and off then
219   2               return(true);
220   2           if (80h and char) = 80h and on then
```

```
221   2             return(true);
222   2         return(false);
223   2     end test$att;

224   1     right$attributes: procedure(name$adr) boolean;
225   2         dcl name$adr address,
                      name based name$adr (1) byte;
226   2         return
                      test$att(name(f$rw-1),find.rw,find.ro) and
                      test$att(name(f$dirsys-1),find.dir,find.sys);
227   2     end right$attributes;

228   1     short$dir: procedure;              /* looks like "DIR" command */
229   2         dcl dcnt byte;
230   2         fcb(f$drvusr) = '?';
231   2         files$per$line = 4;
232   2         dcnt = search$first(.fcb);
233   2         do while dcnt <> 0ffh;
234   3             buf$fcb$adr = shl(dcnt and 11b,5)+.buff;     /* dcnt mod 4 * 32       */
235   3             if (buf$fcb(f$drvusr) and 0f0h) = 0 and
                      buf$fcb(f$ex) = 0 and
                      buf$fcb(f$ex)<= dpb$byte(extmsk$b) then /* no dir labels, xfcbs */
236   3                 if match then
237   3                     if right$attributes(.buf$fcb(f$name)) then
238   3                         call short$display(.buf$fcb(f$name));
239   3             dcnt = search$next;
240   3         end;
241   2     end short$dir;

242   1     dcl (last$plus$one,index) address;

243   1     getnxt$file$info: procedure;      /* set f$i$adr to base file$info on file    */
244   2     dcl right$usr boolean;            /* to be displayed, f$i$adr = 0ffffh if end */
245   2         right$usr = false;
246   2         if sorted then
247   2         do; index = index + 1;
249   3             f$i$adr = mult23(f$i$indices(index));
250   3             do while file$info.usr <> cur$usr and index <> filesfound;
251   4                index = index + 1;
252   4                f$i$adr = mult23(f$i$indices(index));
253   4             end;
254   3             if index = files$found then
255   3                f$i$adr = last$plus$one;               /* no more files */
256   3         end;
                  else /* not sorted display in order found in directory */
257   2         do;  /* use last$plus$one to avoid wrap around problems */
258   3             f$i$adr = f$i$adr + size(file$info);
259   3             do while file$info.usr <> cur$usr and f$i$adr <> last$plus$one;
260   4                 f$i$adr = f$i$adr + size(file$info);
261   4             end;
262   3         end;
263   2     end getnxt$file$info;

264   1     size$display: procedure;
265   2         if (format and form$size) <> 0 then
266   2             files$per$line = 3;
267   2         else files$per$line = 4;
```

```
268   2          do while f$i$adr <> last$plus$one;
269   3          if ((file$info.x$i$adr <> 0 and find.xfcb) or
                      file$info.x$i$adr  = 0 and find.nonxfcb) and
                      right$attributes(.file$info.name(0)) then
270   3              do;
271   4                  call add$totals;
272   4                  call short$display(.file$info.name(0));
273   4                  call pdecimal(file$info.kbytes,10000,true);
274   4                  call print(.('k$'));
275   4              end;
276   3              call getnxt$file$info;
277   3          end;
278   2      end size$display;

279   1      display$no$dirlabel: procedure;
280   2          files$per$line = 2;
281   2          do while f$i$adr <> last$plus$one;              /* Do all valid files */
282   3             if ((file$info.x$i$adr <> 0 and find.xfcb) or
                       (file$info.x$i$adr = 0   and find.nonxfcb)) and
                       right$attributes(.file$info.name(0))  then
283   3             do;
284   4             if cur$file mod files$per$line <> 0 then call printb;
286   4             else do;                        /* need a new line */
287   5                  if cur$line mod page$len <> 0 then do;           /* just crlf */
289   6                     call crlf;
290   6                     cur$line = cur$line + 1;
291   6                     end;
292   5               else do;                       /* print header */
293   6                     call crlf;
294   6                     call display$title; call crlf;
296   6                     call print(.hdr);       call printb; call print(.hdr);
299   6                     call crlf;
300   6                     call print(.hdr$bars); call printb; call print(.hdr$bars);
303   6                     call crlf;
304   6                     cur$line = cur$line + 3;
305   6                     end;
306   5             end;
307   4                 call display$file$info;
308   4                 cur$file = cur$file + 1;
309   4                 call add$totals;
310   4             end;
311   3             call getnxt$file$info;
312   3          end;/* do loop */
313   2      end display$no$dirlabel;

314   1      display$with$dirlabel: procedure;
315   2          files$per$line = 1;
316   2          do while f$i$adr <> last$plus$one;          /* Display the file info */
317   3            if ((file$info.x$i$adr <> 0 and find.xfcb) or
                       (file$info.x$i$adr  = 0 and find.nonxfcb)) and
                       right$attributes(.file$info.name(0)) then
318   3             do;
319   4               if cur$line mod page$len = 0 then do;        /* display the header */
321   5                  call crlf;
322   5                  call display$title; call crlf;
324   5                  call print(.hdr); call print(.hdr$pu);
326   5                  if (dirlabel and dl$access) <> 0 then
```

```
327   5                   call print(.hdr$access);
328   5               else call print(.hdr$create);
329   5               call crlf;
330   5               call print(.hdr$bars); call print(.hdr$xfcb$bars);
332   5               cur$line = cur$line + 2;
333   5               end;                        /* header display */
334   4             call crlf;
335   4             call display$file$info;       /* display non bdos 3.0 file info    */
336   4             call display$xfcb$info;
337   4             cur$file = cur$file+1; cur$line = cur$line+1;
339   4             call add$totals;
340   4           end;
341   3           call getnxt$file$info;
342   3         end;
343   2       end display$with$dirlabel;


344   1       display$files: procedure public; /* MODULE ENTRY POINT          */
                                               /* display the collected data */
345   2           cur$line, cur$file = 0;      /* force titles and new line  */
346   2           totalkbytes.lword, totalkbytes.hbyte, totalrecs.lword, totalrecs.hbyte =0;
347   2           total$1k$blocks.lword, total$1k$blocks.hbyte = 0;
348   2           f$i$adr = first$f$i$adr - size(file$info);        /* initial if no sort */
349   2           last$plus$one = last$f$i$adr + size(file$info);
350   2           index = 0ffffh;              /* initial if sorted           */
351   2           call getnxt$file$info;       /* base file info record       */

352   2           if format > 2 then
353   2           do;
354   3               call print(.("Illegal Format Value$"));
355   3               call terminate;          /* default could be patched - watch it */
356   3           end;

357   2           do case format;              /* format = */
358   3               call short$dir;                       /* form$short          */
359   3               call size$display;                    /* form$size           */
                                                          /* form = full         */
360   3               if date$opt then do;
362   4                 if ((( dir$label and dl$exists) <> 0 ) and
                          ((( dir$label and dl$access) <> 0 ) or
                          (( dir$label and dl$update) <> 0 ) or
                          (( dir$label and dl$makexfcb) <> 0 ))) then
363   4                   call display$with$dirlabel; /* Timestamping is active! */
364   4                 else do;
365   5                   call print(.("Date and Time Stamping Inactive$"));
366   5                   call terminate;
367   5                 end;
368   4               end;
369   3               else do;    /* No date option; Regular Full display */
370   4                 if (dir$label and dl$exists) <> 0 then
371   4                   call display$with$dirlabel;
                        else
372   4                   call display$no$dirlabel;
373   4               end;
374   3           end;  /* end of case */
375   2           if format <> form$short and cur$file > 0 then          /* print totals */
376   2           do;
```

```
377   3             if cur$line + 4 > page$len and formfeeds then
378   3             do;
379   4                 call printchar(cr);
380   4                 call printchar(ff);                  /* need a new page ? */
381   4             end;
                    else
382   3             do;
383   4                 call crlf;
384   4                 call crlf;
385   4             end;
386   3             call print(.(      'Total Bytes     = $'));
387   3             call p3byte(.total$kbytes,1);          /* 6 digit max */
388   3             call printchar('k');
389   3             call print(.('  Total Records = $'));
390   3             call p3byte(.total$recs,10);       /* 7 digit max */
391   3             call print(.('  Files Found = $'));
392   3             call pdecimal(cur$file,1000,true);     /* 4 digit max */
393   3             call print(.(cr,lf,'Total 1k Blocks = $'));
394   3             call p3byte(.total$1k$blocks,1);            /* 6 digit max */
395   3             call print(.('   Used/Max Dir Entries For Drive $'));
396   3             call print$char('A' + cur$drv);
397   3             call print$char(':'); call printb;
399   3             call pdecimal(used$de,1000,true);
400   3             call print$char('/');
401   3             call pdecimal(dpb$word(dirmax$w) + 1,1000,true);
402   3         end;

403   2         if cur$file = 0 then
404   2         do;
405   3             if message then
406   3             do; call crlf;
408   4                 call display$title;
409   4                 call print(.('File Not Found.',cr,lf,'$'));
410   4             end;
411   3          end;
412   2         else do;
413   3           file$displayed = true;
414   3           if not formfeeds then
415   3             call crlf;
416   3         end;

417   2     end display$files;

418   1     end display;
```

CROSS-REFERENCE LISTING

| DEFN | ADDR | SIZE | NAME, ATTRIBUTES, AND REFERENCES |
|---|---|---|---|
| 4 | 0000H | 2 | A. . . . . WORD PARAMETER 5 |
| 10 | 0000H | 2 | A. . . . . WORD PARAMETER 11 |
| 7 | 0000H | 2 | A. . . . . WORD PARAMETER 8 |
| 58 | 0000H | | ADD3BYTE . . . . . PROCEDURE EXTERNAL(34) STACK=0000H 105 107 |
| 61 | 0000H | | ADD3BYTE3. . . . . PROCEDURE EXTERNAL(35) STACK=0000H 106 |
| 104 | 002BH | 48 | ADDTOTALS. . . . . PROCEDURE STACK=0008H 271 309 339 |
| 14 | 0000H | 1 | BDOS . . . . . BYTE EXTERNAL(9) 186 |
| 15 | | | BDOS20 . . . . . LITERALLY 188 |
| 15 | | | BDOS22 . . . . . LITERALLY |
| 15 | | | BDOS30 . . . . . LITERALLY |
| 83 | | | BLKMAXW. . . . . LITERALLY |
| 83 | | | BLKMSKB. . . . . LITERALLY |
| 83 | | | BLKSHFB. . . . . LITERALLY |
| 2 | | | BOOLEAN. . . . . LITERALLY 17 18 19 25 27 53 72 179 216 224 244 |
| 3 | 0000H | 128 | BUFF . . . . . BYTE ARRAY(128) EXTERNAL(2) 234 |
| 33 | 0000H | 32 | BUFFCB . . . . . BYTE BASED(BUFFCBADR) ARRAY(32) 235 237 238 |
| 33 | 0000H | 2 | BUFFCBADR. . . . . WORD EXTERNAL(21) 33 234 235 237 238 |
| 61 | 0000H | 2 | BYTE3. . . . . WORD PARAMETER 62 |
| 55 | 0000H | 2 | BYTE3ADR . . . . . WORD PARAMETER 56 |
| 58 | 0000H | 2 | BYTE3ADR . . . . . WORD PARAMETER 59 |
| 64 | 0000H | 2 | BYTE3ADR . . . . . WORD PARAMETER 65 |
| 61 | 0000H | 2 | BYTE3ADR . . . . . WORD PARAMETER 62 |
| 15 | | | CCPM86 . . . . . LITERALLY |
| 39 | 0000H | 1 | CHAR . . . . . BYTE PARAMETER 40 |
| 216 | 0008H | 1 | CHAR . . . . . BYTE PARAMETER AUTOMATIC 217 218 220 |
| 94 | 000BH | 1 | CHAR . . . . . BYTE 95 96 97 99 |
| 83 | | | CHKSIZ . . . . . LITERALLY |
| 15 | | | CPM86. . . . . LITERALLY |
| 2 | | | CR . . . . . LITERALLY 379 393 409 |
| 37 | 0000H | 4 | CREATE . . . . . BYTE ARRAY(4) MEMBER(XFCBINFO) 175 176 |
| 47 | 0000H | | CRLF . . . . . PROCEDURE EXTERNAL(30) STACK=0000H 184 193 203 205 207 289 293 295 299 303 321 323 329 334 383 384 407 415 |
| 2 | | | CTRLC. . . . . LITERALLY 99 |
| 13 | 0000H | 1 | CURDRV . . . . . BYTE EXTERNAL(6) 186 209 396 |
| 33 | 0000H | 2 | CURFILE. . . . . WORD 199 214 284 308 337 345 375 392 403 |
| 110 | 0004H | 2 | CURLINE. . . . . WORD 194 201 208 287 290 304 319 332 338 345 377 |
| 13 | 0000H | 1 | CURUSR . . . . . BYTE EXTERNAL(7) 191 250 259 |
| 17 | 0000H | 1 | DATEOPT. . . . . BYTE EXTERNAL(11) 360 |
| 2 | | | DCL. . . . . LITERALLY |
| 229 | 0018H | 1 | DCNT . . . . . BYTE 232 233 234 239 |
| 24 | 0000H | 1 | DIR. . . . . BYTE MEMBER(FIND) 226 |
| 83 | | | DIRBLKW. . . . . LITERALLY |
| 90 | 0000H | 15 | DIRECTCONSOLEIO. . . PROCEDURE BYTE STACK=0008H 95 97 |
| 28 | 0000H | 1 | DIRLABEL . . . . . BYTE EXTERNAL(20) 326 362 370 |
| 31 | | | DIRLABELTYPE . . . . LITERALLY |
| 83 | | | DIRMAXW. . . . . LITERALLY 401 |
| 29 | | | DISKMAPLEN . . . . . LITERALLY |
| 1 | 0000H | | DISPLAY. . . . . PROCEDURE STACK=0000H |
| 18 | 0000H | 1 | DISPLAYATTRIBUTES. . BYTE EXTERNAL(12) 133 |

```
117  0056H  287  DISPLAYFILEINFO. . .  PROCEDURE STACK=0022H                307  335
344  0611H  443  DISPLAYFILES . . . .  PROCEDURE PUBLIC STACK=0614H
279  0480H  182  DISPLAYNODIRLABEL. .  PROCEDURE STACK=0016H                372
 75  0000H       DISPLAYTIMESTAMP . .  PROCEDURE EXTERNAL(40) STACK=0000H          171  176
180  020EH   98  DISPLAYTITLE . . . .  PROCEDURE STACK=000AH                204  294  322  406
314  0566H  171  DISPLAYWITHDIRLABEL.  PROCEDURE STACK=0016H                363  371
157  017AH  148  DISPLAYXFCBINFO. . .  PROCEDURE STACK=0006H                336
 31              DLACCESS . . . . . .  LITERALLY       326  362
 31              DLEXISIS . . . . . .  LITERALLY       362  370
 31              DLMAKEXFCB . . . . .  LITERALLY       362
 31              DLPASSWORD . . . . .  LITERALLY
 31              DLUPDATE . . . . . .  LITERALLY       362
 84  0000H       DPBBYTE. . . . . .  PROCEDURE BYTE EXTERNAL(43) STACK=0000H          235
 87  0000H    1  DPBINDEX . . . . . .  BYTE PARAMETER         88
 84  0000H    1  DPBINDEX . . . . . .  BYTE PARAMETER         85
 87  0000H       DPBWORD. . . . . .  PROCEDURE WORD EXTERNAL(44) STACK=0000H          401
 24  0007H    1  EXCLUDE. . . . . .  BYTE MEMBER(FIND)
 83              EXTMSKB. . . . . .  LITERALLY       235
  4  0000H    1  F. . . . . .  BYTE PARAMETER          5
 10  0000H    1  F. . . . . .  BYTE PARAMETER         11
  7  0000H    1  F. . . . . .  BYTE PARAMETER          8
  2              FALSE. . . . . .  LITERALLY        27  195  222  245
 29              FARC . . . . . .  LITERALLY       135  140
  3  0000H   33  FCB. . . . . .  BYTE ARRAY(33) EXTERNAL(0)          230  232
 73  0002H    2  FCBADR . . . . . .  WORD
 67  0000H    2  FCBADR . . . . . .  WORD PARAMETER         68
 29              FDIRSYS. . . . . .  LITERALLY       125  226
 29              FDISKMAP . . . . .  LITERALLY
 29              FDRVUSR. . . . . .  LITERALLY       230  235
 29              FDRVUSR2 . . . . .  LITERALLY
 29              FEX. . . . . .  LITERALLY       235
  2              FF . . . . . .  LITERALLY       182  380
 33  0000H    2  FIADR. . . . . .  WORD EXTERNAL(22)           36  105  106  107  118  120  123  125  129
                                     135  140  143  146  149  152  158  161  249  250  252  255  258  259
                                     260  268  269  272  273  281  282  316  317  348
 38  0000H    2  FIINDICES. . . . . .  WORD BASED(FIINDICESBASE) ARRAY(1)      249  252
 38  0000H    2  FIINDICESBASE. . . .  WORD EXTERNAL(26)           38  249  252
 27  000AH    1  FILEDISPLAYED. . . .  BYTE PUBLIC INITIAL       413
 36  0000H   23  FILEINFO . . . . . .  STRUCTURE BASED(FIADR)          105  106  107  118  120  123  125  129
                                     135  140  143  146  149  152  158  161  250  258  259  260  269  272
                                     273  282  317  348  349
 20  0000H    2  FILESFOUND . . . . .  WORD EXTERNAL(14)          250  254
109  0016H    1  FILESPERLINE . . . .  BYTE       199  231  266  267  280  284  315
 24  0000H    8  FIND . . . . . .  STRUCTURE EXTERNAL(15)          226  269  292  317
 21              FINDSTRUCTURE. . . .  LITERALLY        24
 34              FINFOSTRUCTURE . . .  LITERALLY        36
 33  0000H    2  FIRSTFIADR . . . . .  WORD EXTERNAL(24)          348
179  0017H    1  FIRSTTITLE . . . . .  BYTE INITIAL          183  195
 29              FNAME. . . . . .  LITERALLY       237  238
 29              FNAME2 . . . . . .  LITERALLY
 49  0000H    2  FNAMEADR . . . . . .  WORD PARAMETER         50
197  0004H    2  FNAMEADR . . . . . .  WORD PARAMETER AUTOMATIC          198  213
 29              FNAMELEN . . . . . .  LITERALLY
  2              FOREVER. . . . . .  LITERALLY
 25  0000H    1  FORMAT . . . . . .  BYTE EXTERNAL(16)          265  352  357  375
 25  0000H    1  FORMFEEDS. . . . . .  BYTE EXTERNAL(19)          181  377  414
 26              FORMFULL . . . . . .  LITERALLY
```

