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


        $ TITLE('SET: Sets BDOS/XFCB options for MP/M & CCP/M')
        $ COMPACT

        /* Revised:
            9/4/81 changes in upper case
            23 Jun 82  by Bill Fitler
            1/26/83 for CCPM 2.0 by F.Borda
        */

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

        $include (:f2:vaxcmd.lit)
=
=       /**** VAX commands for generation - read the name of this program
=               for PROGNAME below.
=
=           $ util := PROGNAME
=           $ ccpmsetup              ! set up environment
=           $ assign 'f$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.
=       ****/

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

                                * * *  SET  * * *

         * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
1       set:
        do;

        $include (:f2:vermpm.lit)
=
```

```
        =   /*   This utility requires MP/M or Concurrent function calls */
        =
        =   /****** commented out for CCP/M-86 :
        =   declare Ver$OS literally '11h',
        =       Ver$Needs$OS literally '''Requires MP/M-86''';
        =    ******/
        =
2   1   =   declare Ver$OS literally '14h',
        =       Ver$Needs$OS literally '''Requires Concurrent CP/M-86''';
        =
        =
3   1   =   declare Ver$Mask literally '0fdh';  /* mask out is_network bit */
        =
4   1   =   declare Ver$BDOS literally '30h';   /* minimal BDOS version rqd */
        =

5   1       declare copyright (*) byte data (
              ' Copyright (c) 1983, Digital Research ');

6   1       declare versiondate (*) byte data ('02/15/83');
7   1       declare version     (*) byte data ('SET 2.1',0);



8   1       declare
                true        literally '1',
                false       literally '0',
                dcl         literally 'declare',
                lit         literally 'literally',
                proc        literally 'procedure',
                addr        literally 'address',
                forever     literally 'while true',
                tab         literally '9',
                cr          literally '13',
                lf          literally '10',
                ctrlc       literally '3h',
                ctrlx       literally '18h',
                ctrlh       literally '8h';
```

```
$ eject
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                    * * *  MESSAGES  * * *

