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


            $ TITLE('SHOW 2.1: Show Disk Data')
            $ COMPACT


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

                                    * * *  SHOW  * * *

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

            /* Modfification log:
                Oct 82 whf
                Version changes CCPM86 v2.0
                Nov 82 F.Borda
            */

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


  1           show:
              do;
  2   1       declare
                  cpmversion literally '20h', /* requires 2.0 cp/m */
                  cpm3       literally '30h';


  3   1       declare copyright(*) byte data
                  (' Copyright (c) 1983, Digital Research ');
```

```
 4   1      declare verdate (*) byte data('10/19/82');
 5   1      declare version (*) byte data ('SHOW 2.1',0);


            /*      m p / m    s h o w    c o m m a n d      */

            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */
            /* show    show    show    show    show    show */

            /*
                  copyright(c) 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983
                  digital research
                  box 579
                  pacific grove, ca
                  93950

             */

            /* modified 10/30/78 to fix the space computation */
            /* modified 01/28/79 to remove despool dependencies */
            /* modified 07/26/79 to operate under cp/m 2.0 */
            /* modified 01/20/80 by Thomas Rolander */
            /* show created 05/19/81 */
            /* show modified for MP/M-86 9/4/81, changes in upper case       */
            /* modified 06/23/82 by Bill Fitler for CCP/M-86 */