```
  26                FORMSHORT. . . . . LITERALLY       375
  26                FORMSIZE . . . . . LITERALLY       265
  29                FRC. . . . . LITERALLY
  29                FRREC. . . . . LITERALLY
  29                FRRECO . . . . . LITERALLY
  29                FRW. . . . . LITERALLY       129  226
  29                FS1. . . . . LITERALLY
  29                FTYPE. . . . . LITERALLY
  29                FTYPE2 . . . . . LITERALLY
  29                FTYPELEN . . . . . LITERALLY
 243  03A9H   141   GETNXTFILEINFO . . . PROCEDURE STACK=0006H          276  311  341  351
 102  000CH     1   GLOBALLINECOUNT. . . BYTE INITIAL
  36  0013H     2   HASHLINK . . . . . WORD MEMBER(FILEINFO)
 103  0002H     1   HBYTE. . . . . BYTE MEMBER(TOTAL1KBLOCKS)          347
 103  0002H     1   HBYTE. . . . . BYTE MEMBER(TOTALKBYTES)       346
 103  0002H     1   HBYTE. . . . . BYTE MEMBER(TOTALRECS)         346
 111  0000H    40   HDR. . . . . BYTE ARRAY(40) DATA       296  298  324
 115  008EH    17   HDRACCESS. . . . . BYTE ARRAY(17) DATA       327
 112  0028H    40   HDRBARS. . . . . BYTE ARRAY(40) DATA       300  302  330
 116  009FH    17   HDRCREATE. . . . . BYTE ARRAY(17) DATA       328
 113  0050H    23   HDRPU. . . . . BYTE ARRAY(23) DATA       325
 114  0067H    39   HDRXFCBBARS. . . . . BYTE ARRAY(39) DATA       331
  80  0000H     2   INDEX. . . . . WORD PARAMETER       81
 242  0008H     2   INDEX. . . . . WORD       248  249  250  251  252  254  350
  36  000EH     2   KBYTES . . . . . WORD MEMBER(FILEINFO)         105  120  273
  33  0000H     2   LASTFIADR. . . . . WORD EXTERNAL(23)       349
 242  0006H     2   LASTPLUSONE. . . . . WORD       255  259  268  281  316  349
   2                LF . . . . . LITERALLY       393  409
   2                LIT. . . . . LITERALLY         2   15   21   23   26   29   30   31   32   34   35
                      83
 103  0000H     2   LWORD. . . . . WORD MEMBER(TOTAL1KBLOCKS)          347
 103  0000H     2   LWORD. . . . . WORD MEMBER(TOTALKBYTES)       346
 103  0000H     2   LWORD. . . . . WORD MEMBER(TOTALRECS)         346
  72  0000H         MATCH. . . . . PROCEDURE BYTE EXTERNAL(39) STACK=0000H       236
   3  0000H     2   MAXB . . . . . WORD EXTERNAL(1)
  22                MAXSEARCHFILES . . . LITERALLY
  25  0000H     1   MESSAGE. . . . . BYTE EXTERNAL(18)       405
   4  0000H         MON1 . . . . . PROCEDURE EXTERNAL(3) STACK=0000H
   7  0000H         MON2 . . . . . PROCEDURE BYTE EXTERNAL(4) STACK=0000H        91
  10  0000H         MON3 . . . . . PROCEDURE WORD EXTERNAL(5) STACK=0000H
  15                MPM. . . . . LITERALLY
  15                MPM86. . . . . LITERALLY
  80  0000H         MULT23 . . . . . PROCEDURE WORD EXTERNAL(42) STACK=0000H       249  252
 225  0000H     1   NAME . . . . . BYTE BASED(NAMEADR) ARRAY(1)       226
  36  0001H     8   NAME . . . . . BYTE ARRAY(3) MEMBER(FILEINFO)       118  125  129  135  140  143  146
                      149  152  269  272  282  317
 224  0004H     2   NAMEADR. . . . . WORD PARAMETER AUTOMATIC       225  226
  24  0006H     1   NONXFCB. . . . . BYTE MEMBER(FIND)       269  282  317
   2                NOPAGEMODEOFFSET . . LITERALLY
 216  0006H     1   OFF. . . . . BYTE PARAMETER AUTOMATIC       217  218
  83                OFFSETW. . . . . LITERALLY
 216  0004H     1   ON . . . . . BYTE PARAMETER AUTOMATIC       217  220
  36  000CH     2   ONEKBLOCKS . . . . . WORD MEMBER(FILEINFO)         107
  14  0000H     1   OS . . . . . BYTE EXTERNAL(8)
  55  0000H         P3BYTE . . . . . PROCEDURE EXTERNAL(34) STACK=0000H       123  387  390  394
  25  0000H     2   PAGELEN. . . . . WORD EXTERNAL(17)       201  287  319  377
   2                PAGELENOFFSET. . . . LITERALLY
```

```
  24  0004H     1  PASS . . . . . .  BYTE MEMBER(FIND)
  30             PASSLEN. . . . . .  LITERALLY
  37  0008H     1  PASSMODE . . . . .  BYTE MEMBER(XFCBINFO)          162  164  166
  52  0000H        PDECIMAL . . . . .  PROCEDURE EXTERNAL(32) STACK=0000H      120  191  273  392  399  401
  32             PMDELETE . . . . .  LITERALLY      166
  32             PMREAD . . . . . .  LITERALLY      162
  32             PMWRITE. . . . . .  LITERALLY      164
  55  0000H     2  PREC . . . . . .  WORD PARAMETER       56
  52  0000H     2  PREC . . . . . .  WORD PARAMETER       53
  42  0000H        PRINT. . . . . .  PROCEDURE EXTERNAL(23) STACK=0000H      126  127  130  131  136  137
                                      163  165  167  168  172  185  190  212  274  296  298  300  302  324
                                      325  327  328  330  331  354  365  386  389  391  393  395  409
  45  0000H        PRINTB . . . . . .  PROCEDURE EXTERNAL(29) STACK=0000H      119  122  124  128  132  142
                                      145  148  151  154  160  169  173  174  211  285  287  301  398
  39  0000H        PRINTCHAR. . . . .  PROCEDURE EXTERNAL(27) STACK=0000H      121  141  144  147  150  153
                                      182  186  187  209  379  380  388  396  397  400
  49  0000H        PRINTFN. . . . . .  PROCEDURE EXTERNAL(31) STACK=0000H      113  213
  36  0012H     1  RECSHBYTE. . . . .  BYTE MEMBER(FILEINFO)
  36  0010H     2  RECSLWORD. . . . .  WORD MEMBER(FILEINFO)          106  123
 224  02EEH    49  RIGHTATTRIBUTES. . .  PROCEDURE BYTE STACK=0010H          237  269  282  317
 244  0019H     1  RIGHTUSR . . . . .  BYTE       245
  24  0002H     1  RO . . . . . .  BYTE MEMBER(FIND)          226
                   ROL. . . . . .  BUILTIN          125  129  135  140  143  146  149  152
  24  0003H     1  RW . . . . . .  BYTE MEMBER(FIND)          226
  67  0000H        SEARCHFIRST. . . . .  PROCEDURE BYTE EXTERNAL(37) STACK=0000H      232
  70  0000H        SEARCHNEXT . . . . .  PROCEDURE BYTE EXTERNAL(38) STACK=0000H      239
  23             SEARCHSTRUCTURE. . .  LITERALLY
   2             SECTORLEN. . . . . .  LITERALLY
                   SHL. . . . . .  BUILTIN          234
 228  031FH   138  SHORTDIR . . . . . .  PROCEDURE STACK=0014H          358
 197  0270H    81  SHORTDISPLAY . . . .  PROCEDURE STACK=0010H          238  272
  64  0000H        SHR3BYTE . . . . . .  PROCEDURE EXTERNAL(36) STACK=0000H
                   SIZE . . . . . .  BUILTIN          258  260  348  349
 264  0436H   122  SIZEDISPLAY. . . . .  PROCEDURE STACK=0016H          359
  19  0000H     1  SORTED . . . . . .  BYTE EXTERNAL(13)          246
  83             SPTW . . . . . .  LITERALLY
  42  0000H     2  STRINGADR. . . . . .  WORD PARAMETER       43
  24  0001H     1  SYS. . . . . .  BYTE MEMBER(FIND)          226
   2             TAB. . . . . .  LITERALLY
  78  0000H        TERMINATE. . . . . .  PROCEDURE EXTERNAL(41) STACK=0000H      100  355  366
 216  02C1H    45  TESTATT. . . . . .  PROCEDURE BYTE STACK=0008H          226
 103  0013H     3  TOTAL1KBLOCKS. . . .  STRUCTURE        107  347  394
 103  000DH     3  TOTALKBYTES. . . . .  STRUCTURE        105  346  387
 103  0010H     3  TOTALRECS. . . . . .  STRUCTURE        106  346  390
   2             TRUE . . . . . .  LITERALLY        120  179  191  219  221  273  392  399  401  413
  75  0000H     2  TSADR. . . . . .  WORD PARAMETER       76
  36  0009H     3  TYPE . . . . . .  BYTE ARRAY(3) MEMBER(FILEINFO)
  37  0004H     4  UPDATE . . . . . .  BYTE ARRAY(4) MEMBER(XFCBINFO)      170  171
  16  0000H     2  USEDDE . . . . . .  WORD EXTERNAL(10)          399
  36  0000H     1  USR. . . . . .  BYTE MEMBER(FILEINFO)          250  259
  52  0000H     2  V. . . . . .  WORD PARAMETER       53
  93  000FH    28  WAITKEYPRESS . . . .  PROCEDURE STACK=000CH
  58  0000H     2  WORDAMT. . . . . .  WORD PARAMETER       59
  24  0005H     1  XFCB . . . . . .  BYTE MEMBER(FIND)          269  282  317
  37  0000H     9  XFCBINFO . . . . . .  STRUCTURE BASED(XIADR)          162  164  166  170  171  175  176
  30             XFCBTYPE . . . . . .  LITERALLY
  30             XFCREATE . . . . . .  LITERALLY
```

```
30                  XFPASS . . . . . .   LITERALLY
30                  XFPASSMODE . . . . .  LITERALLY
30                  XFUPDATE . . . . . .  LITERALLY
37  0000H      2    XIADR. . . . . .     WORD EXTERNAL(25)          37  161  162  164  166  170  171  175  176
36  0015H      2    XIADR. . . . . .     WORD MEMBER(FILEINFO)          158  161  269  282  317
35                  XINFOSTRUCTURE . . .  LITERALLY          37
52  0000H      1    ZEROSUP. . . . . .   BYTE PARAMETER           53

MODULE INFORMATION:

     CODE AREA SIZE     = 07CCH    1996D
     CONSTANT AREA SIZE = 01CDH     461D
     VARIABLE AREA SIZE = 001AH      26D
     MAXIMUM STACK SIZE = 001AH      26D
     666 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE DPB86
OBJECT MODULE PLACED IN DPB86
COMPILER INVOKED BY:  :F0: DPB86.PLM DEBUG OBJECT(DPB86) OPTIMIZE(3) XREF


                $compact
                $title ('SDIR 8086 - Get Disk Parameters')
  1             dpb86:
                do;
                        /* the purpose of this module is to allow independence */
                        /* of processor, i.e., 8080 or 8086                    */

                $include (comlit.lit)
                =
  2   1         =     declare
                =             lit                literally          'literally',
                =             dcl                lit                'declare',
                =             true               lit                '0ffh',
                =             false              lit                '0',
                =             boolean            lit                'byte',
                =             forever            lit                'while true',
                =             cr                 lit                '13',
                =             lf                 lit                '10',
                =             tab                lit                '9',
                =             ctrlc              lit                '3',
                =             ff                 lit                '12',
                =             page$len$offset    lit                '1ch',
                =             nopage$mode$offset lit                '20h',
                =             sectorlen          lit                '128';

                /* function call 32 in 2.0 or later BDOS, returns the address of the disk
                parameter block for the currently selected disk, which consists of:
                        spt                     (2 bytes) number of sectors per track
                        blkshf                  (1 byte)  block size = shl(double(128),blkshf)
                        blkmsk                  (1 byte)  sector# and blkmsk = block number
                        extmsk                  (1 byte)  logical/physical extents
                        blkmax                  (2 bytes) max alloc number
                        dirmax                  (2 bytes) size of directory-1
                        dirblk                  (2 bytes) reservation bits for directory
                        chksiz                  (2 bytes) size of checksum vector
                        offset                  (2 bytes) offset for operating system
                */

                $include(dpb.lit)
                =
                =     /* indices into disk parameter block, used as parameters to dpb procedure */
                =
  3   1         =     dcl         spt$w         lit         '0',
                =                 blkshf$b      lit         '2',
                =                 blkmsk$b      lit         '3',
                =                 extmsk$b      lit         '4',
                =                 blkmax$w      lit         '5',
                =                 dirmax$w      lit         '7',
                =                 dirblk$w      lit         '9',
                =                 chksiz        lit         '11',
                =                 offset$w      lit         '13';
```

=

```
 4   1      declare k$per$block byte public;
 5   1      declare dpb$base pointer;
 6   1      declare dpb$array based dpb$base (15) byte;

 7   1      mon4: procedure (f,a) pointer external;
 8   2          dcl f byte, a address;
 9   2      end mon4;

10   1      dcl get$dpb lit '31';

11   1      dpb$byte: procedure(param) byte public;
12   2          dcl param byte;
13   2          return(dpb$array(param));
14   2      end dpb$byte;

15   1      dpb$word: procedure(param) address public;
16   2          dcl param byte;
17   2          return(dpb$array(param) + shl(double(dpb$array(param+1)),8));
18   2      end dpb$word;

19   1      base$dpb: procedure public;
20   2          dpb$base = mon4(get$dpb,0);
21   2          k$per$block = shr(dpb$byte(blkmsk$b)+1 ,3);
22   2      end base$dpb;

23   1      end dpb86;
```

CROSS-REFERENCE LISTING

```
 DEFN  ADDR   SIZE  NAME, ATTRIBUTES, AND REFERENCES
-----  ------ -----  ------------------------------

    7  0000H     2  A. . . . . .    WORD PARAMETER          8
   19  003AH    58  BASEDPB. . . .  PROCEDURE PUBLIC STACK=0008H
    3             BLKMAXW. . . . .  LITERALLY
    3             BLKMSKB. . . . .  LITERALLY          21
    3             BLKSHFB. . . . .  LITERALLY
    2             BOOLEAN. . . . .  LITERALLY
    3             CHKSIZ . . . . .  LITERALLY
    2             CR . . . . . .    LITERALLY
    2             CTRLC. . . . . .  LITERALLY
    2             DCL. . . . . .    LITERALLY
    3             DIRBLKW. . . . .  LITERALLY
    3             DIRMAXW. . . . .  LITERALLY
                  DOUBLE . . . . .  BUILTIN            17
    1  0002H      DPB86. . . . . .  PROCEDURE STACK=0000H
    6  0000H     15  DPBARRAY . . .  BYTE BASED(DPBBASE) ARRAY(15)          13   17
    5  0000H      4  DPBBASE. . . .  POINTER             6   13   17   20
   11  0002H     21  DPBBYTE. . . .  PROCEDURE BYTE PUBLIC STACK=0004H          21
   15  0017H     35  DPBWORD. . . .  PROCEDURE WORD PUBLIC STACK=0004H
    3             EXTMSKB. . . . .  LITERALLY
    7  0000H      1  F. . . . . .    BYTE PARAMETER          8
    2             FALSE. . . . . .  LITERALLY
    2             FF . . . . . .    LITERALLY
    2             FOREVER. . . . .  LITERALLY
   10             GETDPB . . . . .  LITERALLY          20
    4  0004H      1  KPERBLOCK. . .  BYTE PUBLIC            21
    2             LF . . . . . .    LITERALLY
    2             LIT. . . . . .    LITERALLY           2    3   10
    7  0000H         MON4 . . . . .  PROCEDURE POINTER EXTERNAL(0) STACK=0000H          20
    2             NOPAGEMODEOFFSET  LITERALLY
    3             OFFSETW. . . . .  LITERALLY
    2             PAGELENOFFSET. .  LITERALLY
   15  0004H      1  PARAM. . . . .  BYTE PARAMETER AUTOMATIC        16   17
   11  0004H      1  PARAM. . . . .  BYTE PARAMETER AUTOMATIC        12   13
    2             SECTORLEN. . . .  LITERALLY
                  SHL. . . . . .    BUILTIN            17
                  SHR. . . . . .    BUILTIN            21
    3             SPTW . . . . . .  LITERALLY
    2             TAB. . . . . .    LITERALLY
    2             TRUE . . . . . .  LITERALLY
```



MODULE INFORMATION:

```
CODE AREA SIZE     = 0060H     96D
CONSTANT AREA SIZE = 0000H      0D
VARIABLE AREA SIZE = 0005H      5D
MAXIMUM STACK SIZE = 0008H      8D
79 LINES READ
```

0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION

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


                $title('SDIR - Utility Routines')
  1             utility:
                do;

                /* Utility Module for SDIR */

                $include(comlit.lit)
          =
  2     1   =   declare
          =             lit                 literally           'literally',
          =             dcl                 lit                 'declare',
          =             true                lit                 '0ffh',
          =             false               lit                 '0',
          =             boolean             lit                 'byte',
          =             forever             lit                 'while true',
          =             cr                  lit                 '13',
          =             lf                  lit                 '10',
          =             tab                 lit                 '9',
          =             ctrlc               lit                 '3',
          =             ff                  lit                 '12',
          =             page$len$offset     lit                 '1ch',
          =             nopage$mode$offset  lit                 '2Ch',
          =             sectorlen           lit                 '128';

                /* -------- arithmetic functions -------- */

  3     1       add3byte: procedure(byte3adr,num) public;
  4     2             dcl (byte3adr,num) address,
                          b3 based byte3adr structure (
                          lword address,
                          hbyte byte),
                          temp address;

  5     2             temp = b3.lword;
  6     2             if (b3.lword := b3.lword + num) < temp then             /* overflow */
  7     2                 b3.hbyte = b3.hbyte + 1;
  8     2       end add3byte;

                        /* add three byte number to 3 byte value structure */
  9     1       add3byte3: procedure(totalb,numb) public;
 10     2             dcl (totalb,numb) address,
                          num based numb structure (
                          lword address,
                          hbyte byte),
                          total based totalb structure (
                          lword address,
                          hbyte byte);

 11     2             call add3byte(totalb,num.lword);
 12     2             total.hbyte = num.hbyte + total.hbyte;
```

```
13   2      end add3byte3;

                                                    /* divide 3 byte value by 8 */
14   1      shr3byte: procedure(byte3adr) public;
15   2          dcl byte3adr address,
                   b3 based byte3adr structure (
                   lword address,
                   hbyte byte),
                   temp1 based byte3adr (2) byte,
                   temp2 byte;

16   2              temp2  = ror(b3.hbyte,3) and 11100000b;  /* get 3 bits          */
17   2              b3.hbyte = shr(b3.hbyte,3);
18   2              b3.lword = shr(b3.lword,3);
19   2              temp1(1) = temp1(1) or temp2;            /* or in 3 bits from hbyte */
20   2      end shr3byte;


            /* ------- print routines -------- */

21   1      mon1: procedure(f,a) external;
22   2          declare f byte, a address;
23   2      end mon1;

            /*break: procedure external;
            end break;*/

            $include(fcb.lit)
        =
24   1   =  declare
        =      f$drvusr          lit '0',      /* drive/user byte              */
        =      f$name            lit '1',      /* file name                    */
        =      f$namelen         lit '8',      /* file name length             */
        =      f$type            lit '9',      /* file type field              */
        =      f$typelen         lit '3',      /* type length                  */
        =      f$rw              lit '9',      /* high bit is R/W attribute    */
        =      f$dirsys          lit '10',     /* high bit is dir/sys attribute */
        =      f$arc             lit '11',     /* high bit is archive attribute */
        =      f$ex              lit '12',     /* extent                       */
        =      f$s1              lit '13',     /* module byte                  */
        =      f$rc              lit '15',     /* record count                 */
        =      f$diskmap         lit '16',     /* file disk map                */
        =      diskmaplen        lit '16',     /* disk map length              */
        =      f$drvusr2         lit '16',     /* fcb2                         */
        =      f$name2           lit '17',
        =      f$type2           lit '25',
        =      f$rrec            lit '33',     /* random record                */
        =      f$rreco           lit '35';     /*   "      "    overflow        */
        =

            /* BDOS calls */

25   1      print$char: procedure(char) public;
26   2          declare char byte;
27   2          call mon1(2,char);
28   2      end print$char;
```

```
 29  1      print: procedure(string$adr) public;
 30  2          dcl string$adr address;
 31  2          call mon1(9,string$adr);
 32  2      end print;

 33  1      printb: procedure public;
 34  2          call print$char(' ');
 35  2      end printb;

 36  1      crlf: procedure public;
 37  2          call print$char(cr);
 38  2          call print$char(lf);
 39  2      end crlf;

 40  1      printfn: procedure(fname$adr) public;
 41  2          dcl fname$adr address,
                  file$name based fname$adr (1) byte,
                  i byte;                                  /* <filename> ' ' <filetype> */

 42  2          do i = 0 to f$namelen - 1;
 43  3              call printchar(file$name(i) and 7fh);
 44  3          end;
 45  2          call printchar(' ');
 46  2          do i = f$namelen to f$namelen + f$typelen - 1;
 47  3              call printchar(file$name(i) and 7fh);
 48  3          end;
 49  2      end printfn;

 50  1      pdecimal: procedure(v,prec,zerosup) public;
                                  /* print value v, field size = (log10 prec) + 1  */
                                  /* with leading zero suppression if zerosup = true */
 51  2          declare v address,                           /* value to print           */
                      prec address,                        /* precision                */
                      zerosup boolean,                     /* zero suppression flag    */
                      d byte;                              /* current decimal digit    */

 52  2          do while prec <> 0;
 53  3              d = v / prec;                            /* get next digit           */
 54  3              v = v mod prec;                          /* get remainder back to v  */
 55  3              prec = prec / 10;                        /* ready for next digit     */
 56  3              if prec <> 0 and zerosup and d = 0 then
 57  3                  call printb;
                     else
 58  3                  do;
 59  4                      zerosup = false;
 60  4                      call printchar('0'+d);
 61  4                  end;
 62  3              end;
 63  2      end pdecimal;

 64  1      p3byte: procedure(byte3adr,prec) public;
                                      /* print 3 byte value with 0 suppression */
 65  2            dcl byte3adr address,        /* assume high order bit is < 10          */
                     prec address,
                     b3 based byte3adr structure (
                     lword address,
                     hbyte byte),
```

```
                    i byte;

                                                /* prec = 1 for 6 chars, 2 for 7 */
 66   2             if b3.hbyte <> 0 then
 67   2             do;
 68   3                 call pdecimal(b3.hbyte,prec,true);  /* 3 for 8 chars printed      */
 69   3                 call pdecimal(b3.lword,10000,false);
 70   3             end;
                     else
 71   2             do;
 72   3                 i = 1;
 73   3                 do while i <= prec;
 74   4                     call printb;
 75   4                     i = i * 10;
 76   4                 end;
 77   3                 call pdecimal(b3.lword,10000,true);
 78   3             end;
 79   2       end p3byte;

 80   1       end utility;
```

CROSS-REFERENCE LISTING

| DEFN | ADDR | SIZE | NAME, ATTRIBUTES, AND REFERENCES | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 0000H | 2 | A. . . . . | WORD PARAMETER | 22 | | | | | |
| 3 | 0000H | 29 | ADD3BYTE . . . . . | PROCEDURE PUBLIC STACK=0006H | | 11 | | | | |
| 9 | 001DH | 30 | ADD3BYTE3. . . . . | PROCEDURE PUBLIC STACK=000EH | | | | | | |
| 4 | 0000H | 3 | B3 . . . . . | STRUCTURE BASED(BYTE3ADR) | 5 | 6 | 7 | | | |
| 65 | 0000H | 3 | B3 . . . . . | STRUCTURE BASED(BYTE3ADR) | 66 | 68 | 69 | 77 | | |
| 15 | 0000H | 3 | B3 . . . . . | STRUCTURE BASED(BYTE3ADR) | 16 | 17 | 18 | | | |
| 2 | | | BOOLEAN. . . . . | LITERALLY | 51 | | | | | |
| 14 | 0004H | 2 | BYTE3ADR . . . . . | WORD PARAMETER AUTOMATIC | 15 | 16 | 17 | 18 | 19 | |
| 3 | 0006H | 2 | BYTE3ADR . . . . . | WORD PARAMETER AUTOMATIC | 4 | 5 | 6 | 7 | | |
| 64 | 0006H | 2 | BYTE3ADR . . . . . | WORD PARAMETER AUTOMATIC | 65 | 66 | 68 | 69 | 77 | |
| 25 | 0004H | 1 | CHAR . . . . . | BYTE PARAMETER AUTOMATIC | 26 | 27 | | | | |
| 2 | | | CR . . . . . | LITERALLY | 37 | | | | | |
| 36 | 0087H | 17 | CRLF . . . . . | PROCEDURE PUBLIC STACK=000EH | | | | | | |
| 2 | | | CTRLC. . . . . | LITERALLY | | | | | | |
| 51 | 0004H | 1 | D. . . . . | BYTE | 53 | 56 | 60 | | | |
| 2 | | | DCL. . . . . | LITERALLY | | | | | | |
| 24 | | | DISKMAPLEN . . . . | LITERALLY | | | | | | |
| 21 | 0000H | 1 | F. . . . . | BYTE PARAMETER | 22 | | | | | |
| 2 | | | FALSE. . . . . | LITERALLY | 59 | 69 | | | | |
| 24 | | | FARC . . . . . | LITERALLY | | | | | | |
| 24 | | | FDIRSYS. . . . . | LITERALLY | | | | | | |
| 24 | | | FDISKMAP . . . . | LITERALLY | | | | | | |
| 24 | | | FDRVUSR. . . . . | LITERALLY | | | | | | |
| 24 | | | FDRVUSR2 . . . . | LITERALLY | | | | | | |
| 24 | | | FEX. . . . . | LITERALLY | | | | | | |
| 2 | | | FF . . . . . | LITERALLY | | | | | | |
| 41 | 0000H | 1 | FILENAME . . . . | BYTE BASED(FNAMEADR) ARRAY(1) | | 43 | 47 | | | |
| 24 | | | FNAME. . . . . | LITERALLY | | | | | | |
| 24 | | | FNAME2 . . . . . | LITERALLY | | | | | | |
| 40 | 0004H | 2 | FNAMEADR . . . . | WORD PARAMETER AUTOMATIC | 41 | 43 | 47 | | | |
| 24 | | | FNAMELEN . . . . | LITERALLY | 42 | 46 | | | | |
| 2 | | | FOREVER. . . . . | LITERALLY | | | | | | |
| 24 | | | FRC. . . . . | LITERALLY | | | | | | |
| 24 | | | FRREC. . . . . | LITERALLY | | | | | | |
| 24 | | | FRRECO . . . . . | LITERALLY | | | | | | |
| 24 | | | FRW. . . . . | LITERALLY | | | | | | |
| 24 | | | FS1. . . . . | LITERALLY | | | | | | |
| 24 | | | FTYPE. . . . . | LITERALLY | | | | | | |
| 24 | | | FTYPE2 . . . . . | LITERALLY | | | | | | |
| 24 | | | FTYPELEN . . . . | LITERALLY | 46 | | | | | |
| 65 | 0002H | 1 | HBYTE. . . . . | BYTE MEMBER(B3) | | 66 | 68 | | | |
| 15 | 0002H | 1 | HBYTE. . . . . | BYTE MEMBER(B3) | | 16 | 17 | | | |
| 10 | 0002H | 1 | HBYTE. . . . . | BYTE MEMBER(TOTAL) | | | 12 | | | |
| 10 | 0002H | 1 | HBYTE. . . . . | BYTE MEMBER(NUM) | | | 12 | | | |
| 4 | 0002H | 1 | HBYTE. . . . . | BYTE MEMBER(B3) | | 7 | | | | |
| 65 | 0005H | 1 | I. . . . . | BYTE | 72 | 73 | 75 | | | |
| 41 | 0003H | 1 | I. . . . . | BYTE | 42 | 43 | 46 | 47 | | |
| 2 | | | LF . . . . . | LITERALLY | 38 | | | | | |
| 2 | | | LIT. . . . . | LITERALLY | 2 | 24 | | | | |

```
 15  0000H     2  LWORD. . . . . .  WORD MEMBER(B3)        13
 10  0000H     2  LWORD. . . . . .  WORD MEMBER(TOTAL)
 10  0000H     2  LWORD. . . . . .  WORD MEMBER(NUM)           11
  4  0000H     2  LWORD. . . . . .  WORD MEMBER(B3)         5    6
 65  0000H     2  LWORD. . . . . .  WORD MEMBER(B3)        69   77
 21  0000H        MON1 . . . . . .  PROCEDURE EXTERNAL(0) STACK=0000H          27   31
  2                NOPAGEMODEOFFSET .  LITERALLY
 10  0000H     3  NUM. . . . . .  STRUCTURE BASED(NUMB)            11   12
  3  0004H     2  NUM. . . . . .  WORD PARAMETER AUTOMATIC          4    6
  9  0004H     2  NUMB . . . . . .  WORD PARAMETER AUTOMATIC       10   11   12
 64  0143H    89  P3BYTE . . . . . .  PROCEDURE PUBLIC STACK=0020H
  2                PAGELENOFFSET. . .  LITERALLY
 50  00EDH    86  PDECIMAL . . . . .  PROCEDURE PUBLIC STACK=0018H          68   69   77
 64  0004H     2  PREC . . . . . .  WORD PARAMETER AUTOMATIC       65   68   73
 50  0006H     2  PREC . . . . . .  WORD PARAMETER AUTOMATIC       51   52   53   54   55   56
 29  006CH    16  PRINT. . . . . .  PROCEDURE PUBLIC STACK=00CAH
 33  007CH    11  PRINTB . . . . . .  PROCEDURE PUBLIC STACK=00EH       57   74
 25  0059H    19  PRINTCHAR. . . . .  PROCEDURE PUBLIC STACK=000AH       34   37   38   43   45   47   60
 40  0098H    85  PRINTFN. . . . . .  PROCEDURE PUBLIC STACK=0010H
                  ROR. . . . . .  BUILTIN          16
  2                SECTORLEN. . . . .  LITERALLY
                   SHR. . . . . .  BUILTIN          17   18
 14  003BH    30  SHR3BYTE . . . . .  PROCEDURE PUBLIC STACK=0004H
 29  0004H     2  STRINGADR. . . . .  WORD PARAMETER AUTOMATIC       30   31
  2                TAB. . . . . .  LITERALLY
  4  0000H     2  TEMP . . . . . .  WORD          5    6
 15  0000H     2  TEMP1. . . . . .  BYTE BASED(BYTE3ADR) ARRAY(2)          19
 15  0002H     1  TEMP2. . . . . .  BYTE          16   19
 10  0000H     3  TOTAL. . . . . .  STRUCTURE BASED(TOTALB)       12
  9  0006H     2  TOTALB . . . . . .  WORD PARAMETER AUTOMATIC      10   11   12
  2                TRUE . . . . . .  LITERALLY          68   77
  1  0000H        UTILITY. . . . . .  PROCEDURE STACK=0000H
 50  0008H     2  V. . . . . .  WORD PARAMETER AUTOMATIC       51   53   54
 50  0004H     1  ZEROSUP. . . . . .  BYTE PARAMETER AUTOMATIC       51   56   59
```