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



```
9  1        declare
                not$found  (*) byte data (' File not found',0),
                no$space   (*) byte data (' or no directory space',0),
                invalid    (*) byte data ('Invalid ',0),
                set$prot   (*) byte data ('[protect=on]',0),
                dirlabel   (*) byte data ('Directory Label ',0),
                option$set (*) byte data (' attribute set ',0),
                errATTRIB  (*) byte data (' Attribute.',0),
                read$only  (*) byte data ('read only',0),
                ro         (*) byte data (' (RO)',0),
                read$write (*) byte data ('read write (RW)',0),
                comma      (*) byte data (', ',0),
                set$to     (*) byte data ('set to ',0),
                error$msg  (*) byte data ('ERROR: ',0),
                readmode   (*) byte data ('READ',0),
                writemode  (*) byte data ('WRITE',0),
                deletemode (*) byte data ('DELETE',0),
                nopasswd   (*) byte data ('NONE',0),
                on         (*) byte data ('    on    ',0),
                off        (*) byte data ('    off   ',0),
                fail       (*) byte data ('Invalid drive attribute.',0),
                failed     (*) byte data ('Could not reset an open drive.',0),
                label$name (*) byte data ('Label'),
                errRDLBL   (*) byte data ('Directory Label does not exist.',0),
                errNOPASS  (*) byte data ('Assign a password to this file.',0),
                errENAB    (*) byte data
            ('Enable password protection first: SET d: [PROTECT=ON].',0),
                errCRAC    (*) byte data
            ('Cannot have both create and access time stamps.',0),
                errFORMAT  (*) byte data
            ('Directory needs to be reformatted for time/date stamping.',0),
                errFORM2   (*) byte data ('  Use "INITDIR d:"',0),
                err$nofile (*) byte data ('Option requires a file reference.',0);

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

                    * * *  CP/M INTERFACE * * *

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

$include (:f2:proces.lit)
=
=   /*
=       Proces Literals MP/M-8086 II
```

```
        =   */
        =
10   1  =   declare pnamsiz literally '8';
        =
11   1  =   declare pd$hdr literally 'structure
        =     (link word,thread word,stat byte,prior byte,flag word,
        =     name (8) byte,uda word,dsk byte,user byte,ldsk byte,luser byte,
        =     mem word';
        =
12   1  =   declare pd$structure literally 'pd$hdr,
        =     dvract word,wait word,org byte,net byte,parent word,
        =     cns byte,abort byte,conmode word,lst byte,sf3 byte,sf4 byte,sf5 byte,
        =     reservd (4) byte,pret word,scratch word)';
        =
13   1  =     declare psrun                lit '00',
        =             pspoll               lit '01',
        =             psdelay              lit '02',
        =             psswap               lit '03',
        =             psterm               lit '04',
        =             pssleep              lit '05',
        =             psdq                 lit '06',
        =             psnq                 lit '07',
        =             psflagwait           lit '08',
        =             psciowait            lit '09';
        =
14   1  =     declare pf$sys               lit '00001h',
        =             pf$keep              lit '00002h',
        =             pf$kernal            lit '00004h',
        =             pf$pure              lit '00008h',
        =             pf$table             lit '00010h',
        =             pf$resource          lit '00020h',
        =             pf$raw               lit '00040h',
        =             pf$ctlc              lit '00080h',
        =             pf$active            lit '00100h',
        =             pf$tempkeep          lit '00200h',
        =             pf$ctld              lit '00400h',
        =             pf$childabort        lit '00800h',
        =             pf$noctls            lit '01000h';
        =
15   1  =     declare pcm$ll               lit '00001h',
        =             pcm$ctls             lit '00002h',
        =             pcm$rout             lit '00004h',
        =             pcm$ctlc             lit '00008h',
        =             pcm$ctlo             lit '00080h',
        =             pcm$rsx              lit '00300h';

        =   $include (:f2:uda.lit)
        =
        =   /* MP/M-86 II  User Data Area format - August 8, 1981 */
        =
16   1  =   declare uda$structure lit 'structure (
        =       dparam          word,
        =       dma$ofst        word,
        =       dma$seg         word,
        =       func            byte,
        =       searchl         byte,
        =       searcha         word,
```

```
=       searchdbase     word,
=       dcnt            word,
=       dblk            word,
=       error$mode      byte,
=       mult$cnt        byte,
=       df$password     (8) byte,
=       pd$cnt          byte)';
=
```



```
17  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 */
            sectorlen literally '128',       /* sector length */
            user$code byte;                  /* current user code */


            /* reset drive mask */
18  1       declare reset$mask (16) address data (
              0000000000000001b,
              0000000000000010b,
              0000000000000100b,
              0000000000001000b,
              0000000000010000b,
              0000000000100000b,
              0000000001000000b,
              0000000010000000b,
              0000000100000000b,
              0000001000000000b,
              0000010000000000b,
              0000100000000000b,
              0001000000000000b,
              0010000000000000b,
              0100000000000000b,
              1000000000000000b );

19  1   mon1: procedure(f,a) external;
20  2       declare f byte, a address;
21  2       end mon1;

22  1   mon2: procedure(f,a) byte external;