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

                          * * *  MP/M INTERFACE * * *

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

```
                /* function call 32 returns the address of the disk parameter
                block for the currently selected disk, which consists of:
                        scptrk      (2 by) number of sectors per track
                        blkshf      (1 by) log2 of blocksize (2**blkshf=blksize)
                        blkmsk      (1 by) 2**blkshf-1
                        extmsk      (1 by) logical/physical extents
                        maxall      (2 by) max alloc number
                        dirmax      (2 by) size of directory-1
                        dirblk      (2 by) reservation bits for directory
                        chksiz      (2 by) size of checksum vector
                        offset      (2 by) offset for operating system
                */

  6   1         declare
                    maxb  address external,       /* addr field of jmp BDOS */
                    fcb (33) byte external,       /* default file control block */
                    buff(128)byte external,       /* default buffer */
                    buffa literally '.buff',      /* default buffer */
                    fcba  literally '.fcb',      /* default file control block */
                    dolla literally '.fcb(6dh-5ch)',      /* dollar sign position */
                    parma literally '.fcb(6eh-5ch)',      /* parameter, if sent */
                    rreca literally '.fcb(7dh-5ch)',      /* random record 7d,7e,7f */
                    rreco literally '.fcb(7fh-5ch)',      /* high byte of random overflow */
                    sectorlen literally '128',    /* sector length */
                    memsize literally 'maxb',     /* end of memory */
                    rrec address at(rreca),       /* random record address */
                    rovf byte at(rreco),          /* overflow on getfile */
                    doll byte at(dolla),          /* dollar parameter */
                    parm byte at(parma),          /* parameter */
                    user$code byte,               /* current user code */
                    cversion byte,                /* cpm version # */
                    cdisk byte,                   /* current disk  */
                    DPBPTR POINTER,               /* disk parameter block address */
                    DPB BASED DPBPTR structure
                    (spt address, bls byte, bms byte, exm byte, mxa address,
                     dmx address, dbl address, cks address, ofs address),
                    scptrk literally 'dpb.spt',
                    blkshf literally 'dpb.bls',
                    blkmsk literally 'dpb.bms',
                    extmsk literally 'dpb.exm',
                    maxall literally 'dpb.mxa',
                    dirmax literally 'dpb.dmx',
                    dirblk literally 'dpb.dbl',
                    chksiz literally 'dpb.cks',
                    offset literally 'dpb.ofs';


  7   1         mon1: procedure(f,a) external;
  8   2             declare f byte, a address;
  9   2             end mon1;

 10   1         mon2: procedure(f,a) byte external;
 11   2             declare f byte, a address;
 12   2             end mon2;
```

```
                /* declare mon3 literally 'mon2a'; */

  13   1        mon3: procedure(f,a) address external;
  14   2            declare f byte, a address;
  15   2            end mon3;

  16   1        MON4: PROCEDURE(F,A) POINTER EXTERNAL;
  17   2            DECLARE F BYTE, A ADDRESS;
  18   2            END MON4;


  19   1        declare alloca address,
                    /* alloca is the address of the disk allocation vector */
                    alloc based alloca (1024) byte;  /* allocation vector */

  20   1        declare
                    true literally '1',
                    false literally '0',
                    forever literally 'while true',
                    lit literally 'literally',
                    proc literally 'procedure',
                    dcl literally 'declare',
                    addr literally 'address',
                    cr literally '13',
                    lf literally '10';

  21   1        printchar: procedure(char);
  22   2            declare char byte;
  23   2            call mon1(2,char);
  24   2            end printchar;

  25   1        printb: procedure;
                    /* print blank character */
  26   2            call printchar(' ');
  27   2            end printb;

  28   1        printx: procedure(a);
  29   2            declare a address;
  30   2            declare s based a byte;
  31   2                do while s <> 0;
  32   3                call printchar(s);
  33   3                a = a + 1;
  34   3                end;
  35   2            end printx;

  36   1        break: procedure byte;
  37   2            return mon2(11,0); /* console ready */
  38   2            end break;

  39   1        crlf: procedure;
  40   2            call printchar(cr);
  41   2            call printchar(lf);
  42   2            if break then
  43   2            do; call mon1 (1,0);  /* read character */
  45   3                call mon1 (0,0);  /* system reset */
```

```
46  3           end;
47  2           end crlf;

48  1       print: procedure(a);
49  2           declare a address;
                /* print the string starting at address a until the
                next 0 is encountered */
50  2           call crlf;
51  2           call printx(a);
52  2           end print;

53  1       declare dcnt byte;

54  1       get$version: procedure byte;
                /* returns current cp/m version # */
55  2           return mon2(12,0);
56  2           end get$version;

57  1       select: procedure(d);
58  2           declare d byte;
59  2           call mon1(14,d);
60  2           end select;

61  1       open: procedure(fcb);
62  2           declare fcb address;
63  2           dcnt = mon2(15,fcb);
64  2           end open;

65  1       declare anything byte;
66  1       declare dirbuf (128) byte;
67  1       declare user(16)     byte,          /* any files in user i? */
                 used(16)     address,       /* # files in user i */
                 nSFCB        address,       /* # SFCB's */
                 free$dir     address;       /* # free directory entries */


68  1       check$user: procedure;
69  2           do forever;
70  3               if anything then return;
72  3               if dcnt = 0ffh then return;
74  3               if dirbuf(ror (dcnt,3) and 110$0000b) = user$code
                        then return;
76  3               dcnt = mon2(18,0);
77  3           end;
78  2           end check$user;

79  1       search: procedure(fcb);
80  2           declare fcb address;
81  2           declare fcb0 based fcb byte;
82  2           anything = (fcb0 = '?');
83  2           dcnt = mon2(17,fcb);
84  2           call check$user;
85  2           end search;

86  1       searchn: procedure;
87  2           dcnt = mon2(18,0);
88  2           call check$user;
```

```
 89   2          end searchn;

 90   1      cselect: procedure byte;
                 /* return current disk number */
 91   2          return mon2(25,0);
 92   2          end cselect;

 93   1      setdma: procedure(dma);
 94   2          declare dma address;
 95   2          call mon1(26,dma);
 96   2          end setdma;

 97   1      getalloca: procedure address;
                 /* get base address of alloc vector */
 98   2          return mon3(27,0);
 99   2          end getalloca;

100   1      getlogin: procedure address;
                 /* get the login vector */
101   2          return mon3(24,0);
102   2          end getlogin;

103   1      writeprot: procedure;
                 /* write protect the current disk */
104   2          call mon1(28,0);
105   2          end writeprot;

106   1      getrodisk: procedure address;
                 /* get the read-only disk vector */
107   2          return mon3(29,0);
108   2          end getrodisk;

109   1      setind: procedure;
                 /* set file indicators for current fcb */
110   2          call mon1(30,fcba);
111   2          end setind;

112   1      set$dpb: procedure;
                 /* set disk parameter block values */
113   2          DPBPTR = MON4(31,0);    /* base of dpb */
114   2          end set$dpb;

115   1      getuser: procedure byte;
                 /* return current user number */
116   2          return mon2(32,0ffh);
117   2          end getuser;

118   1      setuser: procedure(user);
119   2          declare user byte;
120   2          call mon1(32,user);
121   2          end setuser;

122   1      getfilesize: procedure(fcb);
123   2          declare fcb address;
124   2          call mon1(35,fcb);
125   2          end getfilesize;
```

```
126   1       getfreesp: procedure(d);
127   2           declare d byte;

128   2           call mon1(46,d);
129   2           end getfreesp;

130   1       getlbl: procedure(d) byte;
131   2           declare d byte;

132   2           return mon2(101,d);
133   2           end getlbl;

134   1         declare
                  parse$fn structure (
                    buff$adr  address,
                    fcb$adr   address),
                  delimiter based parse$fn.buff$adr byte;

135   1         parse: procedure address;
136   2           return mon3(152,.parse$fn);
137   2           end parse;

138   1       terminate: procedure;
139   2           call mon1 (0,0);                /* system reset */
140   2           end terminate;


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

                          Time & Date ASCII Conversion Code

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

141   1       declare tod$adr address;
142   1       declare tod based tod$adr structure (
                  opcode byte,
                  date address,
                  hrs byte,
                  min byte,
                  sec byte,
                  ASCII (21) byte );

143   1       declare string$adr address;
144   1       declare string based string$adr (1) byte;
145   1       declare index byte;

146   1       emitchar: procedure(c);
147   2           declare c byte;
148   2           string(index := index + 1) = c;
149   2           end emitchar;

150   1       emitn: procedure(a);
151   2           declare a address;
152   2           declare c based a byte;
153   2           do while c <> '$';
154   3             string(index := index + 1) = c;
155   3             a = a + 1;
```

```
156   3          end;
157   2          end emitn;


158   1      emit$bcd: procedure(b);
159   2          declare b byte;
160   2          call emitchar('0'+b);
161   2          end emit$bcd;

162   1      emit$bcd$pair: procedure(b);
163   2          declare b byte;
164   2          call emit$bcd(shr(b,4));
165   2          call emit$bcd(b and 0fh);
166   2          end emit$bcd$pair;

167   1      emit$colon: procedure(b);
168   2          declare b byte;
169   2          call emit$bcd$pair(b);
170   2          call emitchar(':');
171   2          end emit$colon;

172   1      emit$bin$pair: procedure(b);
173   2          declare b byte;
174   2          call emit$bcd(b/10);    /* makes garbage if not < 10 */
175   2          call emit$bcd(b mod 10);
176   2          end emit$bin$pair;

177   1      emit$slant: procedure(b);
178   2          declare b byte;
179   2          call emit$bin$pair(b);
180   2          call emitchar('/');
181   2          end emit$slant;

182   1      declare chr byte;

183   1      gnc: procedure;
                  /* get next command byte */
184   2          if chr = 0 then return;
186   2          if index = 20 then
187   2          do;
188   3            chr = 0;
189   3            return;
190   3          end;
191   2          chr = string(index := index + 1);
192   2          end gnc;

193   1      deblank: procedure;
194   2                do while chr = ' ';
195   3                call gnc;
196   3                end;
197   2          end deblank;

198   1      numeric: procedure byte;
                  /* test for numeric */
199   2          return (chr - '0') < 10;
200   2          end numeric;
```

```
201   1     scan$numeric: procedure(lb,ub) byte;
202   2         declare (lb,ub) byte;
203   2         declare b byte;
204   2         b = 0;
205   2         call deblank;
206   2         if not numeric then call terminate;
208   2             do while numeric;
209   3             if (b and 1110$0000b) <> 0 then call terminate;
211   3             b = shl(b,3) + shl(b,1); /* b = b * 10 */
212   3             if carry then call terminate;
214   3             b = b + (chr - '0');
215   3             if carry then call terminate;
217   3             call gnc;
218   3             end;
219   2         if (b < lb) or (b > ub) then call terminate;
221   2         return b;
222   2         end scan$numeric;

223   1     scan$delimiter: procedure(d,lb,ub) byte;
224   2         declare (d,lb,ub) byte;
225   2         call deblank;
226   2         if chr <> d then call terminate;
228   2         call gnc;
229   2         return scan$numeric(lb,ub);
230   2         end scan$delimiter;

231   1     declare
            base$year lit '78',    /* base year for computations */
            base$day  lit '0',     /* starting day for base$year 0..6 */
            month$size (*) byte data
            /* jan feb mar apr may jun jul aug sep oct nov dec */
            (   31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31),
            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);

232   1     leap$days: procedure(y,m) byte;
233   2         declare (y,m) byte;
            /* compute days accumulated by leap years */
234   2         declare yp byte;
235   2         yp = shr(y,2); /* yp = y/4 */
236   2         if (y and 11b) = 0 and month$days(m) < 59 then
                /* y not 00, y mod 4 = 0, before march, so not leap yr */
237   2             return yp - 1;
            /* otherwise, yp is the number of accumulated leap days */
238   2         return yp;
239   2         end leap$days;

240   1     declare word$value address;

241   1     get$next$digit: procedure byte;
            /* get next lsd from word$value */
242   2         declare lsd byte;
243   2         lsd = word$value mod 10;
244   2         word$value = word$value / 10;
245   2         return lsd;
246   2         end get$next$digit;
```

```
247   1       bcd:
                procedure (val) byte;
248   2           declare val byte;
249   2           return shl((val/10),4) + val mod 10;
250   2         end bcd;

251   1       declare (month, day, year, hrs, min, sec) byte;

252   1        set$date$time: procedure;
253   2           declare
                    (i, leap$flag) byte; /* temporaries */
254   2           month = scan$numeric(1,12) - 1;
                  /* may be feb 29 */
255   2           if (leap$flag := month = 1) then i = 29;
257   2               else i = month$size(month);
258   2           day   = scan$delimiter('/',1,i);
259   2           year  = scan$delimiter('/',base$year,99);
                  /* ensure that feb 29 is in a leap year */
260   2           if leap$flag and day = 29 and (year and 11b) <> 0 then
261   2               /* feb 29 of non-leap year */ call terminate;
                  /* compute total days */
262   2           tod.date = month$days(month)
                             + 365 * (year - base$year)
                             + day
                             - leap$days(base$year,0)
                             + leap$days(year,month);

263   2           tod.hrs   = bcd (scan$numeric(0,23));
264   2           tod.min   = bcd (scan$delimiter(':',0,59));
265   2           if tod.opcode = 2 then
                  /* date, hours and minutes only */
266   2           do;
267   3             if chr = ':'
                      then i = scan$delimiter (':',0,59);
269   3             tod.sec = 0;
270   3           end;
                  /* include seconds */
271   2           else tod.sec   = bcd (scan$delimiter(':',0,59));

272   2           end set$date$time;

273   1       bcd$pair: procedure(a,b) byte;
274   2           declare (a,b) byte;
275   2           return shl(a,4) or b;
276   2           end bcd$pair;


277   1       compute$year: procedure;
                  /* compute year from number of days in word$value */
278   2           declare year$length address;
279   2           year = base$year;
280   2               do forever;
281   3               year$length = 365;
282   3               if (year and 11b) = 0 then /* leap year */
283   3                   year$length = 366;
284   3               if word$value <= year$length then
```

```
285   3                 return;
286   3               word$value = word$value - year$length;
287   3               year = year + 1;
288   3               end;
289   2           end compute$year;

290   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 */

291   1       compute$month: procedure;
292   2           month = 12;
293   2               do while month > 0;
294   3               if (month := month - 1) < 2 then /* jan or feb */
295   3                   leap$bias = 0;
296   3               if month$days(month) + leap$bias < word$value then return;
298   3               end;
299   2           end compute$month;

300   1       declare
                  date$test byte,    /* true if testing date */
                  test$value address;   /* sequential date value under test */

301   1       get$date$time: procedure;
                  /* get date and time */
302   2           hrs = tod.hrs;
303   2           min = tod.min;
304   2           sec = tod.sec;
305   2           word$value = tod.date;
                  /* word$value contains total number of days */
306   2           week$day = (word$value + base$day - 1) mod 7;
307   2           call compute$year;
                  /* year has been set, word$value is remainder */
308   2           leap$bias = 0;
309   2           if (year and 11b) = 0 and word$value > 59 then
310   2               /* after feb 29 on leap year */ leap$bias = 1;
311   2           call compute$month;
312   2           day = word$value - (month$days(month) + leap$bias);
313   2           month = month + 1;
314   2           end get$date$time;

315   1       emit$date$time: procedure;
316   2           if tod.opcode = 0 then
317   2             do;
318   3             call emitn(.day$list(shl(week$day,2)));
319   3             call emitchar(' ');
320   3             end;
321   2           call emit$slant(month);
322   2           call emit$slant(day);
323   2           call emit$bin$pair(year);
324   2           call emitchar(' ');
325   2           call emit$colon(hrs);
326   2           call emit$colon(min);
327   2           if tod.opcode = 0 then
328   2             call emit$bcd$pair(sec);
```

```
329   2           end emit$date$time;

330   1       tod$ASCII:
                procedure (parameter);
331   2           declare parameter address;
332   2           declare ret address;

333   2           ret = 0;
334   2           tod$adr = parameter;
335   2           string$adr = .tod.ASCII;
336   2           if  (tod.opcode = 0) or
                  (tod.opcode = 3) then
337   2           do;
338   3             call get$date$time;
339   3             index = -1;
340   3             call emit$date$time;
341   3           end;
                  else
342   2           do;
343   3             if (tod.opcode = 1) or
                      (tod.opcode = 2) then
344   3             do;
345   4               chr = string(index:=0);
346   4               call set$date$time;
347   4               ret = .string(index);
348   4             end;
                    else
349   3             do;
350   4               call terminate;
351   4             end;
352   3           end;
353   2         end tod$ASCII;

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

                              TOD INTERFACE TO SHOW

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

354   1         declare lcltod structure (
                    opcode byte,
                    date address,
                    hrs byte,
                    min byte,
                    sec byte,
                    ASCII (21) byte );

355   1         declare datapgadr address;
356   1         declare datapg based datapgadr address;

357   1         declare extrnl$todadr address;
358   1         declare extrnl$tod based extrnl$todadr structure (
                    date address,
                    hrs byte,
```

```
                min byte,
                sec byte );

359   1       declare ret address;

              /*  display$tod:
                  procedure;
                    lcltod.opcode = 0;
                    call move (5,.extrnl$tod.date,.lcltod.date);
                    call tod$ASCII (.lcltod);
                    call write$console (0dh);
                    do i = 0 to 20;
                      call write$console (lcltod.ASCII(i));
                    end;
                  end display$tod; */

360   1       display$ts:
                  procedure (tsadr);
361   2           dcl i byte;
362   2           dcl tsadr address;
363   2           lcltod.opcode = 3;     /* display time and date stamp, no seconds */
364   2           call move (4,tsadr,.lcltod.date);  /* don't copy seconds */
365   2           call tod$ASCII (.lcltod);
366   2           do i = 0 to 13;
367   3             call printchar (lcltod.ASCII(i));
368   3           end;
369   2       end display$ts;

              /******** End TOD Code ********/

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

                                    * * *  BASIC ROUTINES * * *

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


370   1       declare
                  fcbmax literally '512'; /* max fcb count */

371   1       declare bpb address; /* bytes per block */

372   1       set$bpb: procedure;
373   2           call set$dpb; /* disk parameters set */
374   2           bpb = shl(double(1),blkshf) * sectorlen;
375   2           end set$bpb;

376   1       select$disk: procedure(d);
377   2           declare d byte;
                  /* select disk and set bpb */
378   2           call select(cdisk:=d);
379   2           call set$bpb; /* bytes per block */
```

```
380   2          end select$disk;

381   1      getalloc: procedure(i) byte;
                 /* return the ith bit of the alloc vector */
382   2          declare i address;
383   2          return
                 rol(alloc(shr(i,3)), (i and 111b) + 1);
384   2          end getalloc;

385   1      declare
                 accum(4) byte,     /* accumulator */
                 ibp byte;          /* input buffer pointer */

386   1      compare: procedure(a) byte;
                 /* compare accumulator with four bytes addressed by a */
387   2          declare a address;
388   2          declare (s based a) (4) byte;
389   2          declare i byte;
390   2              do i = 0 to 3;
391   3              if s(i) <> accum(i) then return false;
393   3              end;
394   2          return true;
395   2          end compare;

396   1      scan: procedure;
                 /* fill accum with next input value */
397   2          declare (i,b) byte;
398   2          setacc: procedure(b);
399   3              declare b byte;
400   3              accum(i) = b; i = i + 1;
402   3              end setacc;
                 /* deblank input */
403   2              do while buff(ibp) = ' '; ibp=ibp+1;
405   3              end;
                 /* initialize accum length */
406   2          i = 0;
407   2              do while i < 4;
408   3              if (b := buff(ibp)) > 1 then /* valid */
409   3                  call setacc(b); else /* blank fill */
410   3                  call setacc(' ');
411   3              if b <= 1 or b = ',' or b = ':' or
                       b = '*' or b = '.' or b = '>' or
                       b = '<' or b = '=' then buff(ibp) = 1;
                     else
413   3                  ibp = ibp + 1;
414   3              end;
415   2          ibp = ibp + 1;
416   2          end scan;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                              /* fill string @ s for c bytes with f */
417   1      fill:   proc(s,f,c);
418   2          dcl s addr,
                     (f,c) byte,
                     a based s byte;
```

```
419   2             do while (c:=c-1)<>255;
420   3             a = f;
421   3             s = s+1;
422   3             end;
423   2         end fill;


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

                              * * *  PRINT A NUMBER  * * *

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


424   1       declare
                  val (7) byte initial(0,0,0,0,0,0,0),    /* BCD digits     */
                  fac (7) byte initial(0,0,0,0,0,0,0),    /* hibyte factor */
                  f0  (7) byte initial(6,3,5,5,6,0,0),    /*     65,536     */
                  f1  (7) byte initial(2,7,0,1,3,1,0),    /*    131,072     */
                  f2  (7) byte initial(4,4,1,2,6,2,0),    /*    262,144     */
                  f3  (7) byte initial(8,8,2,4,2,5,0),    /*    524,288     */
                  f4  (7) byte initial(6,7,5,8,4,0,1),    /* 1,048,576     */
                  f5  (7) byte initial(2,5,1,7,9,0,2),    /* 2,097,152     */
                  f6  (7) byte initial(4,0,3,4,9,1,4),    /* 4,194,304     */
                  ptr (7) address initial(.f0,.f1,.f2,.f3,.f4,.f5,.f6);



                              /* print decimal value of address v */
425   1       pdecimal: procedure(v,prec,zerosup);
                  /* print value v with precision prec (1,10,100,1000,10000)
                  with leading zero suppression if zerosup = true */
426   2           declare
                      v address,     /* value to print */
                      prec address, /* precision */
                      zerosup byte, /* zero suppression flag */
                      d byte;       /* current decimal digit */
427   2               do while prec <> 0;
428   3               d = v / prec ; /* get next digit */
429   3               v = v mod prec;/* get remainder back to v */
430   3               prec = prec / 10; /* ready for next digit */
431   3               if prec <> 0 and zerosup and d = 0 then call printb;
                    else
433   3                   do;
434   4               zerosup = false;
435   4               call printchar('0'+d);
436   4                   end;
437   3               end;
438   2           end pdecimal;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                              /* BCD - convert 16 bit binary to
                                 7 one byte BCD digits */
439   1       getbcd: procedure(value);
```

```
440   2         declare
                    (value,prec) address,
                    i byte;

441   2         prec = 10000;
442   2         i = 5;                              /* digits: 4,3,2,1,0 */
443   2         do while prec <> 0;
444   3             val(i:=i-1) = value / prec;     /* get next digit */
445   3             value = value mod prec;         /* remainder in value */
446   3             prec = prec / 10;
447   3             end;
448   2         end getbcd;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                              /* print BCD number in val array */
449   1       printbcd: procedure;
450   2         declare
                    (zerosup, i) byte;

451   2         pchar: procedure(c);
452   3             declare c byte;
453   3             if val(i) = 0 then
454   3                 if zerosup then
455   3                     if i <> 0 then do;
457   4                         call printb;
458   4                         return;
459   4                         end;
                    /* else */
460   3             call printchar(c);
461   3             zerosup = false;
462   3         end pchar;

463   2         zerosup = true;
464   2         i = 7;
465   2         do while (i:=i-1) <> -1;
466   3             call pchar('0'+val(i));
467   3             if i = 6 or i = 3 then
468   3                 call pchar(',');
469   3             end;
470   2         end printbcd;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                              /* add two BCD numbers result in second */
471   1       add: procedure(ap,bp);
472   2           declare
                    (ap,bp)         address,
                    a based ap (7) byte,
                    b based bp (7) byte,
                    (c,i)           byte;

473   2           c = 0;                                /* carry   */
474   2               do i = 0 to 6;                    /* 0 = LSB */
475   3               b(i) = a(i) + b(i) + c;
476   3               c = b(i) / 10;
477   3               b(i) = b(i) mod 10;
```

```
478   3             end;
479   2         end add;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* print 3 byte value based at byte3adr */
480   1       p3byte: procedure(byte3adr);
481   2          declare
                     i          byte,
                     high$byte byte,
                     byte3adr  address,
                     b3 based  byte3adr structure (
                         lword address,
                         hbyte byte);

482   2          call fill(.val,0,7);
483   2          call fill(.fac,0,7);
484   2          call getbcd(b3.lword);          /* put 16 bit value in val */
485   2          high$byte = b3.hbyte;
486   2              do i = 0 to 6;                  /* factor for bit i */
487   3              if high$byte then               /* LSB is 1 */
488   3                  call add(ptr(i),.fac);      /* add in factor */
489   3                high$byte = shr(high$byte,1);  /* get next bit  */
490   3                end;
491   2          call add(.fac,.val);               /* add factor to value */
492   2          call printbcd;                     /* print value */
493   2          end p3byte;


              /* divide 3 byte value by 8 */
494   1       shr3byte: procedure(byte3adr);
495   2             dcl byte3adr address,
                      b3 based byte3adr structure (
                     lword address,
                      hbyte byte),
                     temp1 based byte3adr (2) byte,
                     temp2 byte;

496   2          temp2  = ror(b3.hbyte,3) and 11100000b;  /* get 3 bits */
497   2          b3.hbyte = shr(b3.hbyte,3);
498   2          b3.lword = shr(b3.lword,3);
499   2          temp1(1) = temp1(1) or temp2;   /* or in 3 bits from hbyte */
500   2          end shr3byte;


              /* multiply 3 byte value by #records per block */
501   1       shl3byte: procedure(byte3adr);
502   2             dcl byte3adr address,
                      b3 based byte3adr structure (
                     lword address,
                      hbyte byte),
                     temp1 based byte3adr (2) byte;

503   2          b3.hbyte = (rol(temp1(1),blkshf) and blkmsk) or shl(b3.hbyte,blkshf);
504   2          b3.lword = shl(b3.lword,blkshf);
505   2          end shl3byte;
```

```
506   1     show$drive: procedure;
507   2         call printchar(.diskf'A');
508   2         call printx(.(': ',0));
509   2         end show$drive;



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

                        * * *  CALCULATE SIZE  * * *

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


510   1     add$block: procedure(ak,ab);
511   2         declare (ak, ab) address;
                  /* add one block to the kilobyte accumulator */
512   2         declare kaccum based ak address; /* kilobyte accum */
513   2         declare baccum based ab address; /* byte accum */
514   2         baccum = baccum + bpb;
515   2             do while baccum >= 1024;
516   3             baccum = baccum - 1024;
517   3             kaccum = kaccum + 1;
518   3             end;
519   2         end add$block;

520   1     count: procedure(mode) address;
521   2         declare mode byte; /* true if counting 0's */
                  /* count kb remaining, kaccum set upon exit */
522   2         declare
                      ka  address,  /* kb accumulator */
                      ba  address,  /* byte accumulator */
                      i   address,  /* local index */
                      bit byte;     /* always 1 if mode = false */
523   2         ka, ba = 0;
524   2         bit = 0;
525   2             do i = 0 to maxall;
526   3             if mode then bit = getalloc(i);
528   3             if not bit then call add$block(.ka,.ba);
530   3             end;
531   2         return ka;
532   2         end count;



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

                        * * *  STATUS ROUTINES  * * *

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