MODULE INFORMATION:

```
    CODE AREA SIZE     = 019CH    4120
    CONSTANT AREA SIZE = 0000H      0D
    VARIABLE AREA SIZE = 0006H      6D
    MAXIMUM STACK SIZE = 0020H     32D
    185 LINES READ
    0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

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


                $title('SDIR - Display Time Stamps')
  1             timestamp:
                do;
                      /* Display time stamp module for extended directory */
                      /* Time & Date ASCII Conversion Code              */
                      /* From MP/M 1.1 TOD program                    */

                $include(comlit.lit)
                =
  2   1         =   declare
                =         lit                 literally         'literally',
                =         dcl                 lit               'declare',
                =         true                lit               '0ffh',
                =         false               lit               '0',
                =         boolean             lit               'byte',
                =         forever             lit               'while true',
                =         cr                  lit               '13',
                =         lf                  lit               '10',
                =         tab                 lit               '9',
                =         ctrlc               lit               '3',
                =         ff                  lit               '12',
                =         page$len$offset     lit               '1ch',
                =         nopage$mode$offset  lit               '2Ch',
                =         sectorlen           lit               '128';

  3   1         print$char: procedure (char) external;
  4   2             declare char byte;
  5   2         end print$char;

  6   1         terminate: procedure external;
  7   2         end terminate;

  8   1         declare tod$adr address;
  9   1         declare tod based tod$adr structure (
                  opcode byte,
                  date address,
                  hrs byte,
                  min byte,
                  sec byte,
                  ASCII (21) byte );

 10   1         declare string$adr address;
 11   1         declare string based string$adr (1) byte;
 12   1         declare index byte;

 13   1         emitchar: procedure(c);
 14   2             declare c byte;
 15   2             string(index := index + 1) = c;
 16   2             end emitchar;

 17   1         emitn: procedure(a);
```

```
18  2           declare a address;
19  2           declare c based a byte;
20  2           do while c <> '$';
21  3             string(index := index + 1) = c;
22  3             a = a + 1;
23  3           end;
24  2           end emitn;

25  1       emit$bcd: procedure(b);
26  2           declare b byte;
27  2           call emitchar('0'+b);
28  2           end emit$bcd;

29  1       emit$bcd$pair: procedure(b);
30  2           declare b byte;
31  2           call emit$bcd(shr(b,4));
32  2           call emit$bcd(b and 0fh);
33  2           end emit$bcd$pair;

34  1       emit$colon: procedure(b);
35  2           declare b byte;
36  2           call emit$bcd$pair(b);
37  2           call emitchar(':');
38  2           end emit$colon;

39  1       emit$bin$pair: procedure(b);
40  2           declare b byte;
41  2           call emit$bcd(b/10);    /* makes garbage if not < 10 */
42  2           call emit$bcd(b mod 10);
43  2           end emit$bin$pair;

44  1       emit$slant: procedure(b);
45  2           declare b byte;
46  2           call emit$bin$pair(b);
47  2           call emitchar('/');
48  2           end emit$slant;

49  1       declare
             base$year lit '73',   /* base year for computations */
             base$day  lit '0',    /* starting day for base$year 0..6 */
             month$days (*) address data
             /* jan feb mar apr may jun jul aug sep oct nov dec */
             (  000,031,059,090,120,151,181,212,243,273,304,334);

50  1       leap$days: procedure(y,m) byte;
51  2           declare (y,m) byte;
             /* compute days accumulated by leap years */
52  2           declare yp byte;
53  2           yp = shr(y,2); /* yp = y/4 */
54  2           if (y and 11b) = 0 and month$days(m) < 59 then
                 /* y not 00, y mod 4 = 0, before march, so not leap yr */
55  2               return yp - 1;
             /* otherwise, yp is the number of accumulated leap days */
56  2           return yp;
57  2           end leap$days;

58  1       declare word$value address;
```

```
59  1   get$next$digit: procedure byte;
            /* get next lsd from word$value */
60  2       declare lsd byte;
61  2       lsd = word$value mod 10;
62  2       word$value = word$value / 10;
63  2       return lsd;
64  2       end get$next$digit;

65  1   bcd:
          procedure (val) byte;
66  2       declare val byte;
67  2       return shl((val/10),4) + val mod 10;
68  2     end bcd;

69  1   declare (month, day, year, hrs, min, sec) byte;

70  1   bcd$pair: procedure(a,b) byte;
71  2       declare (a,b) byte;
72  2       return shl(a,4) or b;
73  2       end bcd$pair;


74  1   compute$year: procedure;
            /* compute year from number of days in word$value */
75  2       declare year$length address;
76  2       year = base$year;
77  2           do while true;
78  3           year$length = 365;
79  3           if (year and 11b) = 0 then /* leap year */
80  3               year$length = 366;
81  3           if word$value <= year$length then
82  3               return;
83  3           word$value = word$value - year$length;
84  3           year = year + 1;
85  3           end;
86  2       end compute$year;

87  1   declare
            week$day  byte, /* day of week 0 ... 6 */
            day$list (*) byte data
            ('Sun$Mon$Tue$Wed$Thu$Fri$Sat$'),
            leap$bias byte; /* bias for feb 29 */

88  1   compute$month: procedure;
89  2       month = 12;
90  2           do while month > 0;
91  3           if (month := month - 1) < 2 then /* jan or feb */
92  3               leapbias = 0;
93  3           if month$days(month) + leap$bias < word$value then return;
95  3           end;
96  2       end compute$month;

97  1   declare
            date$test byte,    /* true if testing date */
            test$value address;   /* sequential date value under test */
```

```
 98   1      get$date$time: procedure;
               /* get date and time */
 99   2          hrs = tod.hrs;
100   2          min = tod.min;
101   2          sec = tod.sec;
102   2          word$value = tod.date;
               /* word$value contains total number of days */
103   2          week$day = (word$value + base$day - 1) mod 7;
104   2          call compute$year;
               /* year has been set, word$value is remainder */
105   2          leap$bias = 0;
106   2          if (year and 11b) = 0 and word$value > 59 then
107   2              /* after feb 29 on leap year */ leap$bias = 1;
108   2          call compute$month;
109   2          day = word$value - (month$days(month) + leap$bias);
110   2          month = month + 1;
111   2          end get$date$time;

112   1      emit$date$time: procedure;
113   2          if tod.opcode = 0 then
114   2            do;
115   3            call emitn(.day$list(shl(week$day,2)));
116   3            call emitchar(' ');
117   3            end;
118   2          call emit$slant(month);
119   2          call emit$slant(day);
120   2          call emit$bin$pair(year);
121   2          call emitchar(' ');
122   2          call emit$colon(hrs);
123   2          call emit$colon(min);
124   2          if tod.opcode = 0 then
125   2            call emit$bcd$pair(sec);
126   2          end emit$date$time;

127   1      tod$ASCII:
             procedure (parameter);
128   2          declare parameter address;
129   2          declare ret address;

130   2          ret = 0;
131   2          tod$adr = parameter;
132   2          string$adr = .tod.ASCII;
133   2          if  (tod.opcode = 0) or (tod.opcode = 3) then
134   2          do;
135   3            call get$date$time;
136   3            index = -1;
137   3            call emit$date$time;
138   3          end;
               else
139   2            call terminate;              /* error */
140   2      end tod$ASCII;

141   1        declare lcltod structure (
              opcode byte,
              date address,
              hrs byte,
              min byte,
```

```
                sec byte,
                ASCII (21) byte );

142   1     display$time$stamp: procedure (tsadr) public;
143   2         dcl tsadr address,
                  i byte;

144   2          lcltod.opcode = 3;     /* display time and date stamp, no seconds */
145   2          call move (4,tsadr,.lcltod.date);  /* don't copy seconds */

146   2          call tod$ASCII (.lcltod);
147   2          do i = 0 to 13;
148   3            call printchar (lcltod.ASCII(i));
149   3          end;
150   2     end display$time$stamp;

151   1     dcl last$data$byte byte initial(0);

152   1     end timestamp;
```

CROSS-REFERENCE LISTING

| DEFN | ADDR | SIZE | NAME, ATTRIBUTES, AND REFERENCES | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 17 | 0004H | 2 | A. . . . . | WORD PARAMETER AUTOMATIC | 18 | 19 | 20 | 21 | 22 | | |
| 70 | 0006H | 1 | A. . . . . | BYTE PARAMETER AUTOMATIC | 71 | 72 | | | | | |
| 141 | 0006H | 21 | ASCII. . . . . | BYTE ARRAY(21) MEMBER(LCLTOD) | 148 | | | | | | |
| 9 | 0006H | 21 | ASCII. . . . . | BYTE ARRAY(21) MEMBER(TOD) | 132 | | | | | | |
| 34 | 0004H | 1 | B. . . . . | BYTE PARAMETER AUTOMATIC | 35 | 36 | | | | | |
| 70 | 0004H | 1 | B. . . . . | BYTE PARAMETER AUTOMATIC | 71 | 72 | | | | | |
| 29 | 0004H | 1 | B. . . . . | BYTE PARAMETER AUTOMATIC | 30 | 31 | 32 | | | | |
| 44 | 0004H | 1 | B. . . . . | BYTE PARAMETER AUTOMATIC | 45 | 46 | | | | | |
| 39 | 0004H | 1 | B. . . . . | BYTE PARAMETER AUTOMATIC | 40 | 41 | 42 | | | | |
| 25 | 0004H | 1 | B. . . . . | BYTE PARAMETER AUTOMATIC | 26 | 27 | | | | | |
| 49 | | | BASEDAY. . . . . | LITERALLY 103 | | | | | | | |
| 49 | | | BASEYEAR . . . . . | LITERALLY 76 | | | | | | | |
| 65 | 0108H | 39 | BCD. . . . . | PROCEDURE BYTE STACK=0006H | | | | | | | |
| 70 | 012FH | 17 | BCDPAIR. . . . . | PROCEDURE BYTE STACK=0006H | | | | | | | |
| 2 | | | BOOLEAN. . . . . | LITERALLY | | | | | | | |
| 13 | 0004H | 1 | C. . . . . | BYTE PARAMETER AUTOMATIC | 14 | 15 | | | | | |
| 19 | 0000H | 1 | C. . . . . | BYTE BASED(A) 20 21 | | | | | | | |
| 3 | 0000H | 1 | CHAR . . . . . | BYTE PARAMETER 4 | | | | | | | |
| 88 | 0175H | 59 | COMPUTEMONTH . . . | PROCEDURE STACK=0002H | 108 | | | | | | |
| 74 | 0140H | 53 | COMPUTEYEAR. . . . | PROCEDURE STACK=0002H | 104 | | | | | | |
| 2 | | | CR . . . . . | LITERALLY | | | | | | | |
| 2 | | | CTRLC. . . . . | LITERALLY | | | | | | | |
| 141 | 0001H | 2 | DATE . . . . . | WORD MEMBER(LCLTOD) 145 | | | | | | | |
| 9 | 0001H | 2 | DATE . . . . . | WORD MEMBER(TOD) 102 | | | | | | | |
| 97 | 0017H | 1 | DATETEST . . . . . | BYTE | | | | | | | |
| 69 | 0010H | 1 | DAY. . . . . | BYTE 109 119 | | | | | | | |
| 87 | 0018H | 28 | DAYLIST. . . . . | BYTE ARRAY(28) DATA 115 | | | | | | | |
| 2 | | | DCL. . . . . | LITERALLY | | | | | | | |
| 142 | 02AFH | 64 | DISPLAYTIMESTAMP . | PROCEDURE PUBLIC STACK=0026H | | | | | | | |
| 25 | 0044H | 16 | EMITBCD. . . . . | PROCEDURE STACK=000AH | 31 | 32 | 41 | 42 | | | |
| 29 | 0054H | 27 | EMITBCDPAIR. . . . | PROCEDURE STACK=0010H | 36 | 125 | | | | | |
| 39 | 0082H | 39 | EMITBINPAIR. . . . | PROCEDURE STACK=0010H | 46 | 120 | | | | | |
| 13 | 0000H | 28 | EMITCHAR . . . . . | PROCEDURE STACK=0004H | 27 | 37 | 47 | 116 | 121 | | |
| 34 | 006FH | 19 | EMITCOLON. . . . . | PROCEDURE STACK=0016H | 122 | 123 | | | | | |
| 112 | 021AH | 95 | EMITDATETIME . . . | PROCEDURE STACK=001AH | 137 | | | | | | |
| 17 | 001CH | 40 | EMITN. . . . . | PROCEDURE STACK=0004H | 115 | | | | | | |
| 44 | 00A9H | 19 | EMITSLANT. . . . . | PROCEDURE STACK=0016H | 118 | 119 | | | | | |
| 2 | | | FALSE. . . . . | LITERALLY | | | | | | | |
| 2 | | | FF . . . . . | LITERALLY | | | | | | | |
| 2 | | | FOREVER. . . . . | LITERALLY | | | | | | | |
| 98 | 0180H | 106 | GETDATETIME. . . . | PROCEDURE STACK=0006H | 135 | | | | | | |
| 59 | 00E8H | 32 | GETNEXTDIGIT . . . | PROCEDURE BYTE STACK=0002H | | | | | | | |
| 141 | 0003H | 1 | HRS. . . . . | BYTE MEMBER(LCLTOD) | | | | | | | |
| 69 | 0012H | 1 | HRS. . . . . | BYTE 99 122 | | | | | | | |
| 9 | 0003H | 1 | HRS. . . . . | BYTE MEMBER(TOD) 99 | | | | | | | |
| 143 | 0033H | 1 | I. . . . . | BYTE 147 148 | | | | | | | |
| 12 | 000CH | 1 | INDEX. . . . . | BYTE 15 21 136 | | | | | | | |
| 151 | 0034H | 1 | LASTDATABYTE . . . | BYTE INITIAL | | | | | | | |
| 141 | 0018H | 27 | LCLTOD . . . . . | STRUCTURE 144 145 146 148 | | | | | | | |

```
  87  0016H     1  LEAPBIAS . . . . .  BYTE          92   95  105  107  109
  50  008CH    44  LEAPDAYS . . . . .  PROCEDURE BYTE STACK=0006H
   2              LF . . . . . . . . .  LITERALLY
   2              LIT. . . . . . . . .  LITERALLY           2   49
  60  000EH     1  LSD. . . . . . . . .  BYTE          61   63
  50  0004H     1  M. . . . . . . . . .  BYTE PARAMETER AUTOMATIC          51   54
  69  0013H     1  MIN. . . . . . . . .  BYTE         100  125
   9  0004H     1  MIN. . . . . . . . .  BYTE MEMBER(TOD)            100
 141  0004H     1  MIN. . . . . . . . .  BYTE MEMBER(LCLTOD)
  69  000FH     1  MONTH. . . . . . . .  BYTE          89   90   91   93  109  110  118
  49  0000H    24  MONTHDAYS. . . . . .  WORD ARRAY(12) DATA          54   93  109
                   MOVE . . . . . . . .  BUILTIN           145
   2              NOPAGEMODEOFFSET . .  LITERALLY
   9  0000H     1  OPCODE . . . . . . .  BYTE MEMBER(TOD)            113  124  133
 141  0000H     1  OPCODE . . . . . . .  BYTE MEMBER(LCLTOD)         144
   2              PAGELENOFFSET. . . .  LITERALLY
 127  0004H     2  PARAMETER. . . . . .  WORD PARAMETER AUTOMATIC         128  131
   3  0000H        PRINTCHAR. . . . . .  PROCEDURE EXTERNAL(0) STACK=0000H          148
 129  000AH     2  RET. . . . . . . . .  WORD         130
  69  0014H     1  SEC. . . . . . . . .  BYTE         101  125
   9  0005H     1  SEC. . . . . . . . .  BYTE MEMBER(TOD)            101
 141  0005H     1  SEC. . . . . . . . .  BYTE MEMBER(LCLTOD)
   2              SECTORLEN. . . . . .  LITERALLY
                  SHL. . . . . . . . .  BUILTIN           67   72  115
                  SHR. . . . . . . . .  BUILTIN           31   53
  11  0000H     1  STRING . . . . . . .  BYTE BASED(STRINGADR) ARRAY(1)          15   21
  10  0002H     2  STRINGADR. . . . . .  WORD          11   15   21  132
   2              TAB. . . . . . . . .  LITERALLY
   6  0000H        TERMINATE. . . . . .  PROCEDURE EXTERNAL(1) STACK=0000H          139
  97  0008H     2  TESTVALUE. . . . . .  WORD
   1  0000H        TIMESTAMP. . . . . .  PROCEDURE STACK=0000H
   9  0000H    27  TOD. . . . . . . . .  STRUCTURE BASED(TODADR)          99  100  101  102  113  124  132  133
   8  0000H     2  TODADR . . . . . . .  WORD           9   99  100  101  102  113  124  131  132  133
 127  0279H    54  TODASCII . . . . . .  PROCEDURE STACK=0020H            146
   2              TRUE . . . . . . . .  LITERALLY          77
 142  0004H     2  TSADR. . . . . . . .  WORD PARAMETER AUTOMATIC          143  145
  65  0004H     1  VAL. . . . . . . . .  BYTE PARAMETER AUTOMATIC          66   67
  87  0015H     1  WEEKDAY. . . . . . .  BYTE         103  115
  58  0004H     2  WORDVALUE. . . . . .  WORD          61   62   81   83   93  102  103  106  109
  50  0006H     1  Y. . . . . . . . . .  BYTE PARAMETER AUTOMATIC          51   53   54
  69  0011H     1  YEAR . . . . . . . .  BYTE          76   79   84  106  120
  75  0006H     2  YEARLENGTH . . . . .  WORD          78   80   81   83
  52  0000H     1  YP . . . . . . . . .  BYTE          53   55   56
```



MODULE INFORMATION:

```
  CODE AREA SIZE     = 02EFH    751D
  CONSTANT AREA SIZE = 0034H     52D
  VARIABLE AREA SIZE = 0035H     53D
  MAXIMUM STACK SIZE = 0026H     38D
  241 LINES READ
  0 PROGRAM ERROR(S)
```