23  2       declare f byte, a address;
24  2       end mon2;

            /* declare mon3 literally 'mon2a'; */

25  1   mon3: procedure(f,a) address external;
26  2       declare f byte, a address;
27  2       end mon3;

28  1   MON4: PROCEDURE (F,A) POINTER EXTERNAL;
29  2       DECLARE F BYTE, A ADDRESS;
30  2       END MON4;
```

```
                /********** SYSTEM FUNCTION CALLS **********************/

31   1      BOOT: PROCEDURE;
32   2          CALL MON1(0,0);
                /* reboot */
33   2          END BOOT;

34   1      printchar: procedure(char);
35   2          declare char byte;
36   2          call mon1(2,char);
37   2          end printchar;

38   1      printb: procedure;
                /* print blank character */
39   2          call printchar(' ');
40   2          end printb;

41   1      printx: procedure(a);
42   2          declare a address;
43   2          declare s based a byte;
44   2              do while s <> 0;
45   3              call printchar(s);
46   3              a = a + 1;
47   3              end;
48   2          end printx;

49   1      check$con$stat: procedure byte;
50   2          return mon2(11,0); /* console ready */
51   2          end check$con$stat;

52   1      crlf: procedure;
53   2          call printchar(cr);
54   2          call printchar(lf);

55   2          end crlf;

56   1      print: procedure(a);
57   2          declare a address;
                /* print the string starting at address a until the
                next 0 is encountered */
58   2          call crlf;
59   2          call printx(a);
60   2          end print;

61   1      get$version: procedure addr;
                /* returns current cp/m version # */
62   2          return mon3(12,0);
63   2          end get$version;


64   1      conin: procedure byte;
65   2          return mon2(6,0fdh);
66   2          end conin;
```

```
 67   1      select: procedure(d);
 68   2          declare d byte;
 69   2          call mon1(14,d);
 70   2          end select;

 71   1      open: procedure(fcb) byte;
 72   2          declare fcb address;
 73   2          return mon2(15,fcb);
 74   2          end open;

 75   1      search$first: procedure(fcb) byte;
 76   2          declare fcb address;
 77   2          return mon2(17,fcb);
 78   2          end search$first;

 79   1      search$next: procedure byte;
 80   2          return mon2(18,0);
 81   2          end search$next;

 82   1      cselect: procedure byte;
                 /* return current disk number */
 83   2          return mon2(25,0);
 84   2          end cselect;

 85   1      setdma: procedure(dma);
 86   2          declare dma address;
 87   2          call mon1(26,dma);
 88   2          end setdma;

 89   1      writeprot: procedure byte;
                 /* write protect the current disk */
 90   2          return mon2(28,0);
 91   2          end writeprot;

 92   1      getuser: procedure byte;
                 /* return current user number */
 93   2          return mon2(32,0ffh);
 94   2          end getuser;

 95   1      setuser: procedure(user);
 96   2          declare user byte;
 97   2          call mon1(32,user);
 98   2          end setuser;

 99   1      getfilesize: procedure(fcb);
100   2          declare fcb address;
101   2          call mon1(35,fcb);
102   2          end getfilesize;

               /* 0ff => return BDOS errors */
103   1      return$errors:
                 procedure(mode);
104   2          declare mode byte;
105   2            call mon1 (45,mode);
106   2          end return$errors;
```

```
107   1       setind: procedure(fcb) address;
108   2           dcl fcb addr;
109   2           call setdma(.passwd);
                  /* set file indicators for current fcb */
110   2           return mon3(30,fcb);
111   2           end setind;

                  /********** DISK PARAMETER BLOCK **********************/

112   1       declare
                  DPBPTR POINTER,
                  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';

113   1       set$dpb: procedure;
                  /* set disk parameter block values */
114   2           DPBPTR = MON4(31,0); /* base of dpb */
115   2           end set$dpb;

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

116   1       wrlbl: procedure(fcb) address;
117   2           declare fcb address;
118   2           call setdma(.passwd);    /* set dma=password */
119   2           return mon3(100,fcb);
120   2           end wrlbl;

121   1       getlbl: procedure(d) byte;
122   2           declare d byte;

123   2           return mon2(101,d);
124   2           end getlbl;

125   1       readxfcb: procedure(fcb);
126   2           declare fcb address;
127   2           call setdma(.passwd);    /* set dma=password */
128   2           call mon1(102,fcb);
129   2           end readxfcb;

130   1       wrxfcb: procedure(fcb) address;
131   2           declare fcb address;

132   2           call setdma(.passwd);
133   2           return mon3(103,fcb);
134   2           end wrxfcb;

135   1       declare
```

```
            PD$POINTER POINTER,
            PD$PTR     STRUCTURE (
                           OFF ADDRESS,
                           SEGMENT ADDRESS) AT (@PD$POINTER),
            pd         based PD$POINTER PD$STRUCTURE,

            PD$PARENT$POINTER POINTER,
            PD$PARENT$PTR     STRUCTURE (
                           OFF ADDRESS,
                           SEGMENT ADDRESS) AT (@PD$PARENT$POINTER),
            PD$PARENT      based PD$PARENT$POINTER PD$STRUCTURE;

136   1     DECLARE

            UDA$POINTER POINTER,
            UDA$PTR     STRUCTURE (
                            OFF ADDRESS,
                            SEGMENT ADDRESS) AT (@UDA$POINTER),
            UDA         BASED UDA$POINTER UDA$STRUCTURE,

            UDA$PARENT$POINTER POINTER,
            UDA$PARENT$PTR     STRUCTURE (
                                 OFF ADDRESS,
                                 SEGMENT ADDRESS) AT (@UDA$PARENT$POINTER),
            UDA$PARENT         BASED UDA$PARENT$POINTER UDA$STRUCTURE;


137   1     GET$PD$UDA: PROCEDURE;