```
                        /* characteristics of current drive */
533   1     drivestatus: procedure;
534   2           dcl b3a address,
                      b3 based b3a structure (
                         lword address,
                         hbyte byte);

                    /* print 3 byte value */
535   2         pv3: procedure;
536   3              call crlf;
537   3              call p3byte(.dirbuf);
538   3              call printchar(':');
539   3              call printb;
540   3              end pv3;

                    /* print address value v */
541   2          pv: procedure(v);
542   3              dcl v address;
543   3              b3.hbyte = 0;
544   3              b3.lword = v;
545   3              call pv3;
546   3              end pv;

              /* print the characteristics of the currently selected drive */
547   2         b3a = .dirbuf;
548   2         call print(.('        ',0));
549   2         call show$drive;
550   2         call printx(.('Drive Characteristics',0));
551   2         b3.hbyte = 0;
552   2         b3.lword = maxall + 1;        /* = # blocks */
553   2         call shl3byte(.dirbuf);         /* # blocks * records/block */
554   2         call pv3;
555   2         call printx(.('128 Byte Record Capacity',0));
556   2         call shr3byte(.dirbuf);         /* divide by 8 */
557   2         call pv3;
558   2         call printx(.('Kilobyte Drive  Capacity',0));
559   2         call pv(dirmax+1);
560   2         call printx(.('32 Byte  Directory Entries',0));
561   2         call pv(shl(chksiz,2));
562   2         call printx(.('Checked  Directory Entries',0));
563   2         call pv((extmsk+1) * 128);
564   2         call printx(.('Records / Directory Entry',0));
565   2         call pv(shl(double(1),blkshf));
566   2         call printx(.('Records / Block',0));
567   2         call pv(scptrk);
568   2         call printx(.('Sectors / Track',0));
569   2         call pv(offset);
570   2         call printx(.('Reserved  Tracks',0));
571   2         call crlf;
572   2         end drivestatus;
              /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* characteristics of all logged in disks */
```

```
573   1     diskstatus: procedure;
              /* display disk status */
574   2       declare login address, d byte;
575   2       login = getlogin; /* login vector set */
576   2       d = 0;
577   2           do while login <> 0;
578   3           if low(login) then
579   3               do; call select$disk(d);
581   4               call drivestatus;
582   4               end;
583   3           login = shr(login,1);
584   3           d = d + 1;
585   3           end;
586   2       end diskstatus;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                          /* help message */
587   1     help: procedure;
              /* display possible commands */

588   2           call print(.('Drive Status    : SHOW DRIVE:  SHOW d:DRIVE:',0));
589   2           call print(.('User Status     : SHOW USERS:  SHOW d:USERS:',0));
590   2           call print(.('Directory Label : SHOW LABEL:  SHOW d:LABEL:',0));
591   2           call print(.('Free Disk Space : SHOW SPACE:  SHOW d:SPACE:',0));
          /*
              call print(.('Locked Records  : LOCKED:',0));
              call print(.('Open Files      : OPEN:',0));
          */
592   2           call crlf;
593   2           end help;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                          /* parse error message */
594   1     parse$error: procedure;

595   2           call print(.version);
596   2           call crlf;
597   2           call print(.('Invalid Option, use the following:',0));
598   2           call crlf;
599   2           call help;
600   2           call terminate;
601   2           end parse$error;



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

                              * * *  DISK STATUS  * * *

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

```
602   1     pvalue: procedure(v);
603   2         declare (d,zero) byte,
                      (k,v) address;
604   2         k = 10000;
605   2         zero = false;
606   2             do while k <> 0;
607   3             d = low(v/k); v = v mod k;
609   3             k = k / 10;
610   3             if zero or k = 0 or d <> 0 then
611   3                  do; zero = true; call printchar('0'+d);
614   4                  end;
615   3             end;
616   2         end pvalue;


617   1     prcount: procedure;

                  /* print the actual byte count */
618   2         if cversion < cpm3 then do;
620   3             alloca = getalloca;
621   3         call pvalue(count(true));
622   3             end;
623   2         else do;
624   3         call setdma(.dirbuf);
625   3         call getfreesp(cdisk);
626   3         call shr3byte(.dirbuf);
627   3         call p3byte(.dirbuf);
628   3         end;
629   2         call printchar('k');
630   2         end prcount;

631   1     stat: procedure(rodisk);
632   2         declare rodisk address;

633   2             call crlf;
634   2             call show$drive;
635   2             call printchar('R');
636   2             if low(rodisk) then
637   2                 call printchar('0'); else
638   2                 call printchar('W');
639   2             call printx(.(', Space: ',0));
640   2             call prcount;
641   2         end stat;

642   1     prstatus: procedure;
                  /* print the status of the disk system */
643   2         declare (login, rodisk) address;
644   2         declare d byte;

645   2             login = getlogin; /* login vector set */
646   2         rodisk = getrodisk; /* read only disk vector set */
647   2         d = 0;
648   2             do while login <> 0;
649   3             if low(login) then
650   3                 do;
651   4             if fcb(0) <> 0 then do;
653   5             if fcb(0)-1 = d then
```

```
654   5                   call stat(rodisk);      /* to specific disk */
655   5               end;
656   4           else do;
657   5                   call selectdisk(d);
658   5               call stat(rodisk);      /* do all disks */
659   5               end;
660   4               end;
661   3           login = shr(login,1); rodisk = shr(rodisk,1);
663   3           d = d + 1;
664   3           end;
665   2       end prstatus;
```



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

                        * * *  USER STATUS * * *

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

666   1     get$usr$files: procedure;
667   2             declare ufcb(*) byte data ('?????????????',0,0,0),
                              (i,j)   byte,
                              nfcbs   address,
                              extptr  address,
                              modptr  address,
                              fmod    based modptr byte,
                              fext    based extptr byte;

668   2             do i = 0 to 15;
669   3                user(i),used(i) = 0;
670   3             end;
671   2             nSFCB = 0;

672   2             call setdma(.dirbuf);
673   2             call search(.ufcb);
674   2             nfcbs = 0;

675   2             do while dcnt <> 255;
676   3                      j = shl(dcnt,5);                /* which fcb in dirbuf */

677   3     ge0:             if (i := dirbuf(j)) <> 0e5h then do;
679   4                        if i <> 33 then do;          /* SFCB ? */
681   5                              extptr = .dirbuf(j + 12);
682   5                              modptr = extptr + 2;
683   5                              nfcbs = nfcbs + 1;
684   5                              j = i;                  /* Save for xfcb test */
685   5                              user(i := i and 0fh) = true;

686   5                              if j > 15 then go to ge2;
688   5                              if fext > extmsk then go to ge2;
690   5                              if fmod = 0 then used(i) = used(i) + 1;
692   5                        end;
```

```
 693   4                            else nSFCB = nSFCB + 1;
 694   4                          end;

 695   3       ge2:               call searchn;
 696   3                 end;

 697   2                 if nSFCB > 0 then nSFCB = shr(dirmax+1,2);/* because search ends
                                                                    at high water mark*/
 699   2                 free$dir = ((dirmax + 1) - nSFCB) - nfcbs;

 700   2       end get$usr$files;


 701   1       userstatus: procedure;
                   /* display active user numbers */
 702   2           declare i byte;
                   /*declare user(15) byte;
                   declare ufcb(*) byte data ('????????????',0,0,0);*/

 703   2           call set$bpb;
 704   2           call crlf;
 705   2           call show$drive;
 706   2           call printx(.('Active User :',0));
 707   2           call pdecimal(getuser,100,true);
 708   2           call crlf;
 709   2           call show$drive;
 710   2           call printx(.('Active Files:',0));
 711   2           call get$usr$files;
                       /*do i = 0 to last(user);
                       user(i) = false;
                       end;
                   call setdma(.dirbuf);
                   call search(.ufcb);
                       do while dcnt <> 255;
                       if (i := dirbuf(shl(dcnt and 11b,5))) <> 0e5h then
                           user(i and 0fh) = true;
                       call searchn;
                       end;*/
 712   2               do i = 0 to last(user);
 713   3               if user(i) then call pdecimal(i,100,true);
 715   3               end;

 716   2           call crlf;
 717   2           call show$drive;
 718   2           call printx(.('# of files  :',0));
 719   2           do i = 0 to last(user);
 720   3                 if user(i) then call pdecimal(used(i),100,true);
 722   3           end;

 723   2           end userstatus;


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

```
                    * * *  MP/M II DISK & FILE STATUS * * *

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


724   1      versionerr: procedure;

725   2          call print(.('Option not compatible with this O.S.',0));
726   2          call terminate;
727   2          end versionerr;

728   1      openfiles: procedure;

729   2          if cversion < cpm3 then
730   2          call versionerr;
731   2          call print(.('Not yet implemented',0));
732   2          end openfiles;

733   1      lockedstatus: procedure;

734   2          if cversion < cpm3 then
735   2          call versionerr;
736   2          call print(.('Not yet implemented',0));
737   2          end lockedstatus;

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

                L A B E L   S T A T U S

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

738   1      readlbl: proc;
739   2          dcl d byte data('?');

740   2          call setdma(.dirbuf);
741   2          call search(.d);
742   2          do while dcnt <> 0ffH;
743   3          if dirbuf(ror(dcnt,3) and 110$0000b)=20H then
744   3              return;
745   3          call searchn;
746   3          end;
747   2          end readlbl;

           /* HEADER */

748   1      dcl label1 (*) byte data (
           'Directory     Passwds  Stamp   Stamp',0);
749   1      dcl label2 (*) byte data (
           'Label         Reqd     ',0);
750   1      dcl label3 (*) byte data (
                                       '  Update  Label Created   Label Updated',0);
751   1      dcl label4 (*) byte data (
           '------------  -------  ------  ------  --------------  --------------',0);
```

```
752   1       labelstatus: procedure;
753   2           dcl (lbl, make)  byte;
754   2           dcl fnam lit '11';
755   2           dcl ftyp lit '9';
756   2           dcl fcbp address;
757   2           dcl fcbv based fcbp (32) byte;  /* template over dirbuf */

                  /* print file name */
758   2           printfn: proc;
759   3           declare k byte;

760   3           do k = 1 to fnam;
761   4           if k = ftyp then
762   4               call printchar('.');
763   4           call printchar(fcbv(k) and 7fh);
764   4           end;
765   3           end printfn;

766   2           if cversion < cpm3 then
767   2           call versionerr;
768   2           lbl = getlbl(cdisk);
769   2           if lbl > 0 then do;
771   3           call readlbl;
772   3           fcbp = shl(dcnt,5) + .dirbuf;

                  /* print heading */
773   3           call print(.('Label for drive ',0));
774   3           call show$drive;
775   3           call crlf;
776   3           call print(.label1);
777   3           call print(.label2);
778   3           if (lbl and 40h) = 40h then
779   3               call printx(.('Access',0));
                  else
780   3               call printx(.('Create',0));
781   3           call printx(.label3);
782   3           call print(.label4);
783   3           call crlf;
784   3           call printfn;
785   3           if (lbl and 80h) = 80h then
786   3               call printx(.('    on   ',0));
                  else
787   3               call printx(.('    off  ',0));

                  /* if (make:=(lbl and 10h) = 10h) then
                         call printx(.('   on  ',0));        (Removed with Make XFCB option)
                  else
                      call printx(.('   off ',0));*/

788   3           if ((lbl and 40h) = 40h) or make then
789   3               call printx(.('   on   ',0));
                  else
790   3               call printx(.('   off  ',0));
791   3           if (lbl and 20h) = 20h then
792   3               call printx(.('   on ',0));
                  else
793   3               call printx(.('   off',0));
```

```
794   3          call printx(.('    ',0));
795   3          call display$ts(.fcbv(24));
796   3          call printx(.('  ',0));
797   3          call display$ts(.fcbv(28));
798   3          end;
              else
799   2          call print(.('No Directory Label exists',0));
800   2          call crlf;
801   2          end labelstatus;

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

                              * * *  PARSING  * * *

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


802   1    parse$next: procedure;

              /* skip comma or space delimiter */
803   2          parse$fn.buff$adr = parse$fn.buff$adr + 1;
804   2          parse$fn.buff$adr = parse;
805   2          if parse$fn.buff$adr = 0ffffh then
806   2              call parse$error;
807   2          if delimiter = '[' or delimiter = ':' then      /* skip */
808   2              parse$fn.buff$adr = parse$fn.buff$adr + 1;
809   2          if delimiter = 0 then
810   2              parse$fn.buff$adr = 0;
811   2          end parse$next;


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

                              * * *  MAIN PROGRAM  * * *

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

812   1    declare
              i              byte initial(1),
              last$dseg$byte byte initial (0);


813   1    PLMSTART: PROCEDURE PUBLIC;

              /* process request */
814   2    cversion = get$version;
815   2    ibp=1;
816   2    if cversion < cpmversion then
817   2        call printx(.('Requires CP/M 2.0',0));
              else
```

```
818   2             do;
                        /* scan for global option */
819   3                 do while buff(i)=' ';
820   4                 i = i + 1;
821   4                 end;
822   3             if buff(i) = '[' then     /* skip leading [ */
823   3                 parse$fn.buff$adr = .buff(i);
                      else
824   3                 parse$fn.buff$adr = .buff;
825   3             parse$fn.fcb$adr = .fcb;
826   3         cdisk = cselect;
827   3             user$code = getuser;
828   3                 do while parse$fn.buff$adr <> 0;
829   4                 call parse$next;
830   4                 if fcb(0) <> 0 then          /* get drive  */
831   4                     call select$disk(fcb(0)-1);
832   4                 if delimiter = '[' then
833   4                     call parse$next;         /* get option */
834   4                 if fcb(1) = ' ' or fcb(1) = 'S' then
835   4                     call prstatus;
836   4                 else if fcb(1) = 'U' then
837   4                     call userstatus;
838   4                 else if fcb(1) = 'H' then
839   4                     call help;
840   4                 else if fcb(1) = 'D' then
841   4                     do;
842   5                     if fcb(0)<>0 then
843   5                         call drivestatus;
                          else
844   5                         call diskstatus;
845   5                     end;
846   4                 else if fcb(1) = 'O' then
847   4                     call openfiles;
848   4                 else if fcb(1) = 'L' then do;
850   5                     if fcb(2) = 'A' then
851   5                         call labelstatus;
852   5                     else if fcb(2) = 'O' then
853   5                         call lockedstatus;
                          else
854   5                         call parse$error;
855   5                     end;
                      else
856   4                     call parse$error;
857   4                 end;
858   3             end;
859   2         call terminate;
860   2     END PLMSTART;
861   1     end show;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES								
418	0000H	1	A.	BYTE BASED(S)	420						
273	0006H	1	A.	BYTE PARAMETER AUTOMATIC	274	275					
386	0004H	2	A.	WORD PARAMETER AUTOMATIC	387	388	391				
28	0004H	2	A.	WORD PARAMETER AUTOMATIC	29	30	31	32	33		
10	0000H	2	A.	WORD PARAMETER	11						
472	0000H	7	A.	BYTE BASED(AP) ARRAY(7)	475						
13	0000H	2	A.	WORD PARAMETER	14						
7	0000H	2	A.	WORD PARAMETER	8						
150	0004H	2	A.	WORD PARAMETER AUTOMATIC	151	152	153	154	155		
48	0004H	2	A.	WORD PARAMETER AUTOMATIC	49	51					
16	0000H	2	A.	WORD PARAMETER	17						
510	0004H	2	AB	WORD PARAMETER AUTOMATIC	511	513	514	515	516		
385	012BH	4	ACCUM.	BYTE ARRAY(4)	391	400					
471	098DH	75	ADD.	PROCEDURE STACK=0006H	488	491					
510	0AA8H	35	ADDBLOCK	PROCEDURE STACK=0006H	529						
20			ADDR	LITERALLY	418						
510	0006H	2	AK	WORD PARAMETER AUTOMATIC	511	512	517				
19	0000H	1024	ALLOC.	BYTE BASED(ALLOCA) ARRAY(1024)		383					
19	0004H	2	ALLOCA	WORD	19	383	620				
65	006EH	1	ANYTHING	BYTE	70	82					
471	0006H	2	AP	WORD PARAMETER AUTOMATIC	472	475					
354	0006H	21	ASCII.	BYTE ARRAY(21) MEMBER(LCLTOD)		367					
142	0006H	21	ASCII.	BYTE ARRAY(21) MEMBER(TOD)		335					
472	0000H	7	B.	BYTE BASED(BP) ARRAY(7)	475	476	477				
273	0004H	1	B.	BYTE PARAMETER AUTOMATIC	274	275					
397	0132H	1	B.	BYTE	408	409	411				
167	0004H	1	B.	BYTE PARAMETER AUTOMATIC	168	169					
398	0004H	1	B.	BYTE PARAMETER AUTOMATIC	399	400					
177	0004H	1	B.	BYTE PARAMETER AUTOMATIC	178	179					
158	0004H	1	B.	BYTE PARAMETER AUTOMATIC	159	160					
172	0004H	1	B.	BYTE PARAMETER AUTOMATIC	173	174	175				
203	0101H	1	B.	BYTE	204	209	211	214	219	221	
162	0004H	1	B.	BYTE PARAMETER AUTOMATIC	163	164	165				
495	0000H	3	B3	STRUCTURE BASED(BYTE3ADR)	496	497	498				
481	0000H	3	B3	STRUCTURE BASED(BYTE3ADR)	484	485					
502	0000H	3	B3	STRUCTURE BASED(BYTE3ADR)	503	504					
534	0000H	3	B3	STRUCTURE BASED(B3A)	543	544	551	552			
534	0058H	2	B3A.	WORD	534	543	544	547	551	552	
522	0054H	2	BA	WORD	523	529					
513	0000H	2	BACCUM	WORD BASED(AB)	514	515	516				
231			BASEDAY.	LITERALLY	306						
231			BASEYEAR	LITERALLY	259	262	279				
247	0417H	39	BCD.	PROCEDURE BYTE STACK=0006H		263	264	271			
273	055FH	17	BCDPAIR.	PROCEDURE BYTE STACK=0006H							
522	0178H	1	BIT.	BYTE	524	527	528				
6			BLKMSK	LITERALLY	503						
6			BLKSHF	LITERALLY	374	503	504	565			
6	0002H	1	BLS.	BYTE MEMBER(DPB)		374	503	504	565		
6	0003H	1	BMS.	BYTE MEMBER(DPB)		503					