END OF PL/M-86 COMPILATION

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

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

BASE   OFFSET TYPE SYMBOL                BASE   OFFSET TYPE SYMBOL

0000H  28F0H  PUB  DISPLAYTIMESTAMP      0000H  2628H  PUB  BASEOPB
0000H  2605H  PUB  DPBWORD               0000H  25F0H  PUB  DP8BYTE
0000H  2595H  PUB  P38YTE                0000H  253FH  PUB  PDECIMAL
0000H  24EAH  PUB  PRINTFN               0000H  2499H  PUB  CRLF
0000H  24CEH  PUB  PRINTB                0000H  248EH  PUB  PRINT
0000H  24ABH  PUB  PRINTCHAR             0000H  248DH  PUB  SHR3BYTE
0000H  246FH  PUB  ADD38YTE3             0000H  2452H  PUB  ADD3BYTE
0000H  2297H  PUB  DISPLAYFILES          0000H  1C20H  PUB  SORT
0000H  1A44H  PUB  MULT23                0000H  1A26H  PUB  SEARCHINIT
0000H  193CH  PUB  GETFILES              0000H  1580H  PUB  MATCH
0000H  1530H  PUB  SEARCHNEXT           0000H  1520H  PUB  SEARCHFIRST
0000H  14D6H  PUB  SCANINIT              0000H  1413H  PUB  SCAN
0000H  0388H  PUB  SETVEC                0000H  02F1H  PUB  TERMINATE
0000H  0110H  PUB  PLMSTART              0000H  0050H  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  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  RD
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
1000H  0272H  PUB  KPERBLOCK             1000H  0258H  PUB  FILEDISPLAYED
1000H  0206H  PUB  FINDICESBASE          1000H  0248H  PUB  SORTED
1000H  01F0H  PUB  XIADR                 1000H  01ECH  PUB  LASTFIADR
1000H  01EAH  PUB  FIADR                 1000H  01E8H  PUB  FIRSTFIADR
1000H  01E6H  PUB  BUFFCBADR             1000H  01F8H  PUB  DIRLABEL
1000H  01E4H  PUB  FILESFOUND            1000H  01E2H  PUB  USEDDE
1000H  01BAH  PUB  CURDRV                1000H  01B9H  PUB  CURUSR
1000H  01B7H  PUB  DISPLAYATTRIBUTE      1000H  01B6H  PUB  DATEOPT
                   -S
1000H  01B5H  PUB  FORMFEEDS             1000H  01B4H  PUB  MESSAGE
1000H  0116H  PUB  PAGELEN              1000H  01B3H  PUB  FORMAT
1000H  0112H  PUB  ACTIVEUSRVECTOR      1000H  0110H  PUB  USRVECTOR
1000H  01B2H  PUB  FIRSTPASS             1000H  01B1H  PUB  GETALLDIRENTRIES
1000H  012FH  PUB  SEARCH                1000H  012EH  PUB  NOPAGEMODE
1000H  012DH  PUB  NUMSEARCHFILES        1000H  0103H  PUB  FIND
1000H  012CH  PUB  BDOS                  1000H  0128H  PUB  OS
1000H  012AH  PUB  DEBUG

SDIR:  SYMBOLS AND LINES
1000H  07A0H  SYM  MEMORY                1000H  012AH  SYM  DEBUG
0000H  0110H  SYM  PLMSTART              1000H  0128H  SYM  OS
1000H  012CH  SYM  BDOS                  1000H  0103H  SYM  FIND
1000H  012DH  SYM  NUMSEARCHFILES        1000H  012EH  SYM  NOPAGEMODE
1000H  012FH  SYM  SEARCH                1000H  01B1H  SYM  GETALLDIRENTRIES
1000H  01B2H  SYM  FIRSTPASS             1000H  0110H  SYM  USRVECTOR
```

```
1000H  0112H  SYM  ACTIVEUSRVECTOR      1000H  0114H  SYM  DRVVECTOR
1000H  0183H  SYM  FORMAT              1000H  0116H  SYM  PASSLEN
1000H  0184H  SYM  MESSAGE             1000H  0135H  SYM  FORWARDS
1000H  0186H  SYM  DATEOPT             1000H  0187H  SYM  DISPLAYATTRIBUTE
                                                         -S
1000H  01B8H  SYM  SORTOP              1000H  0189H  SYM  CURUSR
1000H  01BAH  SYM  CURDRV              0000H  029DH  SYM  GETVERSION
0000H  02ACH  SYM  SELECTDRIVE         STACK  0004H  SYM  D
0000H  02BFH  SYM  GETCURDRV           0000H  02CEH  SYM  GETLOGIN
0000H  02D0H  SYM  GETUSR              0000H  02ECH  SYM  SETCONSOLEMODE
0000H  02F1H  SYM  TERMINATE           0000H  02FFH  SYM  NUMBER
STACK  0004H  SYM  CHAR                0000H  031CH  SYM  MAKENUMERIC
STACK  0008H  SYM  CHARADR             STACK  0006H  SYM  LEN
STACK  0004H  SYM  VALADR              1000H  0118H  SYM  PLACE
STACK  0008H  BAS  CHARS               STACK  0004H  BAS  VALUE
1000H  01BBH  SYM  I                   0000H  0388H  SYM  SETVEC
STACK  0006H  SYM  VADR                STACK  0004H  SYM  NUM
STACK  0006H  BAS  VECTOR              0000H  03ABH  SYM  BITLOC
STACK  0004H  SYM  VECTOR              1000H  018CH  SYM  I
0000H  03E3H  SYM  GETNXT              STACK  0004H  SYM  VECTORADR
1000H  018DH  SYM  I                   1000H  011AH  SYM  MASK
STACK  0004H  BAS  VECTOR              1000H  011CH  SYM  PCB
1000H  010CH  BAS  TOKEN               1000H  01BEH  SYM  GOTOPTIONS
0000H  041DH  SYM  GETOPTIONS          1000H  01BFH  SYM  TEMP
0000H  0772H  SYM  OPERR               0000H  077EH  SYM  GETFILESPEC
1000H  01C0H  SYM  I                   0000H  0830H  SYM  SETDEFAULTS
1000H  0126H  SYM  SAVEUVEC            1000H  0128H  SYM  TEMP
1000H  01C1H  SYM  I                   1000H  01C2H  SYM  LASTDSEGBYTE
0000H  029DH  LIN     63               0000H  02A0H  LIN     64
0000H  02ACH  LIN     65               0000H  02ACH  LIN     66
0000H  02AFH  LIN     68               0000H  02B8H  LIN     69
0000H  02BFH  LIN     75               0000H  02C2H  LIN     76
0000H  02CEH  LIN     77               0000H  02CEH  LIN     78
0000H  02D1H  LIN     79               0000H  02D0H  LIN     80
0000H  02D0H  LIN     81               0000H  02E0H  LIN     82
0000H  02ECH  LIN     83               0000H  02ECH  LIN     84
0000H  02EFH  LIN     86               0000H  02F1H  LIN     87
0000H  02F4H  LIN     88               0000H  02FDH  LIN     89
0000H  02FFH  LIN     90               0000H  0302H  LIN     92
0000H  031CH  LIN     93               0000H  031CH  LIN     94
0000H  031FH  LIN     96               0000H  0326H  LIN     97
0000H  032CH  LIN     98               0000H  0339H  LIN     99
0000H  0350H  LIN    100               0000H  0354H  LIN    101
0000H  0371H  LIN    102               0000H  037CH  LIN    103
0000H  0382H  LIN    104               0000H  0388H  LIN    105
0000H  0388H  LIN    106               0000H  038BH  LIN    108
0000H  0391H  LIN    109               0000H  039AH  LIN    110
0000H  03A7H  LIN    111               0000H  03ABH  LIN    112
0000H  03AEH  LIN    114               0000H  03B3H  LIN    115
0000H  03D3H  LIN    116               0000H  03D6H  LIN    117
0000H  03DAH  LIN    118               0000H  03DCH  LIN    119
0000H  03E3H  LIN    120               0000H  03E3H  LIN    121
0000H  03E6H  LIN    123               0000H  03F5H  LIN    124
0000H  03F9H  LIN    125               0000H  03FFH  LIN    126
0000H  0406H  LIN    127               0000H  040EH  LIN    128
0000H  0416H  LIN    129               0000H  041DH  LIN    130
0000H  041DH  LIN    136               0000H  0420H  LIN    138
0000H  0443H  LIN    139               0000H  044DH  LIN    141
0000H  0457H  LIN    142               0000H  045EH  LIN    143
0000H  046EH  LIN    144               0000H  0475H  LIN    145
```

```
0000H  0485H  LIN    147
0000H  048FH  LIN    149
0000H  0498H  LIN    151
0000H  04ACH  LIN    153
0000H  04B9H  LIN    155
0000H  04C8H  LIN    157
0000H  04D7H  LIN    160
0000H  04E5H  LIN    162
0000H  0509H  LIN    164
0000H  0516H  LIN    167
0000H  0527H  LIN    169
0000H  0537H  LIN    171
0000H  0548H  LIN    173
0000H  0559H  LIN    175
0000H  0560H  LIN    177
0000H  0577H  LIN    179
0000H  058EH  LIN    181
0000H  059EH  LIN    183
0000H  05AFH  LIN    185
0000H  05C0H  LIN    187
0000H  05C7H  LIN    189
0000H  05D9H  LIN    192
0000H  05DBH  LIN    195
0000H  05ECH  LIN    198
0000H  060AH  LIN    200
0000H  0618H  LIN    205
0000H  0628H  LIN    208
0000H  0640H  LIN    211
0000H  066EH  LIN    213
0000H  0687H  LIN    216
0000H  0699H  LIN    218
0000H  06B0H  LIN    220
0000H  06C8H  LIN    224
0000H  0602H  LIN    226
0000H  06F8H  LIN    229
0000H  0709H  LIN    232
0000H  073AH  LIN    235
0000H  0747H  LIN    237
0000H  0758H  LIN    239
0000H  0762H  LIN    241
0000H  076BH  LIN    246
0000H  0770H  LIN    248
0000H  0779H  LIN    250
0000H  077EH  LIN    252
0000H  0786H  LIN    256
0000H  07C4H  LIN    258
0000H  07D9H  LIN    260
0000H  07F2H  LIN    262
0000H  080DH  LIN    264
0000H  0816H  LIN    267
0000H  0827H  LIN    270
0000H  0830H  LIN    272
0000H  0840H  LIN    274
0000H  0857H  LIN    276
0000H  086CH  LIN    279
0000H  0878H  LIN    281
0000H  0884H  LIN    283
0000H  0899H  LIN    286
0000H  08A3H  LIN    289
0000H  08BCH  LIN    291
0000H  048AH  LIN    148
0000H  0491H  LIN    150
0000H  04A2H  LIN    152
0000H  04B2H  LIN    154
0000H  04C6H  LIN    156
0000H  04CDH  LIN    158
0000H  04DCH  LIN    161
0000H  04FFH  LIN    163
0000H  0514H  LIN    166
0000H  0520H  LIN    168
0000H  0531H  LIN    170
0000H  053EH  LIN    172
0000H  054FH  LIN    174
0000H  055EH  LIN    176
0000H  0570H  LIN    178
0000H  0587H  LIN    180
0000H  0598H  LIN    182
0000H  05A5H  LIN    184
0000H  05B6H  LIN    186
0000H  05C5H  LIN    188
0000H  05D4H  LIN    190
0000H  05DBH  LIN    193
0000H  05E5H  LIN    197
0000H  05F3H  LIN    199
0000H  0611H  LIN    202
0000H  061BH  LIN    206
0000H  062FH  LIN    209
0000H  065CH  LIN    212
0000H  0676H  LIN    214
0000H  068EH  LIN    217
0000H  06B3H  LIN    219
0000H  06C8H  LIN    223
0000H  06CFH  LIN    225
0000H  0602H  LIN    227
0000H  0702H  LIN    230
0000H  072AH  LIN    233
0000H  0740H  LIN    236
0000H  0749H  LIN    238
0000H  0762H  LIN    240
0000H  0769H  LIN    242
0000H  076BH  LIN    247
0000H  0772H  LIN    249
0000H  077CH  LIN    251
0000H  0781H  LIN    254
0000H  07A7H  LIN    257
0000H  07CBH  LIN    259
0000H  07E2H  LIN    261
0000H  0809H  LIN    263
0000H  080FH  LIN    266
0000H  0824H  LIN    269
0000H  082EH  LIN    271
0000H  0833H  LIN    273
0000H  084AH  LIN    275
0000H  0861H  LIN    277
0000H  0873H  LIN    280
0000H  087AH  LIN    282
0000H  088BH  LIN    285
0000H  089EH  LIN    287
0000H  08AAH  LIN    290
0000H  08CBH  LIN    292
```



SER. #_____

```
0000H  08D2H  LIN    293
0000H  08EEH  LIN    295
0000H  0919H  LIN    298
0000H  0923H  LIN    301
0000H  0930H  LIN    303
0000H  0102H  LIN    308
0000H  011DH  LIN    310
0000H  012AH  LIN    313
0000H  0134H  LIN    315
0000H  0139H  LIN    317
0000H  0146H  LIN    319
0000H  0152H  LIN    321
0000H  015EH  LIN    323
0000H  016CH  LIN    325
0000H  0178H  LIN    328
0000H  017DH  LIN    330
0000H  0189H  LIN    333
0000H  018EH  LIN    337
0000H  0194H  LIN    339
0000H  01ACH  LIN    341
0000H  01B8H  LIN    343
0000H  01C7H  LIN    345
0000H  01DAH  LIN    348
0000H  01EAH  LIN    350
0000H  01F5H  LIN    352
0000H  01FDH  LIN    354
0000H  0217H  LIN    357
0000H  0222H  LIN    359
0000H  0234H  LIN    361
0000H  0251H  LIN    365
0000H  0259H  LIN    367
0000H  0265H  LIN    369
0000H  0268H  LIN    373
0000H  0277H  LIN    375
0000H  0280H  LIN    377
0000H  0298H  LIN    379

0000H  08E6H  LIN    294
0000H  0900H  LIN    296
0000H  0920H  LIN    299
0000H  0929H  LIN    302
0000H  0932H  LIN    304
0000H  0115H  LIN    309
0000H  0123H  LIN    311
0000H  0131H  LIN    314
0000H  0136H  LIN    316
0000H  013FH  LIN    318
0000H  0140H  LIN    320
0000H  0157H  LIN    322
0000H  0165H  LIN    324
0000H  0171H  LIN    327
0000H  017EH  LIN    329
0000H  0184H  LIN    331
0000H  018EH  LIN    336
0000H  0191H  LIN    338
0000H  01A5H  LIN    340
0000H  01B2H  LIN    342
0000H  01C2H  LIN    344
0000H  01D5H  LIN    347
0000H  01E4H  LIN    349
0000H  01F1H  LIN    351
0000H  01FBH  LIN    353
0000H  0212H  LIN    355
0000H  021CH  LIN    358
0000H  0229H  LIN    360
0000H  023DH  LIN    363
0000H  0254H  LIN    366
0000H  025EH  LIN    368
0000H  0268H  LIN    371
0000H  0275H  LIN    374
0000H  0279H  LIN    376
0000H  0291H  LIN    378
0000H  029BH  LIN    381
```



SCANNER: SYMBOLS AND LINES

```
1000H  07A0H  SYM  MEMORY
0000H  093EH  SYM  WHITESPACE
STACK  0004H  BAS  STR
0000H  0978H  SYM  DELIMITER
0000H  09A4H  SYM  DEBLANK
1000H  01C4H  SYM  DEST
1000H  01D1H  SYM  I
1000H  01D3H  SYM  STRING
STACK  0004H  SYM  BUFADR
1000H  01D4H  SYM  I
1000H  01C6H  BAS  PCB
1000H  01C8H  BAS  INBUF
1000H  01CAH  SYM  TOKENADR
1000H  01D6H  SYM  TPTR
1000H  01D8H  SYM  NXTCHAR
0000H  0B7EH  SYM  DIGIT
0000H  0B9BH  SYM  LETTER
0000H  0B88H  SYM  EATCHAR
STACK  0004H  SYM  CHARX
STACK  0004H  SYM  MAX
STACK  0004H  SYM  X
STACK  0004H  SYM  FIELDSIZE

1000H  01CEH  SYM  DEBUG
STACK  0004H  SYM  STRADR
1000H  01D0H  SYM  I
STACK  0004H  SYM  CHAR
STACK  0004H  SYM  BUFADR
STACK  0004H  BAS  BUF
1000H  01D2H  SYM  NUMSPACES
0000H  084FH  SYM  UPPERCASE
STACK  0004H  BAS  BUF
1000H  01C6H  SYM  PCBBASE
1000H  01C8H  SYM  SCANADR
1000H  01D5H  SYM  INPTR
1000H  01CAH  BAS  TOKEN
1000H  01D7H  SYM  CHAR
1000H  01D9H  SYM  TCOUNT
STACK  0004H  SYM  CHAR
STACK  0004H  SYM  CHAR
0000H  0B09H  SYM  PUTCHAR
0000H  0BFFH  SYM  GETIDENTIFIER
0000H  0CC7H  SYM  FILECHAR
0000H  0D13H  SYM  EXPANDWILDCARDS
1000H  01DAH  SYM  I
```

```
1000H  01DBH  SYM  LEFTOVER
0000H  00ABH  SYM  GETFILESPEC
0000H  0E89H  SYM  GETNUMERIC
0000H  0F18H  SYM  GETSTRING
0000H  0FA2H  SYM  GETTOKENALL
0000H  0FF2H  SYM  GETMODIFIER
0000H  1052H  SYM  GETPARAM
1000H  01DFH  SYM  PARENS
0000H  10AAH  SYM  STATE8
0000H  1167H  SYM  STATE6
0000H  1200H  SYM  STATE4
0000H  127BH  SYM  STATE3
0000H  1345H  SYM  STATE1
0000H  1413H  SYM  SCAN
1000H  01E0H  SYM  STATUS
STACK  0004H  SYM  PCBADR
0000H  0941H  LIN     28
0000H  0968H  LIN     30
0000H  0971H  LIN     32
0000H  0978H  LIN     34
0000H  099AH  LIN     37
0000H  09A4H  LIN     39
0000H  09A7H  LIN     43
0000H  09BDH  LIN     45
0000H  09E1H  LIN     47
0000H  0A30H  LIN     49
0000H  0A44H  LIN     52
0000H  0A4BH  LIN     55
0000H  0A4DH  LIN     61
0000H  0A5FH  LIN     64
0000H  0AA0H  LIN     66
0000H  0AC4H  LIN     69
0000H  0ACFH  LIN     71
0000H  0AF9H  LIN     73
0000H  0B28H  LIN     75
0000H  0B34H  LIN     78
0000H  0B48H  LIN     80
0000H  0B4FH  LIN     82
0000H  0B57H  LIN     85
0000H  0B71H  LIN     87
0000H  0B78H  LIN     89
0000H  0B7EH  LIN     99
0000H  0B9BH  LIN    102
0000H  0B9EH  LIN    105
0000H  0BB8H  LIN    107
0000H  0BD1H  LIN    109
0000H  0BD9H  LIN    111
0000H  0BE6H  LIN    114
0000H  0BFFH  LIN    116
0000H  0C07H  LIN    119
0000H  0C28H  LIN    121
0000H  0C71H  LIN    123
0000H  0C78H  LIN    125
0000H  0CACH  LIN    127
0000H  0CB3H  LIN    129
0000H  0CBDH  LIN    131
0000H  0CC7H  LIN    133
0000H  0CCAH  LIN    136
0000H  0D13H  LIN    138
0000H  0D1CH  LIN    141

1000H  01CCH  SYM  SAVETAPTR
1000H  01DCH  SYM  T
STACK  0004H  SYM  MAX
STACK  0004H  SYM  MAX
1000H  01DDH  SYM  SAVEINPTR
0000H  1025H  SYM  GETOPTION
1000H  01DEH  SYM  GOTATOKEN
0000H  1085H  SYM  ENDSTATE
0000H  1117H  SYM  STATE7
0000H  11ACH  SYM  STATE5
STACK  0002H  SYM  TEMP
0000H  12CBH  SYM  STATE2
0000H  13A8H  SYM  STARTSTATE
STACK  0004H  SYM  PCBADR
0000H  14D6H  SYM  SCANINIT
0000H  093EH  LIN     26
0000H  0946H  LIN     29
0000H  096FH  LIN     31
0000H  0978H  LIN     33
0000H  097BH  LIN     36
0000H  099EH  LIN     38
0000H  09A4H  LIN     41
0000H  09ACH  LIN     44
0000H  09DCH  LIN     46
0000H  09F3H  LIN     48
0000H  0A3FH  LIN     51
0000H  0A47H  LIN     54
0000H  0A4DH  LIN     59
0000H  0A5AH  LIN     62
0000H  0A63H  LIN     65
0000H  0A87H  LIN     68
0000H  0ACBH  LIN     70
0000H  0AF5H  LIN     72
0000H  0B1CH  LIN     74
0000H  0B30H  LIN     76
0000H  0B37H  LIN     79
0000H  0B48H  LIN     81
0000H  0B52H  LIN     84
0000H  0B67H  LIN     86
0000H  0B74H  LIN     88
0000H  0B7AH  LIN     90
0000H  0B81H  LIN    101
0000H  0B9BH  LIN    103
0000H  0BB8H  LIN    106
0000H  0BBBH  LIN    108
0000H  0BD7H  LIN    110
0000H  0BDCH  LIN    113
0000H  0BFBH  LIN    115
0000H  0C02H  LIN    118
0000H  0C22H  LIN    120
0000H  0C6AH  LIN    122
0000H  0C74H  LIN    124
0000H  0C7AH  LIN    126
0000H  0CAFH  LIN    128
0000H  0CB5H  LIN    130
0000H  0CC3H  LIN    132
0000H  0CC7H  LIN    134
0000H  0D13H  LIN    137
0000H  0D16H  LIN    140
0000H  0D37H  LIN    142
```

```
0000H  0D3EH  LIN    144     0000H  0D44H  LIN    145
0000H  0D4CH  LIN    146     0000H  0D4FH  LIN    147
0000H  0D5AH  LIN    148     0000H  0D5EH  LIN    149
0000H  0D60H  LIN    150     0000H  0D6CH  LIN    151
0000H  0D69H  LIN    153     0000H  0D6EH  LIN    154
0000H  0D72H  LIN    156     0000H  0D84H  LIN    157
0000H  0D8AH  LIN    158     0000H  0D90H  LIN    159
0000H  0D96H  LIN    160     0000H  0D98H  LIN    161
0000H  0D9FH  LIN    162     0000H  0DA2H  LIN    163
0000H  0DA5H  LIN    164     0000H  0DA6H  LIN    165
0000H  0DABH  LIN    166     0000H  0DAEH  LIN    168
0000H  0DBAH  LIN    169     0000H  0DC8H  LIN    170
0000H  0DCEH  LIN    171     0000H  0DD5H  LIN    172
0000H  0DE3H  LIN    174     0000H  0DE6H  LIN    175
0000H  0DEEH  LIN    176     0000H  0DF1H  LIN    177
0000H  0DF3H  LIN    178     0000H  0DF3H  LIN    179
0000H  0DF9H  LIN    180     0000H  0E3AH  LIN    181
0000H  0E43H  LIN    182     0000H  0E43H  LIN    183
0000H  0E4FH  LIN    185     0000H  0E56H  LIN    187
0000H  0E59H  LIN    188     0000H  0E64H  LIN    190
0000H  0E69H  LIN    191     0000H  0E75H  LIN    192
0000H  0E79H  LIN    195     0000H  0E81H  LIN    196
0000H  0E85H  LIN    197     0000H  0E89H  LIN    198
0000H  0E89H  LIN    199     0000H  0E8CH  LIN    201
0000H  0E99H  LIN    203     0000H  0EC5H  LIN    204
0000H  0ECCH  LIN    205     0000H  0ECFH  LIN    206
0000H  0ED6H  LIN    207     0000H  0ED8H  LIN    208
0000H  0EE7H  LIN    209     0000H  0EF3H  LIN    211
0000H  0EFAH  LIN    212     0000H  0EFDH  LIN    213
0000H  0F04H  LIN    214     0000H  0F06H  LIN    215
0000H  0F0AH  LIN    216     0000H  0F12H  LIN    217
0000H  0F18H  LIN    218     0000H  0F18H  LIN    219
0000H  0F1BH  LIN    221     0000H  0F22H  LIN    223
0000H  0F25H  LIN    224     0000H  0F52H  LIN    225
0000H  0F59H  LIN    226     0000H  0F5CH  LIN    227
0000H  0F63H  LIN    228     0000H  0F65H  LIN    229
0000H  0F81H  LIN    230     0000H  0F84H  LIN    231
0000H  0F86H  LIN    232     0000H  0F8DH  LIN    233
0000H  0F91H  LIN    234     0000H  0F99H  LIN    235
0000H  0F9CH  LIN    236     0000H  0FA2H  LIN    237
0000H  0FA2H  LIN    238     0000H  0FA5H  LIN    240
0000H  0FABH  LIN    241     0000H  0FB2H  LIN    243
0000H  0FBAH  LIN    244     0000H  0FBDH  LIN    245
0000H  0FC2H  LIN    246     0000H  0FC8H  LIN    247
0000H  0FD2H  LIN    248     0000H  0FDEH  LIN    249
0000H  0FEAH  LIN    250     0000H  0FEEH  LIN    251
0000H  0FF2H  LIN    252     0000H  0FF2H  LIN    253
0000H  0FF5H  LIN    254     0000H  1004H  LIN    256
0000H  100CH  LIN    257     0000H  100EH  LIN    259
0000H  1015H  LIN    261     0000H  101DH  LIN    262
0000H  1021H  LIN    264     0000H  1025H  LIN    265
0000H  1025H  LIN    266     0000H  1028H  LIN    267
0000H  102EH  LIN    268     0000H  1038H  LIN    270
0000H  1040H  LIN    271     0000H  1046H  LIN    272
0000H  104AH  LIN    273     0000H  104EH  LIN    275
0000H  1052H  LIN    276     0000H  1052H  LIN    277
0000H  1055H  LIN    278     0000H  1064H  LIN    280
0000H  106CH  LIN    281     0000H  106EH  LIN    283
0000H  1075H  LIN    285     0000H  107DH  LIN    286
0000H  1081H  LIN    288     0000H  1085H  LIN    289
```

```
0000H  1085H  LIN   292
0000H  108FH  LIN   295
0000H  1099H  LIN   298
0000H  10A6H  LIN   300
0000H  10AAH  LIN   302
0000H  1084H  LIN   305
0000H  10C5H  LIN   308
0000H  10D3H  LIN   312
0000H  10E3H  LIN   314
0000H  10F7H  LIN   316
0000H  1101H  LIN   318
0000H  110FH  LIN   321
0000H  1117H  LIN   325
0000H  111AH  LIN   327
0000H  112BH  LIN   330
0000H  1139H  LIN   333
0000H  114CH  LIN   336
0000H  1154H  LIN   338
0000H  1158H  LIN   341
0000H  1163H  LIN   344
0000H  1167H  LIN   346
0000H  1171H  LIN   349
0000H  1182H  LIN   352
0000H  1191H  LIN   355
0000H  1199H  LIN   358
0000H  11A8H  LIN   360
0000H  11ACH  LIN   362
0000H  11B6H  LIN   365
0000H  11C7H  LIN   368
0000H  11D1H  LIN   371
0000H  11DDH  LIN   375
0000H  11E9H  LIN   377
0000H  11F7H  LIN   380
0000H  1200H  LIN   382
0000H  1204H  LIN   385
0000H  1215H  LIN   388
0000H  1223H  LIN   392
0000H  122CH  LIN   394
0000H  123DH  LIN   396
0000H  1252H  LIN   398
0000H  125EH  LIN   400
0000H  1268H  LIN   402
0000H  1273H  LIN   404
0000H  1278H  LIN   406
0000H  1282H  LIN   409
0000H  1293H  LIN   412
0000H  12A2H  LIN   415
0000H  12AAH  LIN   418
0000H  12C2H  LIN   421
0000H  12CBH  LIN   423
0000H  12CEH  LIN   425
0000H  12DFH  LIN   428
0000H  1302H  LIN   431
0000H  1308H  LIN   433
0000H  1312H  LIN   435
0000H  1318H  LIN   438
0000H  1323H  LIN   441
0000H  1339H  LIN   444
0000H  1340H  LIN   448
0000H  1345H  LIN   450

0000H  1088H  LIN   293
0000H  1097H  LIN   296
0000H  10A1H  LIN   299
0000H  10AAH  LIN   301
0000H  10ADH  LIN   303
0000H  10BEH  LIN   306
0000H  10CCH  LIN   310
0000H  10D6H  LIN   313
0000H  10F0H  LIN   315
0000H  10FCH  LIN   317
0000H  1101H  LIN   319
0000H  1112H  LIN   324
0000H  1117H  LIN   326
0000H  1121H  LIN   329
0000H  1132H  LIN   332
0000H  113EH  LIN   334
0000H  114FH  LIN   337
0000H  1154H  LIN   339
0000H  115EH  LIN   342
0000H  1167H  LIN   345
0000H  116AH  LIN   347
0000H  117BH  LIN   350
0000H  1189H  LIN   354
0000H  1195H  LIN   356
0000H  11A3H  LIN   359
0000H  11ACH  LIN   361
0000H  11AFH  LIN   363
0000H  11C0H  LIN   366
0000H  11CEH  LIN   370
0000H  11D6H  LIN   373
0000H  11E5H  LIN   376
0000H  11EDH  LIN   379
0000H  11FCH  LIN   381
0000H  1200H  LIN   383
0000H  120BH  LIN   387
0000H  121CH  LIN   390
0000H  1229H  LIN   393
0000H  1238H  LIN   395
0000H  1243H  LIN   397
0000H  1257H  LIN   399
0000H  1263H  LIN   401
0000H  126EH  LIN   403
0000H  1278H  LIN   405
0000H  1278H  LIN   407
0000H  128CH  LIN   410
0000H  129AH  LIN   414
0000H  12A6H  LIN   416
0000H  12B8H  LIN   420
0000H  12C7H  LIN   422
0000H  12CBH  LIN   424
0000H  12D5H  LIN   427
0000H  12E6H  LIN   430
0000H  1306H  LIN   432
0000H  130EH  LIN   434
0000H  1314H  LIN   436
0000H  131EH  LIN   439
0000H  1332H  LIN   443
0000H  133DH  LIN   445
0000H  1345H  LIN   449
0000H  1348H  LIN   451
```

```
0000H  134FH  LIN    453          0000H  1359H  LIN    454
0000H  1360H  LIN    456          0000H  1367H  LIN    458
0000H  136FH  LIN    459          0000H  1373H  LIN    460
0000H  1377H  LIN    462          0000H  137EH  LIN    463
0000H  1382H  LIN    464          0000H  1385H  LIN    465
0000H  1387H  LIN    466          0000H  1395H  LIN    468
0000H  139FH  LIN    469          0000H  13A4H  LIN    470
0000H  13A8H  LIN    471          0000H  13B8H  LIN    472
0000H  13ABH  LIN    473          0000H  13B2H  LIN    475
0000H  13B7H  LIN    476          0000H  13BAH  LIN    477
0000H  13C4H  LIN    478          0000H  13C8H  LIN    480
0000H  13D2H  LIN    481          0000H  13D7H  LIN    482
0000H  13DEH  LIN    484          0000H  13E5H  LIN    486
0000H  13E9H  LIN    487          0000H  13FCH  LIN    488
0000H  13F1H  LIN    490          0000H  13F8H  LIN    492
0000H  13FBH  LIN    493          0000H  1400H  LIN    495
0000H  140AH  LIN    496          0000H  140FH  LIN    497
0000H  1413H  LIN    498          0000H  1413H  LIN    499
0000H  1416H  LIN    501          0000H  141CH  LIN    502
0000H  1425H  LIN    503          0000H  1428H  LIN    504
0000H  1435H  LIN    505          0000H  1438H  LIN    506
0000H  143DH  LIN    507          0000H  1444H  LIN    508
0000H  1447H  LIN    509          0000H  144DH  LIN    511
0000H  1457H  LIN    512          0000H  145CH  LIN    513
0000H  1466H  LIN    514          0000H  1468H  LIN    515
0000H  1475H  LIN    516          0000H  147AH  LIN    517
0000H  1484H  LIN    518          0000H  1489H  LIN    519
0000H  1493H  LIN    520          0000H  1498H  LIN    521
0000H  14A5H  LIN    522          0000H  14AAH  LIN    523
0000H  14AFH  LIN    524          0000H  1488H  LIN    525
0000H  14C0H  LIN    526          0000H  14CAH  LIN    527
0000H  14D2H  LIN    528          0000H  14D6H  LIN    529
0000H  14D9H  LIN    531          0000H  14DFH  LIN    532
0000H  14E7H  LIN    533          0000H  14F6H  LIN    534
0000H  14FCH  LIN    535          0000H  093EH  LIN    536
```



SEARCH: SYMBOLS AND LINES