```
471  0004H     2  BP . . . . . .  WORD PARAMETER AUTOMATIC          472  475  476  477
371  0040H     2  BPB. . . . . .  WORD          374  514
 36  0039H    15  BREAK. . . . .  PROCEDURE BYTE STACK=0008H        42
  6  0000H   128  BUFF . . . . .  BYTE ARRAY(128) EXTERNAL(2)      403  408  412  813  822  823  824
  6             BUFFA. . . . . .  LITERALLY
134  0000H     2  BUFFADR. . . . .  WORD MEMBER(PARSEFN)      134  303  804  805  807  808  809  810  823
                                    824  828  832
501  0004H     2  BYTE3ADR . . . . .  WORD PARAMETER AUTOMATIC      502  503  504
494  0004H     2  BYTE3ADR . . . . .  WORD PARAMETER AUTOMATIC      495  496  497  498  499
480  0004H     2  BYTE3ADR . . . . .  WORD PARAMETER AUTOMATIC      481  484  485
417  0004H     1  C. . . . . .  BYTE PARAMETER AUTOMATIC            418  419
146  0004H     1  C. . . . . .  BYTE PARAMETER AUTOMATIC            147  148
472  0176H     1  C. . . . . .  BYTE          473  475  476
451  0004H     1  C. . . . . .  BYTE PARAMETER AUTOMATIC            452  460
152  0000H     1  C. . . . . .  BYTE BASED(A)          153  154
                CARRY. . . . . .  BUILTIN          212  215
  6  006CH     1  CDISK. . . . . .  BYTE          378  507  625  768  826
 21  0004H     1  CHAR . . . . . .  BYTE PARAMETER AUTOMATIC         22   23
 68  0088H    63  CHECKUSER. . . . .  PROCEDURE STACK=0008H         84   88
  6             CHKSIZ . . . . . .  LITERALLY        561
182  0100H     1  CHR. . . . . .  BYTE          184  188  191  194  199  214  226  267  345
  6  0008H     2  CKS. . . . . .  WORD MEMBER(DPB)          561
386  07A2H    46  COMPARE. . . . . .  PROCEDURE BYTE STACK=0004H
291  05A5H    59  COMPUTEMONTH . . .  PROCEDURE STACK=0002H         311
277  0570H    53  COMPUTEYEAR. . . .  PROCEDURE STACK=0002H         307
  3  0018H    38  COPYRIGHT. . . . .  BYTE ARRAY(38) DATA
520  0AC8H    82  COUNT. . . . . .  PROCEDURE WORD STACK=000CH        621
  2             CPM3 . . . . . .  LITERALLY        618  729  734  766
  2             CPMVERSION . . . .  LITERALLY        816
 20             CR . . . . . .  LITERALLY         40
 39  0048H    43  CRLF . . . . . .  PROCEDURE STACK=000EH          50  536  571  592  595  598  633  704
                                    708  716  775  783  800
 90  012EH    15  CSELECT. . . . . .  PROCEDURE BYTE STACK=0008H        826
  6  006BH     1  CVERSION . . . . .  BYTE          618  729  734  766  814  816
130  0004H     1  D. . . . . .  BYTE PARAMETER AUTOMATIC            131  132
 57  0004H     1  D. . . . . .  BYTE PARAMETER AUTOMATIC             58   59
223  0008H     1  D. . . . . .  BYTE PARAMETER AUTOMATIC            224  226
644  017FH     1  D. . . . . .  BYTE          647  653  657  663
574  017CH     1  D. . . . . .  BYTE          576  580  584
603  017DH     1  D. . . . . .  BYTE          607  610  613
739  0086H     1  D. . . . . .  BYTE DATA          741
126  0004H     1  D. . . . . .  BYTE PARAMETER AUTOMATIC            127  128
376  0004H     1  D. . . . . .  BYTE PARAMETER AUTOMATIC            377  378
426  0172H     1  D. . . . . .  BYTE          428  431  435
356  0000H     2  DATAPG . . . . . .  WORD BASED(DATAPGADR)
355  003AH     2  DATAPGADR. . . . .  WORD          356
142  0001H     2  DATE . . . . . .  WORD MEMBER(TOD)          262  305
358  0000H     2  DATE . . . . . .  WORD MEMBER(EXTRNLTOD))
354  0001H     2  DATE . . . . . .  WORD MEMBER(LCLTOD))       364
300  010EH     1  DATETEST . . . . .  BYTE
251  0105H     1  DAY. . . . . .  BYTE          258  260  262  312  322
290  0058H    28  DAYLIST. . . . . .  BYTE ARRAY(28) DATA       318
  6  0009H     2  DBL. . . . . .  WORD MEMBER(DPB)
 20             DCL. . . . . .  LITERALLY
 53  006DH     1  DCNT . . . . . .  BYTE           63   72   74   76   83   87  675  676  742  743  772
193  0311H    17  DEBLANK. . . . . .  PROCEDURE STACK=0006H        205  225
134  0000H     1  DELIMITER. . . . .  BYTE BASED(PARSEFN.BUFFADR)      807  809  832
```

```
  6                 DIRBLK . . . . . . .    LITERALLY
 66  006FH   128    DIRBUF . . . . . . .    BYTE ARRAY(128)       74   537   547   553   556   624   625   627   672   677
                                             631  740  743  772
  6                 DIRMAX . . . . . . .    LITERALLY         559  698  699
573  0C1FH    50    DISKSTATUS . . . .      PROCEDURE STACK=0030H              844
360  0713H    64    DISPLAYTS. . . . .      PROCEDURE STACK=002AH              795   797
 93  0004H     2    DMA. . . . . . . .      WORD PARAMETER AUTOMATIC            94    95
  6  0007H     2    DMX. . . . . . . .      WORD MEMBER(DPB)             559   698   699
  6  0011H     1    DOLL . . . . . . .      BYTE EXTERNAL(1) AT
  6                 DOLLA. . . . . . .      LITERALLY           6
                    DOUBLE . . . . . .      BUILTIN           374  565
  6  0000H    15    DPB. . . . . . . .      STRUCTURE BASED(DPBPTR)            374   503   504   525   552   559   561   563
                                             565  567  569  688  698  699
  6  0000H     4    DPBPTR . . . . . .      POINTER             6  115   374   503   504   525   552   559   561   563   565
                                             567  569  688  598  699
533  0810H   211    DRIVESTATUS. . . .      PROCEDURE STACK=002CH              581   843
158  0268H    16    EMITBCD. . . . . .      PROCEDURE STACK=000AH              164   165   174   175
162  0278H    27    EMITBCDPAIR. . . .      PROCEDURE STACK=0010H              169   328
172  02A6H    39    EMITBINPAIR. . . .      PROCEDURE STACK=0010H              179   323
146  0224H    28    EMITCHAR . . . . .      PROCEDURE STACK=0004H              160   170   180   319   324
167  0293H    19    EMITCOLON. . . . .      PROCEDURE STACK=0016H              325   326
315  064AH    95    EMITDATETIME . . .      PROCEDURE STACK=001AH              340
150  0240H    40    EMITN. . . . . . .      PROCEDURE STACK=0004H              318
177  02CDH    19    EMITSLANT. . . . .      PROCEDURE STACK=0016H              321   322
  6  0004H     1    EXM. . . . . . . .      BYTE MEMBER(DPB)             563   688
  6                 EXTMSK . . . . . .      LITERALLY         563  688
667  0064H     2    EXTPTR . . . . . .      WORD          667  681  682  688
358  0000H     5    EXTRNLTOD. . . . .      STRUCTURE BASED(EXTRNLTODADR)
357  003CH     2    EXTRNLTODADR . . .      WORD          358
 16  0000H     1    F. . . . . . . . .      BYTE PARAMETER         17
 10  0000H     1    F. . . . . . . . .      BYTE PARAMETER         11
  7  0000H     1    F. . . . . . . . .      BYTE PARAMETER          8
417  0006H     1    F. . . . . . . . .      BYTE PARAMETER AUTOMATIC           418   420
 13  0000H     1    F. . . . . . . . .      BYTE PARAMETER         14
424  0141H     7    F0 . . . . . . . .      BYTE ARRAY(7) INITIAL              424
424  0148H     7    F1 . . . . . . . .      BYTE ARRAY(7) INITIAL              424
424  014FH     7    F2 . . . . . . . .      BYTE ARRAY(7) INITIAL              424
424  0156H     7    F3 . . . . . . . .      BYTE ARRAY(7) INITIAL              424
424  015DH     7    F4 . . . . . . . .      BYTE ARRAY(7) INITIAL              424
424  0164H     7    F5 . . . . . . . .      BYTE ARRAY(7) INITIAL              424
424  016BH     7    F6 . . . . . . . .      BYTE ARRAY(7) INITIAL              424
424  013AH     7    FAC. . . . . . . .      BYTE ARRAY(7) INITIAL              483   488   491
 20                 FALSE. . . . . . .      LITERALLY         392  434   461   605
 61  0004H     2    FCB. . . . . . . .      WORD PARAMETER AUTOMATIC            62    63
122  0004H     2    FCB. . . . . . . .      WORD PARAMETER AUTOMATIC           123   124
 79  0004H     2    FCB. . . . . . . .      WORD PARAMETER AUTOMATIC            80    81    82    83
  6  0000H    33    FCB. . . . . . . .      BYTE ARRAY(33) EXTERNAL(1)               6   110   651   653   825   830   831
                                             834  836  838  840  842   846   848   850   852
 81  0000H     1    FCB0 . . . . . . .      BYTE BASED(FCB)        62
  6                 FCBA . . . . . . .      LITERALLY         110
134  0002H     2    FCBADR . . . . . .      WORD MEMBER(PARSEFN)         825
370                 FCBMAX . . . . . .      LITERALLY
756  0068H     2    FCBP . . . . . . .      WORD          757  763  772   795   797
757  0000H    32    FCBV . . . . . . .      BYTE BASED(FCBP) ARRAY(32)               763   795   797
667  0000H     1    FEXT . . . . . . .      BYTE BASED(EXTPTR)           688
417  085DH    32    FILL . . . . . . .      PROCEDURE STACK=0008H              482   483
667  0000H     1    FMOD . . . . . . .      BYTE BASED(MODPTR)           690
```

```
754                 FNAM . . . . . LITERALLY        760
 20                 FOREVER. . . . . LITERALLY        69  280
 67  0028H      2   FREEDIR. . . . . WORD       699
755                 FTYP . . . . . LITERALLY        761
677  0E04H          GE0. . . . . LABEL
695  0E6FH          GE2. . . . . LABEL       687  689
381  0784H     30   GETALLOC . . . . . PROCEDURE BYTE STACK=0004H                527
 97  014DH     15   GETALLOCA. . . . . PROCEDURE WORD STACK=0008H                620
439  0803H     76   GETRCD . . . . . PROCEDURE STACK=0004H           484
301  05E0H    106   GETDATETIME. . . . PROCEDURE STACK=0006H           338
122  01D1H     16   GETFILESIZE. . . . PROCEDURE STACK=000AH
126  01E1H     19   GETFREESP. . . . . PROCEDURE STACK=000AH           625
130  01F4H     19   GETLBL . . . . . PROCEDURE BYTE STACK=000AH                758
100  015CH     15   GETLOGIN . . . . . PROCEDURE WORD STACK=0008H                575  645
241  03F7H     32   GETNEXTDIGIT . . . PROCEDURE BYTE STACK=0002H
106  017AH     15   GETRODISK. . . . . PROCEDURE WORD STACK=0008H                646
115  018FH     15   GETUSER. . . . . PROCEDURE BYTE STACK=0008H                707  827
666  0D83H    239   GETUSRFILES. . . . PROCEDURE STACK=0012H           711
 54  0083H     15   GETVERSION . . . . PROCEDURE BYTE STACK=0008H                814
183  02E0H     49   GNC. . . . . PROCEDURE STACK=0002H           195  217  228
534  0002H      1   HBYTE. . . . . BYTE MEMBER(83)       543  551
502  0002H      1   HBYTE. . . . . BYTE MEMBER(83)       503
495  0002H      1   HBYTE. . . . . BYTE MEMBER(83)       496  497
481  0002H      1   HBYTE. . . . . BYTE MEMBER(83)       485
587  0C51H     36   HELP . . . . . PROCEDURE STACK=001AH           599  839
481  0179H      1   HIGHBYTE . . . . . BYTE        485  487  489
251  0107H      1   HRS. . . . . BYTE        302  325
142  0003H      1   HRS. . . . . BYTE MEMBER(TOD)            263  302
358  0002H      1   HRS. . . . . BYTE MEMBER(EXTRNLTOD)
354  0003H      1   HRS. . . . . BYTE MEMBER(LCLTOD)
812  0186H      1   I. . . . . BYTE INITIAL          819  820  822  823
253  010AH      1   I. . . . . BYTE        256  257  258  268
702  0182H      1   I. . . . . BYTE        712  713  714  719  720  721
667  0180H      1   I. . . . . BYTE        668  669  677  679  684  685  691
522  0056H      2   I. . . . . WORD        525  527
481  0178H      1   I. . . . . BYTE        486  488
472  0177H      1   I. . . . . BYTE        474  475  476  477
450  0175H      1   I. . . . . BYTE        453  455  464  465  466  467
440  0173H      1   I. . . . . BYTE        442  444
397  0131H      1   I. . . . . BYTE        400  401  406  407
389  0130H      1   I. . . . . BYTE        390  391
381  0004H      2   I. . . . . WORD PARAMETER AUTOMATIC          382  383
361  012AH      1   I. . . . . BYTE        366  367
385  012FH      1   IBP. . . . . BYTE        403  404  408  412  413  415  815
145  00FFH      1   INDEX. . . . . BYTE        148  154  186  191  339  345  347
667  0181H      1   J. . . . . BYTE        676  677  681  684  686
759  0185H      1   K. . . . . BYTE        760  761  763
603  005CH      2   K. . . . . WORD        604  606  607  608  609  610
522  0052H      2   KA . . . . . WORD        523  529  531
512  0000H      2   KACCUM . . . . . WORD BASED(AK)        517
748  0087H     37   LABEL1 . . . . . BYTE ARRAY(37) DATA        776
749  00ACH     24   LABEL2 . . . . . BYTE ARRAY(24) DATA        777
750  00C4H     40   LABEL3 . . . . . BYTE ARRAY(40) DATA        781
751  00ECH     70   LABEL4 . . . . . BYTE ARRAY(70) DATA        782
752  0FABH    239   LABELSTATUS. . . . PROCEDURE STACK=002EH           851
                    LAST . . . . . BUILTIN          712  719
812  0187H      1   LASTDSEGBYTE . . . BYTE INITIAL
```

```
  223  0006H     1  LB . . . . . .   BYTE PARAMETER AUTOMATIC          224  229
  201  0006H     1  LB . . . . . .   BYTE PARAMETER AUTOMATIC          202  219
  753  0183H     1  LBL. . . . . .   BYTE          768  769  778  785  788  791
  354  010FH    27  LCLTOD . . . . .  STRUCTURE     363  364  365  367
  290  010DH     1  LEAPBIAS . . . . .  BYTE        295  296  308  310  312
  232  03CBH    44  LEAPDAYS . . . . .  PROCEDURE BYTE STACK=0006H          262
  253  010BH     1  LEAPFLAG . . . . .  BYTE        255  260
   20             LF . . . . . .   LITERALLY        41
   20             LIT. . . . . .   LITERALLY       231  754  755
  733  0F60H    22  LOCKEDSTATUS . . .  PROCEDURE STACK=001EH          853
  574  005AH     2  LOGIN. . . . . .  WORD          575  577  578  583
  643  005EH     2  LOGIN. . . . . .  WORD          645  648  649  661
                  LOW. . . . . .   BUILTIN         578  607  636  649
  242  0103H     1  LSD. . . . . .   BYTE          243  245
  534  0000H     2  LWORD. . . . . .  WORD MEMBER(83)        544  552
  502  0000H     2  LWORD. . . . . .  WORD MEMBER(83)        504
  495  0000H     2  LWORD. . . . . .  WORD MEMBER(83)        498
  481  0000H     2  LWORD. . . . . .  WORD MEMBER(83)        484
  232  0004H     1  M. . . . . .   BYTE PARAMETER AUTOMATIC          233  236
  753  0184H     1  MAKE . . . . . .  BYTE          788
    6             MAXALL . . . . . .  LITERALLY     525  552
    6  0000H     2  MAXB . . . . . .  WORD EXTERNAL(0)
    6             MEMSIZE. . . . . .  LITERALLY
  251  0108H     1  MIN. . . . . .   BYTE          303  326
  142  0004H     1  MIN. . . . . .   BYTE MEMBER(TOD)          264  303
  358  0003H     1  MIN. . . . . .   BYTE MEMBER(EXTRNLTOD)
  354  0004H     1  MIN. . . . . .   BYTE MEMBER(LCLTOD)
  520  0004H     1  MODE . . . . . .  BYTE PARAMETER AUTOMATIC          521  526
  667  0066H     2  MODPTR . . . . . .  WORD        667  682  690
    7  0000H        MON1 . . . . . .  PROCEDURE EXTERNAL(3) STACK=0000H          23   44   45   59   95  104
                    110  120  124  128  139
   10  0000H        MON2 . . . . . .  PROCEDURE BYTE EXTERNAL(4) STACK=0000H          37   55   63   76   83
                     87   91  116  132
   13  0000H        MON3 . . . . . .  PROCEDURE WORD EXTERNAL(5) STACK=0000H          98  101  107  136
   16  0000H        MON4 . . . . . .  PROCEDURE POINTER EXTERNAL(6) STACK=0000H          113
  251  0104H     1  MONTH. . . . . .   BYTE          254  255  257  262  292  293  294  296  312  313  321
  231  0000H    24  MONTHDAYS. . . . .  WORD ARRAY(12) DATA        236  262  296  312
  231  004FH    12  MONTHSIZE. . . . .  BYTE ARRAY(12) DATA        257
                  MOVE . . . . . .  BUILTIN          364
    6  0005H     2  MXA. . . . . .   WORD MEMBER(DP3)          525  552
  667  0062H     2  NFCBS. . . . . .  WORD          674  683  699
   67  0026H     2  NSFCB. . . . . .  WORD          671  693  697  698  699
  198  0322H    17  NUMERIC. . . . . .  PROCEDURE BYTE STACK=0002H          206  208
    6             OFFSET . . . . . .  LITERALLY     569
    6  000DH     2  OFS. . . . . .   WORD MEMBER(DP3)          569
  142  0000H     1  OPCODE . . . . . .  BYTE MEMBER(TOD)          265  316  327  336  343
  354  0000H     1  OPCODE . . . . . .  BYTE MEMBER(LCLTOD)        363
   61  00A5H    19  OPEN . . . . . .  PROCEDURE STACK=000AH
  728  0F4AH    22  OPENFILES. . . . .  PROCEDURE STACK=001EH          847
  480  09D8H   112  P3BYTE . . . . . .  PROCEDURE STACK=001EH          537  627
  330  0004H     2  PARAMETER. . . . .  WORD PARAMETER AUTOMATIC      331  334
    6  0012H     1  PARM . . . . . .  BYTE EXTERNAL(1) AT
    6             PARMA. . . . . .  LITERALLY        6
  135  0207H    15  PARSE. . . . . .  PROCEDURE WORD STACK=0008H          804
  594  0C75H    31  PARSEERROR . . . .  PROCEDURE STACK=001EH          806  854  856
  134  002AH     4  PARSEFN. . . . . .  STRUCTURE     134  136  803  804  805  807  808  809  810  823  824
                    825  828  832
```

```
802  1001H   58  PARSENEXT. . . . .  PROCEDURE STACK=0022H                829  833
451  095EH   47  PCHAR. . . . . .  PROCEDURE STACK=0014H                466  468
425  0870H   86  POLCIMAL . . . . .  PROCEDURE STACK=0018H                707  714  721
813  1106H  251  PLMSTART . . . . .  PROCEDURE PUBLIC STACK=0034H
617  0CF1H   64  PRCOUNT. . . . . .  PROCEDURE STACK=0022H                640
440  0050H    2  PREC . . . . . .  WORD          441  443  444  445      446
425  0006H    2  PREC . . . . . .  WORD PARAMETER AUTOMATIC              426  427  428  429  430  431
 48  0073H   16  PRINT. . . . . .  PROCEDURE STACK=0016H                548  588  589  590  591  595  597  725
                                    731  736  773  776  777  782        799
 25  0015H   11  PRINTB . . . . . .  PROCEDURE STACK=0006H              432  457  535
449  091FH   63  PRINTBCD . . . . .  PROCEDURE STACK=0018H              492
 21  0002H   19  PRINTCHAR. . . . .  PROCEDURE STACK=000AH               26   32   40   41  367  435  460  507
                                    538  613  629  635  637  638       762  763
758  109AH   55  PRINTFN. . . . . .  PROCEDURE STACK=000EH              784
 28  0020H   25  PRINTX . . . . . .  PROCEDURE STACK=0010H               51  508  550  555  558  560  562  564
                                    566  568  570  630  706  710       718  779  780  781  786  787  789  790
                                    792  793  794  796  817
 20          PROC . . . . . .  LITERALLY        417  738  758
642  0D5FH   84  PRSTATUS . . . . .  PROCEDURE STACK=002CH              835
424  0042H   14  PTR. . . . . .  WORD ARRAY(7) INITIAL                  488
541  0C08H   23  PV . . . . . .  PROCEDURE STACK=0028H                559  561  563  565  567  569
535  08F0H   24  PV3. . . . . .  PROCEDURE STACK=0022H                545  554  557
602  0C94H   93  PVALUE . . . . . .  PROCEDURE STACK=0010H             621
738  0F76H   53  READLBL. . . . . .  PROCEDURE STACK=0012H             771
332  0038H    2  RET. . . . . .  WORD          333  347
359  003EH    2  RET. . . . . .  WORD
643  0060H    2  RODISK . . . . . .  WORD          646  654  658  662
631  0004H    2  RODISK . . . . . .  WORD PARAMETER AUTOMATIC           632  636
          ROL. . . . . .  BUILTIN          383  503
          ROR. . . . . .  BUILTIN           74  496  743
  6  0023H    1  ROVF . . . . . .  BYTE EXTERNAL(1) AT
  6  0021H    2  RREC . . . . . .  WORD EXTERNAL(1) AT
  6          RRECA. . . . . .  LITERALLY         6
  6          RRECO. . . . . .  LITERALLY         6
417  0008H    2  S. . . . . .  WORD PARAMETER AUTOMATIC                418  420  421
388  0000H    4  S. . . . . .  BYTE BASED(A) ARRAY(4)                  391
 30  0000H    1  S. . . . . .  BYTE BASED(A)          31   32
396  07D0H  117  SCAN . . . . . .  PROCEDURE STACK=0008H
223  03AAH   33  SCANDELIMITER. . .  PROCEDURE BYTE STACK=001AH          258  259  264  268  271
201  0333H  119  SCANNUMERIC. . . .  PROCEDURE BYTE STACK=0010H          229  254  263
  6          SCPTRK . . . . . .  LITERALLY        567
 79  00F7H   34  SEARCH . . . . . .  PROCEDURE STACK=000EH              673  741
 86  0119H   21  SEARCHN. . . . . .  PROCEDURE STACK=000CH              695  745
251  0109H    1  SEC. . . . . .  BYTE          304  328
142  0005H    1  SEC. . . . . .  BYTE MEMBER(TOD)          269  271  304
358  0004H    1  SEC. . . . . .  BYTE MEMBER(EXTRNLTOD)
354  0005H    1  SEC. . . . . .  BYTE MEMBER(LCLTOD)
  6          SECTORLEN. . . . .  LITERALLY        374
 57  0092H   19  SELECT . . . . . .  PROCEDURE STACK=000AH              378
376  0770H   20  SELECTDISK . . . .  PROCEDURE STACK=0012H              580  657  831
398  0845H   24  SETACC . . . . . .  PROCEDURE STACK=0004H              409  410
372  0753H   29  SETBPB . . . . . .  PROCEDURE STACK=000CH              379  703
252  043EH  289  SETDATETIME. . . .  PROCEDURE STACK=001EH              346
 93  013DH   16  SETDMA . . . . . .  PROCEDURE STACK=000AH              624  672  740
112  0198H   23  SETDPB . . . . . .  PROCEDURE STACK=0008H              373
109  0189H   15  SETIND . . . . . .  PROCEDURE STACK=0008H
118  018EH   19  SETUSER. . . . . .  PROCEDURE STACK=000AH
```