```
1000H  07A0H  SYM  MEMORY          1000H  01E2H  SYM  USEDDE
1000H  01E4H  SYM  FILESFOUND      1000H  01F8H  SYM  DIRLABEL
0000H  1502H  SYM  READCHAR        0000H  1511H  SYM  CHECKCONSOLESTAT
                                                     -US
0000H  1520H  SYM  SEARCHFIRST     STACK  0004H  SYM  FCBADDRESS
0000H  1530H  SYM  SEARCHNEXT      1000H  01E6H  SYM  BUFFCBADR
1000H  01E6H  BAS  BUFFCB          1000H  01E8H  SYM  FIRSTFIADR
1000H  01EAH  SYM  FIADR           1000H  01ECH  SYM  LASTFIADR
1000H  01EAH  BAS  FILEINFO        1000H  01EEH  SYM  SFCBADR
1000H  01EEH  BAS  DIRTYPE         1000H  01F9H  SYM  SFCUSPRESENT
1000H  01F0H  SYM  XIADR           1000H  01F0H  BAS  XFCBINFO
0000H  153FH  SYM  COMPARE         STACK  0008H  SYM  LENGTH
STACK  0006H  SYM  STR1ADR         STACK  0004H  SYM  STR2ADR
1000H  01FAH  SYM  I               STACK  0006H  BAS  STR1
STACK  0004H  BAS  STR2            0000H  1580H  SYM  MATCH
1000H  01FBH  SYM  I               1000H  01F2H  SYM  TEMP
1000H  07A0H  SYM  HASHTABLE       1000H  01F4H  SYM  HASHENTRYADR
1000H  01F4H  BAS  HASHENTRY       0000H  1648H  SYM  HASHLOOKUP
1000H  01FCH  SYM  I               1000H  01FDH  SYM  FOUND
1000H  01FEH  SYM  HASHINDEX       0000H  1705H  SYM  STOREFILEINFO
1000H  01FFH  SYM  I               1000H  0200H  SYM  J
1000H  0201H  SYM  DMAPCNT         1000H  01F6H  SYM  TEMP
0000H  183AH  SYM  STOREFILE       0000H  193CH  SYM  GETFILES
```

| 1000H | 0292H | SYM | DCNT |
|---|---|---|---|
| 0000H | 1502H | LIN | 32 |
| 0000H | 1511H | LIN | 34 |
| 0000H | 1514H | LIN | 73 |
| 0000H | 1520H | LIN | 75 |
| 0000H | 1530H | LIN | 78 |
| 0000H | 1533H | LIN | 80 |
| 0000H | 153FH | LIN | 90 |
| 0000H | 1554H | LIN | 93 |
| 0000H | 1574H | LIN | 95 |
| 0000H | 1580H | LIN | 97 |
| 0000H | 1583H | LIN | 100 |
| 0000H | 159DH | LIN | 104 |
| 0000H | 15AEH | LIN | 106 |
| 0000H | 15BBH | LIN | 109 |
| 0000H | 15DBH | LIN | 111 |
| 0000H | 1606H | LIN | 113 |
| 0000H | 1636H | LIN | 116 |
| 0000H | 1643H | LIN | 118 |
| 0000H | 1648H | LIN | 121 |
| 0000H | 1650H | LIN | 124 |
| 0000H | 166FH | LIN | 126 |
| 0000H | 167FH | LIN | 128 |
| 0000H | 1692H | LIN | 130 |
| 0000H | 169EH | LIN | 132 |
| 0000H | 16DDH | LIN | 134 |
| 0000H | 16EEH | LIN | 137 |
| 0000H | 16F6H | LIN | 140 |
| 0000H | 1701H | LIN | 142 |
| 0000H | 183AH | LIN | 146 |
| 0000H | 1853H | LIN | 148 |
| 0000H | 1869H | LIN | 150 |
| 0000H | 1879H | LIN | 152 |
| 0000H | 1895H | LIN | 154 |
| 0000H | 18ACH | LIN | 156 |
| 0000H | 18BCH | LIN | 158 |
| 0000H | 18C2H | LIN | 160 |
| 0000H | 18FAH | LIN | 163 |
| 0000H | 193AH | LIN | 166 |
| 0000H | 1711H | LIN | 169 |
| 0000H | 172BH | LIN | 172 |
| 0000H | 1734H | LIN | 174 |
| 0000H | 1743H | LIN | 176 |
| 0000H | 175BH | LIN | 178 |
| 0000H | 177EH | LIN | 180 |
| 0000H | 178BH | LIN | 185 |
| 0000H | 1799H | LIN | 189 |
| 0000H | 17B9H | LIN | 192 |
| 0000H | 17C4H | LIN | 194 |
| 0000H | 17DEH | LIN | 197 |
| 0000H | 17E2H | LIN | 202 |
| 0000H | 17FCH | LIN | 205 |
| 0000H | 1809H | LIN | 207 |
| 0000H | 182AH | LIN | 209 |
| 0000H | 1833H | LIN | 211 |
| 0000H | 183AH | LIN | 214 |
| 0000H | 193FH | LIN | 217 |
| 0000H | 1954H | LIN | 219 |
| 0000H | 1969H | LIN | 221 |
| 0000H | 1972H | LIN | 223 |

| 0000H | 1A26H | SYM | SEGMENT |
|---|---|---|---|
| 0000H | 1505H | LIN | 33 |
| 0000H | 1511H | LIN | 72 |
| 0000H | 1520H | LIN | 74 |
| 0000H | 1523H | LIN | 77 |
| 0000H | 1530H | LIN | 79 |
| 0000H | 153FH | LIN | 81 |
| 0000H | 1542H | LIN | 92 |
| 0000H | 1570H | LIN | 94 |
| 0000H | 157AH | LIN | 96 |
| 0000H | 1580H | LIN | 98 |
| 0000H | 1594H | LIN | 101 |
| 0000H | 15A3H | LIN | 105 |
| 0000H | 15B7H | LIN | 107 |
| 0000H | 15D0H | LIN | 110 |
| 0000H | 15EDH | LIN | 112 |
| 0000H | 1616H | LIN | 115 |
| 0000H | 163DH | LIN | 117 |
| 0000H | 1648H | LIN | 119 |
| 0000H | 164BH | LIN | 123 |
| 0000H | 165CH | LIN | 125 |
| 0000H | 1675H | LIN | 127 |
| 0000H | 1684H | LIN | 129 |
| 0000H | 1699H | LIN | 131 |
| 0000H | 16B4H | LIN | 133 |
| 0000H | 16E4H | LIN | 136 |
| 0000H | 16F4H | LIN | 139 |
| 0000H | 16FDH | LIN | 141 |
| 0000H | 1705H | LIN | 144 |
| 0000H | 183DH | LIN | 147 |
| 0000H | 1864H | LIN | 149 |
| 0000H | 186EH | LIN | 151 |
| 0000H | 187EH | LIN | 153 |
| 0000H | 18A5H | LIN | 155 |
| 0000H | 18B5H | LIN | 157 |
| 0000H | 18C0H | LIN | 159 |
| 0000H | 18C9H | LIN | 162 |
| 0000H | 192CH | LIN | 164 |
| 0000H | 1708H | LIN | 167 |
| 0000H | 171AH | LIN | 171 |
| 0000H | 172BH | LIN | 173 |
| 0000H | 1737H | LIN | 175 |
| 0000H | 1747H | LIN | 177 |
| 0000H | 1769H | LIN | 179 |
| 0000H | 1784H | LIN | 183 |
| 0000H | 1794H | LIN | 187 |
| 0000H | 17A8H | LIN | 191 |
| 0000H | 17BBH | LIN | 193 |
| 0000H | 1704H | LIN | 195 |
| 0000H | 17E0H | LIN | 200 |
| 0000H | 17EBH | LIN | 204 |
| 0000H | 1800H | LIN | 206 |
| 0000H | 181CH | LIN | 208 |
| 0000H | 1831H | LIN | 210 |
| 0000H | 1836H | LIN | 213 |
| 0000H | 193CH | LIN | 215 |
| 0000H | 1948H | LIN | 218 |
| 0000H | 1963H | LIN | 220 |
| 0000H | 196FH | LIN | 222 |
| 0000H | 197FH | LIN | 224 |

```
0000H  1984H  LIN    225          0000H  198EH  LIN    226
0000H  1994H  LIN    227          0000H  199CH  LIN    229
0000H  19A1H  LIN    230          0000H  19AFH  LIN    231
0000H  1981H  LIN    232          0000H  19B6H  LIN    233
0000H  19BDH  LIN    234          0000H  19C6H  LIN    235
0000H  19D2H  LIN    236          0000H  19E2H  LIN    237
0000H  19E8H  LIN    239          0000H  19F0H  LIN    240
0000H  19F4H  LIN    241          0000H  19FDH  LIN    242
0000H  1A08H  LIN    243          0000H  1A12H  LIN    245
0000H  1A1BH  LIN    247          0000H  1A22H  LIN    248
0000H  1A24H  LIN    253          0000H  1A24H  LIN    254
0000H  1A26H  LIN    255          0000H  1A29H  LIN    256
0000H  1A38H  LIN    258          0000H  1A3FH  LIN    259
0000H  1A42H  LIN    261          0000H  1502H  LIN    262

SORT:  SYMBOLS AND LINES
1000H  07A0H  SYM  MEMORY          1000H  0248H  SYM  SORTED
1000H  01E6H  BAS  BUFFCB          1000H  01EAH  BAS  FILEINFO
1000H  0204H  SYM  MIDADR          1000H  0204H  BAS  MIDFILEINFO
0000H  1A44H  SYM  MULT23          STACK  0004H  SYM  INDEX
0000H  1A69H  SYM  LESSTHAN        STACK  0006H  SYM  STR1ADR
STACK  0004H  SYM  STR2ADR         1000H  0249H  SYM  I
1000H  024AH  SYM  C1              1000H  024BH  SYM  C2
STACK  0006H  BAS  STR1            STACK  0004H  BAS  STR2
1000H  0206H  SYM  FIINDICESBASE   1000H  0206H  BAS  FIINDICES
0000H  1AABH  SYM  QSORT           STACK  0006H  SYM  L
STACK  0004H  SYM  R               1000H  0208H  SYM  I
1000H  020AH  SYM  J               1000H  020CH  SYM  TEMP
1000H  020EH  SYM  STACK           1000H  024CH  SYM  SP
0000H  1C20H  SYM  SORT            1000H  0246H  SYM  I
0000H  1A44H  LIN    10           0000H  1A47H  LIN    12
0000H  1A69H  LIN    13           0000H  1A69H  LIN    14
0000H  1A6CH  LIN    16           0000H  1A78H  LIN    17
0000H  1A96H  LIN    18           0000H  1A9FH  LIN    19
0000H  1AA5H  LIN    20           0000H  1AABH  LIN    21
0000H  1AABH  LIN    23           0000H  1AAEH  LIN    25
0000H  1AB3H  LIN    26           0000H  1AB9H  LIN    27
0000H  1ABFH  LIN    28           0000H  1AC9H  LIN    29
0000H  1AD9H  LIN    30           0000H  1AE0H  LIN    31
0000H  1AE4H  LIN    32           0000H  1AEEH  LIN    33
0000H  1AF2H  LIN    34           0000H  1AF5H  LIN    35
0000H  1B09H  LIN    36           0000H  1B15H  LIN    37
0000H  1B25H  LIN    38           0000H  1B34H  LIN    39
0000H  1B38H  LIN    40           0000H  1B3DH  LIN    41
0000H  1B3DH  LIN    42           0000H  1B4FH  LIN    43
0000H  1B5EH  LIN    44           0000H  1B65H  LIN    45
0000H  1B68H  LIN    46           0000H  1B6BH  LIN    47
0000H  1B74H  LIN    49           0000H  1B84H  LIN    50
0000H  1B8CH  LIN    51           0000H  1B8EH  LIN    52
0000H  1B92H  LIN    53           0000H  1B99H  LIN    54
0000H  1B9DH  LIN    56           0000H  1BA0H  LIN    57
0000H  1BB3H  LIN    59           0000H  1BB8H  LIN    61
0000H  1BC0H  LIN    62           0000H  1BCAH  LIN    63
0000H  1BD1H  LIN    65           0000H  1BD7H  LIN    66
0000H  1BD9H  LIN    68           0000H  1BE2H  LIN    70
0000H  1BEAH  LIN    71           0000H  1BF7H  LIN    72
0000H  1BFEH  LIN    74           0000H  1C04H  LIN    76
0000H  1C07H  LIN    77           0000H  1C07H  LIN    78
0000H  1C0EH  LIN    79           0000H  1C17H  LIN    80
0000H  1C1CH  LIN    81           0000H  1C20H  LIN    82
```

```
0000H  1C25H  LIN     84                0000H  1C2CH  LIN     85
0000H  1C33H  LIN     86                0000H  1C35H  LIN     87
0000H  1C44H  LIN     89                0000H  1C46H  LIN     90
0000H  1C4DH  LIN     92                0000H  1C5FH  LIN     93
0000H  1C6CH  LIN     94                0000H  1C73H  LIN     95
0000H  1C7FH  LIN     96                0000H  1C84H  LIN     97
0000H  1A44H  LIN     98

DISPLAY:  SYMBOLS AND LINES
1000H  07A0H  SYM  MEMORY               1000H  0258H  SYM  FILEDISPLAYED
1000H  01E6H  BAS  BUFFCB               1000H  024EH  SYM  CURFILE
1000H  01EAH  BAS  FILEINFO             1000H  01F0H  BAS  XFCBINFO
1000H  0206H  BAS  FIINDICES            0000H  1C86H  SYM  DIRECTCONSOLEIO
0000H  1C95H  SYM  WAITKEYPRESS         1000H  0259H  SYM  CHAR
1000H  025AH  SYM  GLOBALLINECOUNT      1000H  025BH  SYM  TOTALKBYTES
1000H  025EH  SYM  TOTALRECS            1000H  0261H  SYM  TOTAL1KBLOCKS
0000H  1CB1H  SYM  ADDTOTALS            1000H  0264H  SYM  FILESPERLINE
1000H  0252H  SYM  CURLINE              1000H  048CH  SYM  HDR
1000H  04B4H  SYM  HDRBARS              1000H  04DCH  SYM  HDRPU
1000H  04F3H  SYM  HDRXFCBBARS          1000H  051AH  SYM  HDRACCESS
1000H  052BH  SYM  HDRCREATE           0000H  1CE1H  SYM  DISPLAYFILEINFO
0000H  1E00H  SYM  DISPLAYXFCBINFO      1000H  0265H  SYM  FIRSTTITLE
0000H  1E94H  SYM  DISPLAYTITLE         0000H  1EF6H  SYM  SHORTDISPLAY
STACK  0004H  SYM  FNAMEADR             0000H  1F47H  SYM  TESTATT
STACK  0008H  SYM  CHAR                 STACK  0006H  SYM  OFF
STACK  0004H  SYM  ON                   0000H  1F74H  SYM  RIGHTATTRIBUTES
STACK  0004H  SYM  NAMEADR              STACK  0004H  BAS  NAME
0000H  1FA5H  SYM  SHORTDTR             1000H  0266H  SYM  DCNT
1000H  0254H  SYM  LASTPLUSONE          1000H  0256H  SYM  INDEX
0000H  202FH  SYM  GETNXTFILEINFO       1000H  0267H  SYM  RIGHTUSR
0000H  20BCH  SYM  SIZEDISPLAY          0000H  2136H  SYM  DISPLAYNODIRLABE
                                                          -L
0000H  21ECH  SYM  DISPLAYWITHDIRLA     0000H  2297H  SYM  DISPLAYFILES
                   -BEL
0000H  1C86H  LIN     90                0000H  1C89H  LIN     91
0000H  1C95H  LIN     92                0000H  1C95H  LIN     93
0000H  1C98H  LIN     95                0000H  1C9EH  LIN     96
0000H  1CA5H  LIN     99                0000H  1CACH  LIN    100
0000H  1CAFH  LIN    101                0000H  1CB1H  LIN    104
0000H  1CB4H  LIN    105                0000H  1CC2H  LIN    106
0000H  1CD1H  LIN    107                0000H  1CDFH  LIN    108
0000H  1CE1H  LIN    117                0000H  1CE4H  LIN    118
0000H  1CEFH  LIN    119                0000H  1CF2H  LIN    120
0000H  1D03H  LIN    121                0000H  1D09H  LIN    122
0000H  1D0CH  LIN    123                0000H  1D1BH  LIN    124
0000H  1D1EH  LIN    125                0000H  1D2EH  LIN    126
0000H  1D33H  LIN    127                0000H  1D3AH  LIN    128
0000H  1D3DH  LIN    129                0000H  1D4DH  LIN    130
0000H  1D52H  LIN    131                0000H  1D59H  LIN    132
0000H  1D5CH  LIN    133                0000H  1D65H  LIN    135
0000H  1D75H  LIN    136                0000H  1D7AH  LIN    137
0000H  1D81H  LIN    138                0000H  1D83H  LIN    140
0000H  1D93H  LIN    141                0000H  1D9BH  LIN    142
0000H  1D9EH  LIN    143                0000H  1DABH  LIN    144
0000H  1DB3H  LIN    145                0000H  1DB6H  LIN    146
0000H  1DC3H  LIN    147                0000H  1DCBH  LIN    148
0000H  1DCEH  LIN    149                0000H  1DDBH  LIN    150
0000H  1DE3H  LIN    151                0000H  1DE6H  LIN    152
0000H  1DF3H  LIN    153                0000H  1DFBH  LIN    154
0000H  1DFEH  LIN    156                0000H  1E00H  LIN    157
```

```
0000H  1E03H  LIN   158
0000H  1E10H  LIN   161
0000H  1E22H  LIN   163
0000H  1E31H  LIN   165.
0000H  1E40H  LIN   167
0000H  1E4CH  LIN   169
0000H  1E5FH  LIN   171
0000H  1E73H  LIN   173
0000H  1E79H  LIN   175
0000H  1E92H  LIN   178
0000H  1E97H  LIN   181
0000H  1EA6H  LIN   183
0000H  1EB2H  LIN   185
0000H  1EC2H  LIN   187
0000H  1ECFH  LIN   190
0000H  1EE6H  LIN   193
0000H  1EEFH  LIN   195
0000H  1EF6H  LIN   197
0000H  1F0AH  LIN   201
0000H  1F1AH  LIN   204
0000H  1F20H  LIN   208
0000H  1F2DH  LIN   210
0000H  1F32H  LIN   212
0000H  1F3FH  LIN   214
0000H  1F47H  LIN   216
0000H  1F5AH  LIN   219
0000H  1F6AH  LIN   221
0000H  1F74H  LIN   223
0000H  1F77H  LIN   226
0000H  1FA5H  LIN   228
0000H  1FADH  LIN   231
0000H  1FBCH  LIN   233
0000H  1FD4H  LIN   235
0000H  200EH  LIN   237
0000H  2028H  LIN   239
0000H  202DH  LIN   241
0000H  2032H  LIN   245
0000H  203EH  LIN   248
0000H  2055H  LIN   250
0000H  207EH  LIN   252
0000H  2084H  LIN   254
0000H  2093H  LIN   256
0000H  209AH  LIN   259
0000H  20BAH  LIN   263
0000H  20BFH  LIN   265
0000H  20CDH  LIN   267
0000H  20D8H  LIN   269
0000H  210CH  LIN   272
0000H  2128H  LIN   274
0000H  2132H  LIN   277
0000H  2136H  LIN   279
0000H  213EH  LIN   281
0000H  2178H  LIN   284
0000H  218EH  LIN   287
0000H  219EH  LIN   290
0000H  21A4H  LIN   293
0000H  21AAH  LIN   295
0000H  21B4H  LIN   297
0000H  21BEH  LIN   299
0000H  21C8H  LIN   301