```
                    SHL. . . . . BUILTIN          211  249  275  318  374  503  504  561  565  676  772
  501  0A66H    45   SHL3BYTE . . . . . PROCEDURE STACK=0006H          503
    1  0002H        SHOW . . . . . PROCEDURE STACK=0000H
  506  0A93H    21   SHOWDRIVE. . . . . PROCEDURE STACK=0014H          549  634  705  709  717  774
                    SHR. . . . . BUILTIN          164  235  383  489  497  498  583  651  662  698
  494  0A48H    30   SHR3BYTE . . . . . PROCEDURE STACK=0004H          556  626
    6  0000H     2   SPT. . . . . WORD MEMBER(DPB)          567
  631  0D31H    46   STAT . . . . . PROCEDURE STACK=0028H          654  658
  144  0000H     1   STRING . . . . . BYTE BASED(STRINGADR) ARRAY(1)          148  154  191  345  347
  143  0000H     2   STRINGADR. . . . . WORD          144  148  154  191  345  345  347
  502  0000H     2   TEMP1. . . . . BYTE BASED(BYTE3ADR) ARRAY(2)          503
  495  0000H     2   TEMP1. . . . . BYTE BASED(BYTE3ADR) ARRAY(2)          499
  495  017AH     1   TEMP2. . . . . BYTE          496  499
  138  0216H    14   TERMINATE. . . . . PROCEDURE STACK=0008H          207  210  213  216  220  227  251  350
                     500  726  859
  300  0036H     2   TESTVALUE. . . . . WORD
  142  0000H    27   TOD. . . . . STRUCTURE BASED(TODADR)          262  263  264  265  269  271  302  303
                     304  305  316  327  335  336  343
  141  002EH     2   TODADR . . . . . WORD          142  262  263  264  265  269  271  302  303  304  305  316
                     327  334  335  336  343
  330  06A9H   106   TODASCII . . . . . PROCEDURE STACK=0024H          365
   20               TRUE . . . . . LITERALLY           69  280  394  463  612  621  685  707  714  721
  360  0004H     2   TSADR. . . . . WORD PARAMETER AUTOMATIC          362  364
  223  0004H     1   UB . . . . . BYTE PARAMETER AUTOMATIC          224  229
  201  0004H     1   UB . . . . . BYTE PARAMETER AUTOMATIC          202  219
  667  0077H    15   UFCB . . . . . BYTE ARRAY(15) DATA          673
   67  0006H    32   USED . . . . . WORD ARRAY(16)          669  691  721
  118  0004H     1   USER . . . . . BYTE PARAMETER AUTOMATIC          119  120
   67  00EFH    16   USER . . . . . BYTE ARRAY(16)          669  685  712  713  719  720
    6  006AH     1   USERCODE . . . . . BYTE           74  827
  701  0EA2H   153   USERSTATUS . . . . PROCEDURE STACK=001CH          837
  602  0004H     2   V. . . . . WORD PARAMETER AUTOMATIC          603  607  608
  541  0004H     2   V. . . . . WORD PARAMETER AUTOMATIC          542  544
  425  0008H     2   V. . . . . WORD PARAMETER AUTOMATIC          426  428  429
  247  0004H     1   VAL. . . . . BYTE PARAMETER AUTOMATIC          248  249
  424  0133H     7   VAL. . . . . BYTE ARRAY(7) INITIAL          444  453  466  482  491
  439  0004H     2   VALUE. . . . . WORD PARAMETER AUTOMATIC          440  444  445
    4  003CH     8   VERDATE. . . . . BYTE ARRAY(8) DATA
    5  0046H     9   VERSION. . . . . BYTE ARRAY(9) DATA          595
  724  0F38H    15   VERSIONERR . . . . PROCEDURE STACK=001AH          730  735  757
  290  010CH     1   WEEKDAY. . . . . BYTE          306  318
  240  0032H     2   WORDVALUE. . . . . WORD          243  244  284  286  296  305  306  309  312
  103  016BH    15   WRITEPROT. . . . . PROCEDURE STACK=0008H
  232  0006H     1   Y. . . . . BYTE PARAMETER AUTOMATIC          233  235  236
  251  0106H     1   YEAR . . . . . BYTE          259  260  262  279  282  287  309  323
  278  0034H     2   YEARLENGTH . . . . WORD          281  283  284  286
  234  0102H     1   YP . . . . . BYTE          235  237  238
  603  017EH     1   ZERO . . . . . BYTE          605  610  612
  450  0174H     1   ZEROSUP. . . . . BYTE          454  461  463
  425  0004H     1   ZEROSUP. . . . . BYTE PARAMETER AUTOMATIC          426  431  434
```

MODULE INFORMATION:

```
  CODE AREA SIZE     = 1206H    4614D
  CONSTANT AREA SIZE = 03E6H     998D
```

```
      VARIABLE AREA SIZE = 0168H     3920
      MAXIMUM STACK SIZE = 0034H      52D
      1524 LINES READ
      0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
ISIS-II MCS-86 LOCATER, V1.2 INVOKED BY:
:F0: SHOW.LNK OD(SM(CODE,DATS,DATA,STACK,CONST)) RD(SM(CODE(0),DATS(1D000H))) SS(STACK(+32)) TO SHOW.

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

BASE   OFFSET TYPE SYMBOL                BASE   OFFSET TYPE SYMBOL

0000H  1200H  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  RO
1000H  0080H  PUB  BUFF                  1000H  0080H  PUB  T3UFF
1000H  0080H  PUB  BUFFA                 1000H  005CH  PUB  FCBA
1000H  0100H  PUB  SAVEAX                1000H  0102H  PUB  SAVEBX
1000H  0104H  PUB  SAVECX                1000H  0106H  PUB  SAVEDX

SHOW:  SYMBOLS AND LINES
1000H  06D0H  SYM  MEMORY                1000H  02FCH  SYM  COPYRIGHT
1000H  0322H  SYM  VERDATE               1000H  032AH  SYM  VERSION
1000H  007DH  SYM  RREC                  1000H  007FH  SYM  ROVF
1000H  006DH  SYM  DOLL                  1000H  006EH  SYM  PARM
1000H  0172H  SYM  USERCODE              1000H  0173H  SYM  CVERSION
1000H  0174H  SYM  CDISK                 1000H  0108H  SYM  DPBPTR
1000H  0108H  BAS  DPB                   1000H  010CH  SYM  ALLOCA
1000H  010CH  BAS  ALLOC                 0000H  0102H  SYM  PRINTCHAR
STACK  0004H  SYM  CHAR                  0000H  0115H  SYM  PRINTB
0000H  0120H  SYM  PRINTX                STACK  0004H  SYM  A
STACK  0004H  BAS  S                     0000H  0139H  SYM  BREAK
0000H  0148H  SYM  CRLF                  0000H  0173H  SYM  PRINT
STACK  0004H  SYM  A                     1000H  0175H  SYM  DONT
0000H  0183H  SYM  GETVERSION            0000H  0192H  SYM  SELECT
STACK  0004H  SYM  D                     0000H  01A5H  SYM  OPEN
STACK  0004H  SYM  FCB                   1000H  0176H  SYM  ANYTHING
1000H  0177H  SYM  DIRBUF                1000H  01F7H  SYM  USER
1000H  010EH  SYM  USED                  1000H  012EH  SYM  NSFCB
1000H  0130H  SYM  FREEDIR               0000H  0168H  SYM  CHECKUSER
0000H  01F7H  SYM  SEARCH                STACK  0004H  SYM  FCB
STACK  0004H  BAS  FCB0                  0000H  0219H  SYM  SEARCHN
0000H  022EH  SYM  CSELECT               0000H  023DH  SYM  SETDMA
STACK  0004H  SYM  DMA                   0000H  024DH  SYM  GETALLOCA
0000H  025CH  SYM  GETLOGIN              0000H  0268H  SYM  WRITEPROT
0000H  027AH  SYM  GETRODISK             0000H  0289H  SYM  SETIND
0000H  0298H  SYM  SETDPB                0000H  02AFH  SYM  GETUSER
0000H  02BEH  SYM  SETUSER               STACK  0004H  SYM  USER
0000H  02D1H  SYM  GETFILESIZE           STACK  0004H  SYM  FCB
0000H  02E1H  SYM  GETFREESP             STACK  0004H  SYM  D
0000H  02F4H  SYM  GETLBL                STACK  0004H  SYM  D
1000H  0132H  SYM  PARSEFN               1000H  0108H  BAS  DELIMITER
0000H  0307H  SYM  PARSE                 0000H  0316H  SYM  TERMINATE
1000H  0136H  SYM  TODADR                1000H  0136H  BAS  TOD
1000H  0138H  SYM  STRINGADR             1000H  0138H  BAS  STRING
1000H  0207H  SYM  INDEX                 0000H  0324H  SYM  EMITCHAR
STACK  0004H  SYM  C                     0000H  0340H  SYM  EMITN
```

STACK	0004H	SYM	A	STACK	0004H	BAS	C
0000H	0368H	SYM	EMITBCD	STACK	0004H	SYM	P
0000H	0378H	SYM	EMITBCDPAIR	STACK	0004H	SYM	B
0000H	0393H	SYM	EMITCOLON	STACK	0004H	SYM	P
0000H	03A6H	SYM	EMITBINPAIR	STACK	0004H	SYM	B
0000H	03C0H	SYM	EMITSLANT	STACK	0004H	SYM	D
1000H	0208H	SYM	CHR	0000H	03E0H	SYM	CMC
0000H	0411H	SYM	DLBLANK	0000H	0422H	SYM	NUMERIC
0000H	0433H	SYM	SCANNUMERIC	STACK	0006H	SYM	LB
STACK	0004H	SYM	UB	1000H	0209H	SYM	B
0000H	04AAH	SYM	SCANDELIMITER	STACK	0008H	SYM	D
STACK	0006H	SYM	LB	STACK	0004H	SYM	UB
1000H	0333H	SYM	MONTHSIZE	1000H	02E4H	SYM	MONTHDAYS
0000H	04C3H	SYM	LEAPDAYS	STACK	0006H	SYM	Y
STACK	0004H	SYM	M	1000H	020AH	SYM	YP
1000H	013AH	SYM	WORDVALUE	0000H	04F7H	SYM	GETNEXTDIGIT
1000H	020BH	SYM	LSD	0000H	0517H	SYM	BCD
STACK	0004H	SYM	VAL	1000H	020CH	SYM	MONTH
1000H	020DH	SYM	DAY	1000H	020EH	SYM	YEAR
1000H	020FH	SYM	HRS	1000H	0210H	SYM	MIN
1000H	0211H	SYM	SEC	0000H	053EH	SYM	SETDATETIME
1000H	0212H	SYM	I	1000H	0213H	SYM	LEAPFLAG
0000H	065FH	SYM	BCDPAIR	STACK	0006H	SYM	A
STACK	0004H	SYM	B	0000H	0670H	SYM	COMPUTEYEAR
1000H	013CH	SYM	YEARLENGTH	1000H	0214H	SYM	WEEKDAY
1000H	033FH	SYM	DAYLIST	1000H	0215H	SYM	LEAPBIAS
0000H	06A5H	SYM	COMPUTEMONTH	1000H	0216H	SYM	DATETEST
1000H	013EH	SYM	TESTVALUE	0000H	06E0H	SYM	GETDATETIME
0000H	074AH	SYM	EMITDATETIME	0000H	0749H	SYM	TODASCII
STACK	0004H	SYM	PARAMETER	1000H	0140H	SYM	RET
1000H	0217H	SYM	LCLTOD	1000H	0142H	SYM	DATAPGADR
1000H	0142H	BAS	DATAPG	1000H	0144H	SYM	EXTRNLTODADR
1000H	0144H	BAS	EXTRNLTOD	1000H	0146H	SYM	RET
0000H	0813H	SYM	DISPLAYTS	STACK	0004H	SYM	TSADR
1000H	0232H	SYM	I	1000H	0148H	SYM	BPB
0000H	0853H	SYM	SETBPB	0000H	0870H	SYM	SELECTDISK
STACK	0004H	SYM	D	0000H	0884H	SYM	GETALLOC
STACK	0004H	SYM	I	1000H	0233H	SYM	ACCUM
1000H	0237H	SYM	IBP	0000H	08A2H	SYM	COMPARE
STACK	0004H	SYM	A	STACK	0004H	BAS	S
1000H	0238H	SYM	I	0000H	08D0H	SYM	SCAN
1000H	0239H	SYM	I	1000H	023AH	SYM	B
0000H	0945H	SYM	SETACC	STACK	0004H	SYM	B
0000H	0950H	SYM	FILL	STACK	0008H	SYM	S
STACK	0006H	SYM	F	STACK	0004H	SYM	C
STACK	0008H	BAS	A	1000H	023BH	SYM	VAL
1000H	0242H	SYM	FAC	1000H	0249H	SYM	F0
1000H	0250H	SYM	F1	1000H	0257H	SYM	F2
1000H	025EH	SYM	F3	1000H	0265H	SYM	F4
1000H	026CH	SYM	F5	1000H	0273H	SYM	F6
1000H	014AH	SYM	PTR	0000H	0973H	SYM	PDECIMAL
STACK	0008H	SYM	V	STACK	0006H	SYM	PREC
STACK	0004H	SYM	ZEROSUP	1000H	027AH	SYM	D
0000H	09D3H	SYM	GETBCD	STACK	0004H	SYM	VALUE
1000H	0158H	SYM	PREC	1000H	027BH	SYM	I
0000H	0A1FH	SYM	PRINTBCD	1000H	027CH	SYM	ZEROSUP
1000H	027DH	SYM	I	0000H	0A5EH	SYM	PCHAR
STACK	0004H	SYM	C	0000H	0A8DH	SYM	ADD
STACK	0006H	SYM	AP	STACK	0004H	SYM	BP
STACK	0006H	BAS	A	STACK	0004H	BAS	B

```
1000H  027EH  SYM  C
0000H  0A08H  SYM  P3BYTE
1000H  0280H  SYM  I
STACK  0004H  BAS  P3
STACK  0004H  SYM  BYTE3ADR
STACK  0004H  BAS  TEMP1
0000H  0B66H  SYM  SHL3BYTE
STACK  0004H  BAS  B3
0000H  0B93H  SYM  SHOWDRIVE
STACK  0006H  SYM  AK
STACK  0006H  BAS  KACCUM
0000H  0BCBH  SYM  COUNT
1000H  015AH  SYM  KA
1000H  015EH  SYM  I
0000H  0C1DH  SYM  DRIVESTATUS
1000H  0160H  BAS  B3
0000H  0D08H  SYM  PV
0000H  0D1FH  SYM  DISKSTATUS
1000H  0284H  SYM  D
0000H  0D75H  SYM  PARSEERROR
STACK  0004H  SYM  V
1000H  0286H  SYM  ZERO
0000H  0DF1H  SYM  PRCOUNT
STACK  0004H  SYM  RODISK
1000H  0166H  SYM  LOGIN
1000H  0287H  SYM  D
1000H  035BH  SYM  UFCB
1000H  0289H  SYM  J
1000H  016CH  SYM  EXTPTR
1000H  016EH  BAS  FMOD
0000H  0F04H  SYM  GL0
0000H  0FA2H  SYM  USERSTATUS
0000H  103BH  SYM  VERSIONERR
0000H  1060H  SYM  LOCKEDSTATUS
1000H  036AH  SYM  D
1000H  0390H  SYM  LABEL2
1000H  03D0H  SYM  LABEL4
1000H  028BH  SYM  LBL
1000H  0170H  SYM  FCBP
0000H  119AH  SYM  PRINTFN
0000H  11D1H  SYM  PARSENEXT
1000H  028FH  SYM  LASTDSEGBYTE
0000H  0102H  LIN     21
0000H  0111H  LIN     24
0000H  0118H  LIN     26
0000H  0120H  LIN     28
0000H  012BH  LIN     32
0000H  0133H  LIN     34
0000H  0139H  LIN     36
0000H  0148H  LIN     38
0000H  0148H  LIN     40
0000H  0157H  LIN     42
0000H  0168H  LIN     45
0000H  0173H  LIN     48
0000H  0179H  LIN     51
0000H  0183H  LIN     54
0000H  0192H  LIN     56
0000H  0195H  LIN     59
0000H  01A5H  LIN     61
0000H  01B4H  LIN     64

1000H  027FH  SYM  I
STACK  0004H  SYM  BYTE3ADR
1000H  0231H  SYM  HIGHBYTE
0000H  0B48H  SYM  SHR3BYTE
STACK  0004H  BAS  B3
1000H  028AH  SYM  TEMP2
STACK  0004H  SYM  BYTE3ADR
STACK  0004H  BAS  TEMP1
0000H  0B18H  SYM  ADDBLOCK
STACK  0004H  SYM  AB
STACK  0004H  BAS  BACCUM
STACK  0004H  SYM  MODE
1000H  015CH  SYM  BA
1000H  0283H  SYM  BIT
1000H  0160H  SYM  B3A
0000H  0CF0H  SYM  PV3
STACK  0004H  SYM  V
1000H  0162H  SYM  LOGIN
0000H  0D51H  SYM  HELP
0000H  0D94H  SYM  PVALUE
1000H  0285H  SYM  D
1000H  0164H  SYM  K
0000H  0E31H  SYM  STAT
0000H  0E5FH  SYM  PRSTATUS
1000H  0168H  SYM  RODISK
0000H  0E33H  SYM  GETUSRFILES
1000H  0288H  SYM  I
1000H  016AH  SYM  NFCBS
1000H  016EH  SYM  MODPTR
1000H  016CH  BAS  FEXT
0000H  0F6FH  SYM  GE2
1000H  028AH  SYM  I
0000H  104AH  SYM  OPENFILES
0000H  1076H  SYM  READLBL
1000H  0368H  SYM  LABEL1
1000H  03A8H  SYM  LABEL3
0000H  10ABH  SYM  LABELSTATUS
1000H  028CH  SYM  MAKE
1000H  0170H  BAS  FCBV
1000H  028DH  SYM  K
1000H  028EH  SYM  I
0000H  120BH  SYM  PLMSTART
0000H  0105H  LIN     23
0000H  0115H  LIN     25
0000H  011EH  LIN     27
0000H  0123H  LIN     31
0000H  0130H  LIN     33
0000H  0135H  LIN     35
0000H  013DH  LIN     37
0000H  0148H  LIN     39
0000H  0151H  LIN     41
0000H  015EH  LIN     44
0000H  0171H  LIN     47
0000H  0176H  LIN     50
0000H  017FH  LIN     52
0000H  0186H  LIN     55
0000H  0192H  LIN     57
0000H  01A1H  LIN     60
0000H  01A8H  LIN     63
0000H  01B8H  LIN     68
```

```
0000H  0183H  LIN     69
0000H  01C2H  LIN     71
0000H  01C8H  LIN     73
0000H  01E4H  LIN     75
0000H  01F3H  LIN     77
0000H  01F7H  LIN     79
0000H  0208H  LIN     83
0000H  0215H  LIN     85
0000H  021CH  LIN     87
0000H  022CH  LIN     89
0000H  0231H  LIN     91
0000H  023DH  LIN     93
0000H  0249H  LIN     96
0000H  0250H  LIN     98
0000H  025CH  LIN    100
0000H  026BH  LIN    102
0000H  026EH  LIN    104
0000H  027AH  LIN    106
0000H  0289H  LIN    108
0000H  028CH  LIN    110
0000H  0298H  LIN    112
0000H  02ADH  LIN    114
0000H  02B2H  LIN    116
0000H  02BEH  LIN    118
0000H  02CDH  LIN    121
0000H  02D4H  LIN    124
0000H  02E1H  LIN    126
0000H  02F0H  LIN    129
0000H  02F7H  LIN    132
0000H  0307H  LIN    135
0000H  0316H  LIN    137
0000H  0319H  LIN    139
0000H  0324H  LIN    146
0000H  033CH  LIN    149
0000H  0343H  LIN    153
0000H  035FH  LIN    155
0000H  0364H  LIN    157
0000H  036BH  LIN    160
0000H  0378H  LIN    162
0000H  0386H  LIN    165
0000H  0393H  LIN    167
0000H  039CH  LIN    170
0000H  03A6H  LIN    172
0000H  03B9H  LIN    175
0000H  03CDH  LIN    177
0000H  03D6H  LIN    180
0000H  03E0H  LIN    183
0000H  03EAH  LIN    185
0000H  03F3H  LIN    188
0000H  03FAH  LIN    191
0000H  0411H  LIN    193
0000H  041BH  LIN    195
0000H  0420H  LIN    197
0000H  0425H  LIN    199
0000H  0433H  LIN    201
0000H  0438H  LIN    205
0000H  0447H  LIN    207
0000H  0451H  LIN    209
0000H  0458H  LIN    211
0000H  0476H  LIN    213

0000H  0180H  LIN     70
0000H  01C4H  LIN     72
0000H  01CDH  LIN     74
0000H  01E6H  LIN     76
0000H  01F5H  LIN     78
0000H  01FAH  LIN     82
0000H  0212H  LIN     84
0000H  0219H  LIN     86
0000H  0229H  LIN     88
0000H  022EH  LIN     90
0000H  023DH  LIN     92
0000H  0240H  LIN     95
0000H  024DH  LIN     97
0000H  025CH  LIN     99
0000H  025FH  LIN    101
0000H  026BH  LIN    103
0000H  0278H  LIN    105
0000H  027DH  LIN    107
0000H  0289H  LIN    109
0000H  0296H  LIN    111
0000H  029BH  LIN    113
0000H  02AFH  LIN    115
0000H  02BEH  LIN    117
0000H  02C1H  LIN    120
0000H  02D1H  LIN    122
0000H  02DDH  LIN    125
0000H  02E4H  LIN    128
0000H  02F4H  LIN    130
0000H  0307H  LIN    133
0000H  030AH  LIN    136
0000H  0316H  LIN    138
0000H  0322H  LIN    140
0000H  0327H  LIN    148
0000H  0340H  LIN    150
0000H  034BH  LIN    154
0000H  0362H  LIN    156
0000H  0368H  LIN    158
0000H  0374H  LIN    161
0000H  037BH  LIN    164
0000H  038FH  LIN    166
0000H  0396H  LIN    169
0000H  03A2H  LIN    171
0000H  03A9H  LIN    174
0000H  03C9H  LIN    176
0000H  03D0H  LIN    179
0000H  03DCH  LIN    181
0000H  03E3H  LIN    184
0000H  03FCH  LIN    186
0000H  03F8H  LIN    189
0000H  040FH  LIN    192
0000H  0414H  LIN    194
0000H  041EH  LIN    196
0000H  0422H  LIN    198
0000H  0433H  LIN    200
0000H  0436H  LIN    204
0000H  043EH  LIN    206
0000H  044AH  LIN    208
0000H  0458H  LIN    210
0000H  0460H  LIN    212
0000H  0479H  LIN    214
```

```
0000H  0482H  LIN    215
0000H  048EH  LIN    217
0000H  0493H  LIN    219
0000H  04A3H  LIN    221
0000H  04AAH  LIN    223
0000H  04B0H  LIN    226
0000H  04B8H  LIN    228
0000H  04C8H  LIN    230
0000H  04CEH  LIN    235
0000H  04ECH  LIN    237
0000H  04F7H  LIN    239
0000H  04FAH  LIN    243
0000H  0512H  LIN    245
0000H  0517H  LIN    247
0000H  053EH  LIN    250
0000H  0541H  LIN    254
0000H  0560H  LIN    256
0000H  0574H  LIN    258
0000H  0593H  LIN    260
0000H  05A8H  LIN    262
0000H  0606H  LIN    264
0000H  0626H  LIN    267
0000H  063CH  LIN    269
0000H  0646H  LIN    271
0000H  065FH  LIN    273
0000H  0670H  LIN    276
0000H  0673H  LIN    279
0000H  0678H  LIN    281
0000H  0685H  LIN    283
0000H  0694H  LIN    285
0000H  069DH  LIN    287
0000H  06A3H  LIN    289
0000H  06A8H  LIN    292
0000H  06B4H  LIN    294
0000H  06C5H  LIN    296
0000H  06DEH  LIN    298
0000H  06E0H  LIN    301
0000H  06EDH  LIN    303
0000H  06F9H  LIN    305
0000H  070BH  LIN    307
0000H  0713H  LIN    309
0000H  0726H  LIN    311
0000H  0744H  LIN    313
0000H  074AH  LIN    315
0000H  0756H  LIN    318
0000H  076EH  LIN    321
0000H  077CH  LIN    323
0000H  0789H  LIN    325
0000H  0797H  LIN    327
0000H  07A7H  LIN    329
0000H  07ACH  LIN    333
0000H  07B8H  LIN    335
0000H  07C8H  LIN    338
0000H  07D3H  LIN    340
0000H  07D8H  LIN    343
0000H  07F7H  LIN    346
0000H  080AH  LIN    348
0000H  080FH  LIN    353
0000H  0816H  LIN    363
0000H  0829H  LIN    365
0000H  0483H  LIN    216
0000H  0491H  LIN    218
0000H  04A0H  LIN    220
0000H  04AAH  LIN    222
0000H  04A0H  LIN    225
0000H  04B3H  LIN    227
0000H  04BEH  LIN    229
0000H  04C8H  LIN    232
0000H  04D8H  LIN    236
0000H  04F0H  LIN    238
0000H  04F7H  LIN    241
0000H  0508H  LIN    244
0000H  0517H  LIN    246
0000H  051AH  LIN    249
0000H  053EH  LIN    252
0000H  054FH  LIN    255
0000H  0567H  LIN    257
0000H  0584H  LIN    259
0000H  05A8H  LIN    261
0000H  05F2H  LIN    263
0000H  061DH  LIN    265
0000H  062DH  LIN    268
0000H  0644H  LIN    270
0000H  065DH  LIN    272
0000H  0662H  LIN    275
0000H  0670H  LIN    277
0000H  0678H  LIN    280
0000H  067EH  LIN    282
0000H  068BH  LIN    284
0000H  0696H  LIN    286
0000H  06A1H  LIN    288
0000H  06A5H  LIN    291
0000H  06ADH  LIN    293
0000H  06C0H  LIN    295
0000H  06DCH  LIN    297
0000H  06DEH  LIN    299
0000H  06E3H  LIN    302
0000H  06F3H  LIN    304
0000H  06FFH  LIN    306
0000H  070EH  LIN    308
0000H  0721H  LIN    310
0000H  0729H  LIN    312
0000H  0748H  LIN    314
0000H  0740H  LIN    316
0000H  0768H  LIN    319
0000H  0775H  LIN    322
0000H  0783H  LIN    324
0000H  0790H  LIN    326
0000H  07A0H  LIN    328
0000H  07A9H  LIN    330
0000H  07B2H  LIN    334
0000H  07C1H  LIN    336
0000H  07CEH  LIN    339
0000H  07D6H  LIN    341
0000H  07E6H  LIN    345
0000H  07FAH  LIN    347
0000H  080CH  LIN    350
0000H  0813H  LIN    360
0000H  081BH  LIN    364
0000H  0830H  LIN    366
```

```
0000H  083CH  LIN   367
0000H  084FH  LIN   369
0000H  0856H  LIN   373
0000H  086EH  LIN   375
0000H  0873H  LIN   378
0000H  0880H  LIN   380
0000H  0887H  LIN   383
0000H  08A2H  LIN   386
0000H  0881H  LIN   391
0000H  08C4H  LIN   393
0000H  08D0H  LIN   395
0000H  0945H  LIN   398
0000H  0955H  LIN   401
0000H  08D3H  LIN   403
0000H  08E4H  LIN   405
0000H  08E8H  LIN   407
0000H  0903H  LIN   409
0000H  0909H  LIN   411
0000H  0939H  LIN   413
0000H  093FH  LIN   415
0000H  095DH  LIN   417
0000H  096CH  LIN   420
0000H  0977H  LIN   422
0000H  097DH  LIN   425
0000H  0987H  LIN   428
0000H  099DH  LIN   430
0000H  09B8H  LIN   432
0000H  09C4H  LIN   435
0000H  09CFH  LIN   438
0000H  09D6H  LIN   441
0000H  09E1H  LIN   443
0000H  0A03H  LIN   445
0000H  0A19H  LIN   447
0000H  0A1FH  LIN   449
0000H  0A61H  LIN   453
0000H  0A75H  LIN   455
0000H  0A7CH  LIN   458
0000H  0A84H  LIN   461
0000H  0A22H  LIN   463
0000H  0A2CH  LIN   465
0000H  0A46H  LIN   467
0000H  0A5AH  LIN   469
0000H  0A8DH  LIN   471
0000H  0A95H  LIN   474
0000H  0ABBH  LIN   476
0000H  0ACEH  LIN   478
0000H  0AD8H  LIN   480
0000H  0AE8H  LIN   483
0000H  0AFDH  LIN   485
0000H  0B12H  LIN   487
0000H  0B2CH  LIN   489
0000H  0B36H  LIN   491
0000H  0B44H  LIN   493
0000H  0B4BH  LIN   496
0000H  0B5DH  LIN   498
0000H  0B62H  LIN   500
0000H  0B69H  LIN   503
0000H  0B8FH  LIN   505
0000H  0B96H  LIN   507
0000H  0BA6H  LIN   509