0000H  1E0DH  LIN   159
0000H  1E1AH  LIN   162
0000H  1E27H  LIN   164
0000H  1E36H  LIN   166
0000H  1E45H  LIN   168
0000H  1E4FH  LIN   170
0000H  1E6CH  LIN   172
0000H  1E76H  LIN   174
0000H  1E8BH  LIN   176
0000H  1E94H  LIN   180
0000H  1E9EH  LIN   182
0000H  1EAFH  LIN   184
0000H  1EB9H  LIN   186
0000H  1EC3H  LIN   188
0000H  1ED6H  LIN   191
0000H  1EE9H  LIN   194
0000H  1EF4H  LIN   196
0000H  1EF9H  LIN   199
0000H  1F17H  LIN   203
0000H  1F1DH  LIN   207
0000H  1F24H  LIN   209
0000H  1F2FH  LIN   211
0000H  1F39H  LIN   213
0000H  1F43H  LIN   215
0000H  1F4AH  LIN   218
0000H  1F5AH  LIN   220
0000H  1F6EH  LIN   222
0000H  1F74H  LIN   224
0000H  1FA5H  LIN   227
0000H  1FA8H  LIN   230
0000H  1FB2H  LIN   232
0000H  1FC3H  LIN   234
0000H  2007H  LIN   236
0000H  201DH  LIN   238
0000H  202DH  LIN   240
0000H  202FH  LIN   243
0000H  2037H  LIN   246
0000H  2045H  LIN   249
0000H  2079H  LIN   251
0000H  2084H  LIN   253
0000H  209DH  LIN   255
0000H  2095H  LIN   258
0000H  20BAH  LIN   261
0000H  20BCH  LIN   264
0000H  20C6H  LIN   266
0000H  2002H  LIN   268
0000H  2109H  LIN   271
0000H  2117H  LIN   273
0000H  212FH  LIN   276
0000H  2134H  LIN   278
0000H  2139H  LIN   280
0000H  214AH  LIN   282
0000H  2189H  LIN   285
0000H  2198H  LIN   289
0000H  21A2H  LIN   291
0000H  21A7H  LIN   294
0000H  21ADH  LIN   296
0000H  21B7H  LIN   298
0000H  21C1H  LIN   300
0000H  21CBH  LIN   302
```

```
0000H  21D2H  LIN    303
0000H  21DAH  LIN    307
0000H  21E1H  LIN    309
0000H  21E7H  LIN    312
0000H  21ECH  LIN    314
0000H  21F4H  LIN    316
0000H  222EH  LIN    319
0000H  223EH  LIN    322
0000H  2244H  LIN    324
0000H  2252H  LIN    326
0000H  225EH  LIN    328
0000H  2268H  LIN    330
0000H  2276H  LIN    332
0000H  227EH  LIN    335
0000H  2284H  LIN    337
0000H  228CH  LIN    339
0000H  2292H  LIN    342
0000H  2297H  LIN    344
0000H  22A3H  LIN    346
0000H  22B5H  LIN    348
0000H  22C7H  LIN    350
0000H  22D0H  LIN    352
0000H  22DEH  LIN    355
0000H  22EEH  LIN    358
0000H  22F8H  LIN    360
0000H  2337H  LIN    363
0000H  233EH  LIN    366
0000H  2343H  LIN    370
0000H  234FH  LIN    372
0000H  2354H  LIN    374
0000H  236BH  LIN    377
0000H  2384H  LIN    380
0000H  238CH  LIN    383
0000H  2392H  LIN    386
0000H  23A4H  LIN    388
0000H  23B1H  LIN    390
0000H  23C3H  LIN    392
0000H  23D8H  LIN    394
0000H  23EAH  LIN    396
0000H  23F9H  LIN    398
0000H  240AH  LIN    400
0000H  2422H  LIN    403
0000H  2430H  LIN    407
0000H  2436H  LIN    409
0000H  243FH  LIN    413
0000H  244DH  LIN    415
0000H  1C86H  LIN    418

0000H  21D5H  LIN    304
0000H  21DDH  LIN    308
0000H  21E4H  LIN    311
0000H  21EAH  LIN    313
0000H  21EFH  LIN    315
0000H  2200H  LIN    317
0000H  223BH  LIN    321
0000H  2241H  LIN    323
0000H  2246H  LIN    325
0000H  2259H  LIN    327
0000H  2265H  LIN    329
0000H  226FH  LIN    331
0000H  2278H  LIN    334
0000H  2281H  LIN    336
0000H  2288H  LIN    338
0000H  228FH  LIN    341
0000H  2295H  LIN    343
0000H  229AH  LIN    345
0000H  22AFH  LIN    347
0000H  22BEH  LIN    349
0000H  22C0H  LIN    351
0000H  22D7H  LIN    354
0000H  22E1H  LIN    357
0000H  22F3H  LIN    359
0000H  22FFH  LIN    362
0000H  2337H  LIN    365
0000H  2341H  LIN    368
0000H  234AH  LIN    371
0000H  2352H  LIN    373
0000H  235AH  LIN    375
0000H  237EH  LIN    379
0000H  238AH  LIN    381
0000H  238FH  LIN    384
0000H  2399H  LIN    387
0000H  23AAH  LIN    389
0000H  23BCH  LIN    391
0000H  23D1H  LIN    393
0000H  23E3H  LIN    395
0000H  23F3H  LIN    397
0000H  23FCH  LIN    399
0000H  2410H  LIN    401
0000H  2429H  LIN    405
0000H  2433H  LIN    408
0000H  2430H  LIN    411
0000H  2444H  LIN    414
0000H  2450H  LIN    417
```



SER. #_____

```
UTILITY:  SYMBOLS AND LINES
1000H  07A0H  SYM  MEMORY
STACK  0006H  SYM  BYTE3ADR
STACK  0006H  BAS  B3
0000H  246FH  SYM  ADD3BYTE3
STACK  0004H  SYM  NUMB
STACK  0006H  BAS  TOTAL
STACK  0004H  SYM  BYTE3ADR
STACK  0004H  BAS  TEMP1
0000H  24A8H  SYM  PRINTCHAR
0000H  24BEH  SYM  PRINT
0000H  24CEH  SYM  PRINTB

0000H  2452H  SYM  ADD3BYTE
STACK  0004H  SYM  NUM
1000H  0268H  SYM  TEMP
STACK  0006H  SYM  TOTAL3
STACK  0004H  BAS  NUM
0000H  248DH  SYM  SHR3BYTE
STACK  0004H  BAS  B3
1000H  026AH  SYM  TEMP2
STACK  0004H  SYM  CHAR
STACK  0004H  SYM  STRINGADR
0000H  24D9H  SYM  CRLF
```

```
0000H  24EAH  SYM  PRINTFN              STACK  0004H  SYM  PARAMADR
STACK  0004H  BAS  FILENAME             1000H  026BH  SYM  I
0000H  253FH  SYM  PDECIMAL             STACK  0008H  SYM  V
STACK  0006H  SYM  PREC                 STACK  0004H  SYM  ZEROSUP
1000H  026CH  SYM  D                    0000H  2595H  SYM  PBYTE
STACK  0006H  SYM  BYTE3ADR             STACK  0004H  SYM  PREC
STACK  0006H  BAS  B3                   1000H  026DH  SYM  T
0000H  2452H  LIN      3                0000H  2455H  LIN      5
0000H  2458H  LIN      6                0000H  2468H  LIN      7
0000H  2463H  LIN      8                0000H  246FH  LIN      9
0000H  2472H  LIN     11                0000H  2478H  LIN     12
0000H  2489H  LIN     13                0000H  248DH  LIN     14
0000H  2490H  LIN     16                0000H  249FH  LIN     17
0000H  24A2H  LIN     18                0000H  24A4H  LIN     19
0000H  24A7H  LIN     20                0000H  24ABH  LIN     25
0000H  24AEH  LIN     27                0000H  24BAH  LIN     28
0000H  24BEH  LIN     29                0000H  24C1H  LIN     31
0000H  24CAH  LIN     32                0000H  24CEH  LIN     33
0000H  24D1H  LIN     34                0000H  24D7H  LIN     35
0000H  24D9H  LIN     36                0000H  24DCH  LIN     37
0000H  24E2H  LIN     38                0000H  24E8H  LIN     39
0000H  24EAH  LIN     40                0000H  24EDH  LIN     42
0000H  24F9H  LIN     43                0000H  250BH  LIN     44
0000H  2511H  LIN     45                0000H  2517H  LIN     46
0000H  2523H  LIN     47                0000H  2535H  LIN     48
0000H  2538H  LIN     49                0000H  253FH  LIN     50
0000H  2542H  LIN     52                0000H  2549H  LIN     53
0000H  2555H  LIN     54                0000H  255FH  LIN     55
0000H  2568H  LIN     56                0000H  257DH  LIN     57
0000H  2582H  LIN     59                0000H  2586H  LIN     60
0000H  258FH  LIN     62                0000H  2591H  LIN     63
0000H  2595H  LIN     64                0000H  2598H  LIN     66
0000H  25A1H  LIN     68                0000H  25B0H  LIN     69
0000H  25BDH  LIN     70                0000H  25B9H  LIN     72
0000H  25C2H  LIN     73                0000H  25CCH  LIN     74
0000H  25CFH  LIN     75                0000H  25D9H  LIN     76
0000H  25DBH  LIN     77                0000H  25EAH  LIN     79
0000H  2452H  LIN     80

DPB86:  SYMBOLS AND LINES
1000H  07A0H  SYM  MEMORY               1000H  027ZH  SYM  KPERBLOCK
1000H  026EH  SYM  DPBBASE              1000H  026EH  BAS  DPBARRAY
0000H  25F0H  SYM  DPBBYTE              STACK  0004H  SYM  PARAM
0000H  2605H  SYM  DPBWORD              STACK  0004H  SYM  PARAM
0000H  2628H  SYM  BASEDPB              0000H  25F0H  LIN     11
0000H  25F3H  LIN     13                0000H  2605H  LIN     14
0000H  2605H  LIN     15                0000H  2608H  LIN     17
0000H  2628H  LIN     18                0000H  2628H  LIN     19
0000H  262BH  LIN     20                0000H  263DH  LIN     21
0000H  264CH  LIN     22                0000H  25F0H  LIN     23

TIMESTAMP:  SYMBOLS AND LINES
1000H  07A0H  SYM  MEMORY               1000H  0274H  SYM  TODADR
1000H  0274H  BAS  TOD                  1000H  0276H  SYM  STRINGADR
1000H  0276H  BAS  STRING               1000H  028DH  SYM  INDEX
0000H  264EH  SYM  EMITCHAR             STACK  0004H  SYM  C
0000H  266AH  SYM  EMITN                STACK  0004H  SYM  A
STACK  0004H  BAS  C                    0000H  2692H  SYM  EMITBCD
STACK  0004H  SYM  B                    0000H  26A2H  SYM  EMITBCDPAIR
STACK  0004H  SYM  B                    0000H  26BDH  SYM  EMITCOLON
```

```
STACK  0004H  SYM  B
STACK  0004H  SYM  B
STACK  0004H  SYM  B
0000H  270AH  SYM  LEAPDAYS
STACK  0004H  SYM  M
1000H  0278H  SYM  WORDVALUE
1000H  0282H  SYM  LSD
STACK  0004H  SYM  VAL
1000H  0284H  SYM  DAY
1000H  0286H  SYM  HRS
1000H  0288H  SYM  SEC
STACK  0006H  SYM  A
0000H  278EH  SYM  COMPUTEYEAR
1000H  0289H  SYM  WEEKDAY
1000H  028AH  SYM  LEAPBIAS
1000H  028BH  SYM  DATETEST
0000H  27FEH  SYM  GETDATETIME
0000H  28C7H  SYM  TODASCII
1000H  027EH  SYM  RET
0000H  28FDH  SYM  DISPLAYTIMESTAMP
1000H  02A7H  SYM  I
0000H  264EH  LIN     13
0000H  2666H  LIN     16
0000H  266DH  LIN     20
0000H  2689H  LIN     22
0000H  268EH  LIN     24
0000H  2695H  LIN     27
0000H  26A2H  LIN     29
0000H  26B0H  LIN     32
0000H  26BDH  LIN     34
0000H  26C6H  LIN     37
0000H  26D0H  LIN     39
0000H  26E3H  LIN     42
0000H  26F7H  LIN     44
0000H  2700H  LIN     47
0000H  270AH  LIN     50
0000H  2717H  LIN     54
0000H  272FH  LIN     56
0000H  2736H  LIN     59
0000H  2747H  LIN     62
0000H  2756H  LIN     64
0000H  2759H  LIN     67
0000H  277DH  LIN     70
0000H  278EH  LIN     73
0000H  2791H  LIN     76
0000H  2796H  LIN     78
0000H  27A3H  LIN     80
0000H  27B2H  LIN     82
0000H  27BBH  LIN     84
0000H  27C1H  LIN     86
0000H  27C6H  LIN     89
0000H  27D2H  LIN     91
0000H  27E3H  LIN     93
0000H  27FCH  LIN     95
0000H  27FEH  LIN     98
0000H  2808H  LIN    100
0000H  2817H  LIN    102
0000H  2829H  LIN    104
0000H  2831H  LIN    106
0000H  2844H  LIN    108
0000H  26D9H  SYM  DAYSINMAIO
0000H  26F7H  SYM  DAYSLAST
1000H  0A5AH  SYM  MONTHDAYS
STACK  0006H  SYM  Y
1000H  0281H  SYM  YP
0000H  2736H  SYM  GETNEXTDIGIT
0000H  2756H  SYM  BCD
1000H  0283H  SYM  MONTH
1000H  0285H  SYM  YEAR
1000H  0287H  SYM  MIN
0000H  2770H  SYM  BCDPAIR
STACK  0004H  SYM  B
1000H  027AH  SYM  YEARLENGTH
1000H  0672H  SYM  DAYLIST
0000H  27C3H  SYM  COMPUTEMONTH
1000H  027CH  SYM  TESTVALUE
0000H  2868H  SYM  SMTIDATETIME
STACK  0004H  SYM  PARAMETER
1000H  028CH  SYM  LCLTOD
STACK  0004H  SYM  TSADR
1000H  02A8H  SYM  LASTDATABYTE
0000H  2651H  LIN     15
0000H  266AH  LIN     17
0000H  2675H  LIN     21
0000H  268CH  LIN     23
0000H  2692H  LIN     25
0000H  269EH  LIN     28
0000H  26A5H  LIN     31
0000H  2689H  LIN     33
0000H  26C0H  LIN     36
0000H  26CCH  LIN     38
0000H  26D3H  LIN     41
0000H  26F3H  LIN     43
0000H  26FAH  LIN     46
0000H  2706H  LIN     48
0000H  270DH  LIN     53
0000H  272BH  LIN     55
0000H  2736H  LIN     57
0000H  2739H  LIN     61
0000H  2751H  LIN     63
0000H  2756H  LIN     65
0000H  2770H  LIN     68
0000H  2780H  LIN     72
0000H  278EH  LIN     74
0000H  2796H  LIN     77
0000H  279CH  LIN     79
0000H  27A9H  LIN     81
0000H  27B4H  LIN     83
0000H  27BFH  LIN     85
0000H  27C3H  LIN     88
0000H  27C8H  LIN     90
0000H  27DEH  LIN     92
0000H  27FAH  LIN     94
0000H  27FCH  LIN     96
0000H  2801H  LIN     99
0000H  2811H  LIN    101
0000H  281DH  LIN    103
0000H  282CH  LIN    105
0000H  283FH  LIN    107
0000H  2847H  LIN    109
```

```
0000H  2862H  LIN     110          0000H  2866H  LIN     111
0000H  2868H  LIN     112          0000H  2868H  LIN     113
0000H  2874H  LIN     115          0000H  2878H  LIN     116
0000H  2880H  LIN     118          0000H  2893H  LIN     119
0000H  289AH  LIN     120          0000H  28A1H  LIN     121
0000H  28A7H  LIN     122          0000H  28AEH  LIN     123
0000H  28B5H  LIN     124          0000H  28BEH  LIN     125
0000H  28C5H  LIN     126          0000H  28C7H  LIN     127
0000H  28CAH  LIN     130          0000H  28D0H  LIN     131
0000H  28D6H  LIN     132          0000H  28DFH  LIN     133
0000H  28E9H  LIN     135          0000H  28ECH  LIN     136
0000H  28F1H  LIN     137          0000H  28F4H  LIN     138
0000H  28F6H  LIN     139          0000H  28F9H  LIN     140
0000H  28FDH  LIN     142          0000H  2900H  LIN     144
0000H  2905H  LIN     145          0000H  2913H  LIN     146
0000H  291AH  LIN     147          0000H  2926H  LIN     148
0000H  2933H  LIN     149          0000H  2939H  LIN     150
0000H  264EH  LIN     152

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

MODULE START ADDRESS  PARAGRAPH = 0000H  OFFSET = 0102H
SEGMENT MAP

START     STOP      LENGTH ALIGN NAME            CLASS

00000H    0293CH    293DH    G   CODE            CODE
10000H    10107H    0108H    G   DATS            DATA
10108H    102A8H    01A1H    W   DATA            DATA
102AAH    1068DH    03E4H    W   CONST           CONST
1068EH    10797H    010AH    W   STACK           STACK
107A0H    107A0H    0000H    G   ??SEG
107A0H    107A0H    0000H    W   MEMORY          MEMORY


GROUP MAP

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