0000H  0849H  LIN   368
0000H  0853H  LIN   372
0000H  0859H  LIN   374
0000H  0870H  LIN   376
0000H  087DH  LIN   379
0000H  0884H  LIN   381
0000H  08A2H  LIN   384
0000H  08A5H  LIN   387
0000H  08C0H  LIN   392
0000H  08CAH  LIN   394
0000H  0900H  LIN   396
0000H  0948H  LIN   400
0000H  0959H  LIN   402
0000H  08E0H  LIN   404
0000H  08E6H  LIN   406
0000H  08F2H  LIN   408
0000H  0903H  LIN   410
0000H  092CH  LIN   412
0000H  093DH  LIN   414
0000H  0943H  LIN   416
0000H  0960H  LIN   419
0000H  0974H  LIN   421
0000H  0979H  LIN   423
0000H  0980H  LIN   427
0000H  0993H  LIN   429
0000H  09A9H  LIN   431
0000H  09C0H  LIN   434
0000H  09C0H  LIN   437
0000H  09D3H  LIN   439
0000H  09DCH  LIN   442
0000H  09E8H  LIN   444
0000H  0A0DH  LIN   446
0000H  0A1BH  LIN   448
0000H  0A5EH  LIN   451
0000H  0A6EH  LIN   454
0000H  0A79H  LIN   457
0000H  0A7LH  LIN   460
0000H  0A89H  LIN   462
0000H  0A27H  LIN   464
0000H  0A38H  LIN   466
0000H  0A54H  LIN   468
0000H  0A5CH  LIN   470
0000H  0A90H  LIN   473
0000H  0AA1H  LIN   475
0000H  0AC6H  LIN   477
0000H  0AD4H  LIN   479
0000H  0AD3H  LIN   482
0000H  0AF5H  LIN   484
0000H  0B06H  LIN   486
0000H  0B19H  LIN   488
0000H  0B30H  LIN   490
0000H  0B41H  LIN   492
0000H  0B48H  LIN   494
0000H  0B5AH  LIN   497
0000H  0B5FH  LIN   499
0000H  0B66H  LIN   501
0000H  0B8CH  LIN   504
0000H  0B93H  LIN   506
0000H  0B9FH  LIN   508
0000H  0BA8H  LIN   510
```

```
0000H  0BA3H  LIN    514
0000H  0BBCH  LIN    516
0000H  0BC5H  LIN    518
0000H  0BC8H  LIN    520
0000H  0BD7H  LIN    524
0000H  0BEAH  LIN    526
0000H  0BF8H  LIN    528
0000H  0C0FH  LIN    530
0000H  0C1DH  LIN    532
0000H  0CF0H  LIN    535
0000H  0CF6H  LIN    537
0000H  0D03H  LIN    539
0000H  0D08H  LIN    541
0000H  0D13H  LIN    544
0000H  0D18H  LIN    546
0000H  0C26H  LIN    548
0000H  0C30H  LIN    550
0000H  0C3FH  LIN    552
0000H  0C51H  LIN    554
0000H  0C5BH  LIN    556
0000H  0C65H  LIN    558
0000H  0C79H  LIN    560
0000H  0C90H  LIN    562
0000H  0CA9H  LIN    564
0000H  0CC1H  LIN    566
0000H  0CD2H  LIN    568
0000H  0CE4H  LIN    570
0000H  0CEEH  LIN    572
0000H  0D22H  LIN    575
0000H  0D2DH  LIN    577
0000H  0D38H  LIN    580
0000H  0D45H  LIN    583
0000H  0D40H  LIN    585
0000H  0D51H  LIN    587
0000H  0D5BH  LIN    589
0000H  0D69H  LIN    591
0000H  0D73H  LIN    593
0000H  0D78H  LIN    595
0000H  0D82H  LIN    597
0000H  0D8CH  LIN    599
0000H  0D92H  LIN    601
0000H  0D97H  LIN    604
0000H  0DA2H  LIN    606
0000H  0DB5H  LIN    608
0000H  0DCBH  LIN    610
0000H  0DE2H  LIN    613
0000H  0DEDH  LIN    616
0000H  0DF4H  LIN    618
0000H  0E01H  LIN    621
0000H  0E0DH  LIN    624
0000H  0E1BH  LIN    626
0000H  0E29H  LIN    629
0000H  0E31H  LIN    631
0000H  0E37H  LIN    634
0000H  0E40H  LIN    636
0000H  0E4BH  LIN    638
0000H  0E53H  LIN    640
0000H  0E5FH  LIN    642
0000H  0E68H  LIN    646
0000H  0E73H  LIN    648

0000H  0BB3H  LIN    515
0000H  0BC0H  LIN    517
0000H  0BC7H  LIN    519
0000H  0BCEH  LIN    523
0000H  0BDAH  LIN    525
0000H  0BF1H  LIN    527
0000H  0C04H  LIN    529
0000H  0C16H  LIN    531
0000H  0C1DH  LIN    533
0000H  0CF3H  LIN    536
0000H  0CFDH  LIN    538
0000H  0D06H  LIN    540
0000H  0D0BH  LIN    543
0000H  0D18H  LIN    545
0000H  0C20H  LIN    547
0000H  0C2DH  LIN    549
0000H  0C37H  LIN    551
0000H  0C4AH  LIN    553
0000H  0C54H  LIN    555
0000H  0C62H  LIN    557
0000H  0C6CH  LIN    559
0000H  0C80H  LIN    561
0000H  0C97H  LIN    563
0000H  0CB0H  LIN    565
0000H  0CC8H  LIN    567
0000H  0CD9H  LIN    569
0000H  0CEBH  LIN    571
0000H  0D1FH  LIN    573
0000H  0D28H  LIN    576
0000H  0D34H  LIN    578
0000H  0D42H  LIN    581
0000H  0D49H  LIN    584
0000H  0D4FH  LIN    586
0000H  0D54H  LIN    588
0000H  0D62H  LIN    590
0000H  0D70H  LIN    592
0000H  0D75H  LIN    594
0000H  0D7FH  LIN    596
0000H  0D89H  LIN    598
0000H  0D8FH  LIN    600
0000H  0D94H  LIN    602
0000H  0D9DH  LIN    605
0000H  0DA9H  LIN    607
0000H  0DBFH  LIN    609
0000H  0DDDH  LIN    612
0000H  0DEBH  LIN    615
0000H  0DF1H  LIN    617
0000H  0DFBH  LIN    620
0000H  0E0BH  LIN    622
0000H  0E14H  LIN    625
0000H  0E22H  LIN    627
0000H  0E2FH  LIN    630
0000H  0E34H  LIN    633
0000H  0E3AH  LIN    635
0000H  0E47H  LIN    637
0000H  0E51H  LIN    639
0000H  0E5BH  LIN    641
0000H  0E62H  LIN    645
0000H  0E6EH  LIN    647
0000H  0E7AH  LIN    649
```

```
0000H  0E81H  LIN   651
0000H  0E93H  LIN   654
0000H  0E95H  LIN   657
0000H  0EA3H  LIN   661
0000H  0EABH  LIN   663
0000H  0EB1H  LIN   665
0000H  0EB6H  LIN   668
0000H  0ED3H  LIN   670
0000H  0EDFH  LIN   672
0000H  0EEDH  LIN   674
0000H  0EFAH  LIN   676
0000H  0F15H  LIN   679
0000H  0F21H  LIN   682
0000H  0F2CH  LIN   684
0000H  0F3DH  LIN   686
0000H  0F54H  LIN   690
0000H  0F69H  LIN   692
0000H  0F6FH  LIN   695
0000H  0F75H  LIN   697
0000H  0F8CH  LIN   699
0000H  0FA2H  LIN   701
0000H  0FA8H  LIN   704
0000H  0FAEH  LIN   706
0000H  0FC5H  LIN   708
0000H  0FCBH  LIN   710
0000H  0FD5H  LIN   712
0000H  0FEDH  LIN   714
0000H  0FFEH  LIN   716
0000H  1004H  LIN   718
0000H  1017H  LIN   720
0000H  1033H  LIN   722
0000H  103BH  LIN   724
0000H  1045H  LIN   726
0000H  104AH  LIN   728
0000H  1054H  LIN   730
0000H  105EH  LIN   732
0000H  1063H  LIN   734
0000H  106DH  LIN   736
0000H  1076H  LIN   738
0000H  1080H  LIN   741
0000H  108EH  LIN   743
0000H  10A4H  LIN   745
0000H  10A9H  LIN   747
0000H  119AH  LIN   758
0000H  11A9H  LIN   761
0000H  11B6H  LIN   763
0000H  11CFH  LIN   765
0000H  10B5H  LIN   767
0000H  10C2H  LIN   769
0000H  10CFH  LIN   772
0000H  10E5H  LIN   774
0000H  10EBH  LIN   776
0000H  10F9H  LIN   778
0000H  1107H  LIN   780
0000H  1115H  LIN   782
0000H  111FH  LIN   784
0000H  112BH  LIN   786
0000H  1137H  LIN   788
0000H  114CH  LIN   790
0000H  115CH  LIN   792

0000H  0E88H  LIN   653
0000H  0E95H  LIN   655
0000H  0E9CH  LIN   658
0000H  0EA7H  LIN   662
0000H  0EAFH  LIN   664
0000H  0EB3H  LIN   666
0000H  0EC2H  LIN   669
0000H  0ED9H  LIN   671
0000H  0EE6H  LIN   673
0000H  0EF3H  LIN   675
0000H  0F04H  LIN   677
0000H  0F19H  LIN   681
0000H  0F28H  LIN   683
0000H  0F2FH  LIN   685
0000H  0F44H  LIN   688
0000H  0F5DH  LIN   691
0000H  0F58H  LIN   693
0000H  0F72H  LIN   696
0000H  0F7CH  LIN   698
0000H  0FA0H  LIN   700
0000H  0FA5H  LIN   703
0000H  0FABH  LIN   705
0000H  0FB5H  LIN   707
0000H  0FC8H  LIN   709
0000H  0FD2H  LIN   711
0000H  0FE1H  LIN   713
0000H  0FF8H  LIN   715
0000H  1001H  LIN   717
0000H  1008H  LIN   719
0000H  1023H  LIN   721
0000H  1039H  LIN   723
0000H  103EH  LIN   725
0000H  1048H  LIN   727
0000H  1040H  LIN   729
0000H  1057H  LIN   731
0000H  1060H  LIN   733
0000H  106AH  LIN   735
0000H  1074H  LIN   737
0000H  1079H  LIN   740
0000H  1087H  LIN   742
0000H  10A2H  LIN   744
0000H  10A7H  LIN   746
0000H  10ABH  LIN   752
0000H  1190H  LIN   760
0000H  11B0H  LIN   762
0000H  11C9H  LIN   764
0000H  10AEH  LIN   766
0000H  10B8H  LIN   768
0000H  10CCH  LIN   771
0000H  10DEH  LIN   773
0000H  10E8H  LIN   775
0000H  10F2H  LIN   777
0000H  1102H  LIN   779
0000H  110EH  LIN   781
0000H  111CH  LIN   783
0000H  1122H  LIN   785
0000H  1130H  LIN   787
0000H  1147H  LIN   789
0000H  1153H  LIN   791
0000H  1161H  LIN   793
```

```
0000H  1168H  LIN     794
0000H  117AH  LIN     796
0000H  118CH  LIN     798
0000H  1195H  LIN     800
0000H  11D1H  LIN     802
0000H  11D8H  LIN     804
0000H  11E5H  LIN     806
0000H  11F6H  LIN     808
0000H  1203H  LIN     810
0000H  1208H  LIN     813
0000H  1214H  LIN     815
0000H  1220H  LIN     817
0000H  1239H  LIN     820
0000H  123FH  LIN     822
0000H  1255H  LIN     824
0000H  1261H  LIN     826
0000H  1260H  LIN     828
0000H  1277H  LIN     830
0000H  1287H  LIN     832
0000H  1293H  LIN     834
0000H  12A6H  LIN     836
0000H  12B2H  LIN     838
0000H  12BEH  LIN     840
0000H  12CCH  LIN     843
0000H  12D4H  LIN     845
0000H  12DDH  LIN     847
0000H  12E9H  LIN     850
0000H  12F5H  LIN     852
0000H  1301H  LIN     854
0000H  0102H  LIN     861

0000H  115FH  LIN     795
0000H  1181H  LIN     797
0000H  118EH  LIN     799
0000H  119BH  LIN     801
0000H  11D4H  LIN     803
0000H  11DEH  LIN     805
0000H  11E8H  LIN     807
0000H  11FAH  LIN     809
0000H  1209H  LIN     811
0000H  120EH  LIN     814
0000H  1219H  LIN     816
0000H  122CH  LIN     819
0000H  123DH  LIN     821
0000H  124CH  LIN     823
0000H  1256H  LIN     825
0000H  1267H  LIN     827
0000H  1274H  LIN     829
0000H  127EH  LIN     831
0000H  1290H  LIN     833
0000H  12A1H  LIN     835
0000H  12ADH  LIN     837
0000H  12B9H  LIN     839
0000H  12C5H  LIN     842
0000H  12D1H  LIN     844
0000H  12D6H  LIN     846
0000H  12E2H  LIN     848
0000H  12F0H  LIN     851
0000H  12FCH  LIN     853
0000H  1304H  LIN     855
```


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

SEGMENT MAP

START	STOP	LENGTH	ALIGN	NAME	CLASS
00000H	01305H	1306H	G	CODE	CODE
10000H	10107H	0108H	G	DATS	DATA
10108H	1028FH	0188H	W	DATA	DATA
10290H	102E3H	0054H	W	STACK	STACK
102E4H	106C9H	03E6H	W	CONST	CONST
106D0H	106D0H	0000H	G	??SEG	
106D0H	106D0H	0000H	W	MEMORY	MEMORY

GROUP MAP

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