138   2         PDPOINTER = MON4(156,0);
139   2         UDA$PTR.OFF = 0;
140   2         UDA$PTR.SEGMENT = PD.UDA;
141   2         END GET$PD$UDA;

142   1     reset$drv: procedure(drv) byte;
143   2         dcl drv byte;

144   2         return mon2(37,reset$mask(drv));
145   2         end reset$drv;

146   1     terminate: procedure;
147   2         call crlf;
148   2         call mon1 (0,0);
149   2         end terminate;
```

```
$ eject

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

                * * *  GLOBAL DATA  * * *

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

150  1          declare
                    fnam     literally '11',
                    fmod     literally '14',
                    frc      literally '15',
                    fln      literally '15',
                    fdm      literally '16',
                    fdl      literally '31',
                    ftyp     literally '9',
                    rofile   literally '9',      /* read/only file */
                    infile   literally '10',     /* invisible file */
                    archiv   literally '11',     /* archived  file */
                    attrb1   literally  '1',     /* attribute F1' */
                    attrb2   literally  '2',     /* attribute F2' */
                    attrb3   literally  '3',     /* attribute F3' */
                    attrb4   literally  '4';     /* attribute F4' */

151  1          declare
                    fcbp     address,
                    fcbv     based fcbp (32) byte,
                    fext     literally 'fcbv(12)';

152  1          declare
                    xfcb     (32) byte,
                    xfcbmode byte at (.xfcb(12));  /* password mode */

153  1          declare                          /* command buffer */
                    cmd (27) byte initial(0,'HELP       ',0),
                    passwd (17) byte;            /* password buffer */

154  1          declare
                    scase   byte initial(-1),    /* file attributes */
                    fileref byte initial(false), /* file reference  */
                    lblcmd  byte initial(false), /* label attribute */
                    xfcbcmd byte initial(false), /* xfcb  attribute */
                    wild    byte initial(false), /* file = a wildcard */
                    optdel  byte initial(false), /* delimiter = option */
                    option$found  byte initial(false),/* options exist */
                    time$opt byte initial(false),/* option = [time] */
                    password byte initial(false), /* file has password */
                    option  byte initial(false), /* cmd = a option */
                    mxstamp byte initial(false);/* separates create,update*/

155  1          declare                          /* parsing */
```

```
                more  byte initial(true),   /* more to parse */
                opt$adr     addr,           /* start of options */
                ibp         addr;           /* input buffer ptr */

156   1       declare
                (sav$dcnt, sav$searcha)     addr,
                sav$searchl  byte,
                dirbuf (128) byte;          /* used for searches */

157   1       declare
                cdisk      byte,            /* current disk */
                ver        addr;            /* version checking */

158   1       declare
                error$code addr;            /* for bdos returned
                                               errors */
159   1       declare
                parse$fn structure (
                    buff$adr  addr,
                    fcb$adr   addr),
                last$buff$adr addr;         /* used for parsing */

160   1       declare  /* file attribute bytes and values by scase */
                attr$byte (14) byte
                    /*     RW RO DIR SYS  A F F F F  A F F F F  */
                    initial(9, 9, 10, 10,11,1,2,3,4,11,1,2,3,4),
                attr$value (14) byte
                    /*     RW RO DIR SYS  A F F F F  A F F F F  */
                    initial(0, 1,  0,  1, 1,1,1,1,1, 0,0,0,0,0);

161   1       declare                 /* strings for match routine */
                attributes (*) byte data
                    ('kWRODISYARF1F2F3F4Attribute',0),
                values     (*) byte data
                    ('REWRDENO',0),
                boolean    (*) byte data
                    ('OFONValue, Use ON or OFF',0);

                /*       VALUES                 FILE ATTRIBUTES
                     mode   keyword          scase     attribute
                       0    READ               0      RW
                       1    WRITE              1      RO
                       2    DELETE             2      DIR
                       3    NONE               3      SYS
                                               4      ARCHIVE
                                               5      F1
                       BOOLEAN                 6      F2
                       0    OFF                7      F3
                       1    ON                 8      F4          */
```

```
$ eject


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

                    * * *  BASIC ROUTINES  * * *

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


                    /* help message */
162   1     help: procedure;
163   2         call print(.(tab,tab,tab,'SET EXAMPLES',0));
164   2         call print(.(cr,lf,'FOR FILES',cr,lf,cr,lf,
                        'set *.asm [rw, dir] ',tab,tab,tab,'(File Attributes)',0));
165   2         call print(.(
                        'set *.prl [ro, sys]',0));
166   2         call print(.(
                        'set *.dat [archive=on,f1=off,f2=on,f3=on]',0));
167   2         call print(.(
                        'set *.asm [password = xyz]',
                        tab,tab,'(Password Protection)',0));
168   2         call print(.('set *.asm [protect  = read]',
                        tab,tab,'(read, write, delete or none)',0));
169   2         call print(.(cr,lf,'FOR DRIVES',cr,lf,cr,lf,
                        'set [password = xyz]',tab,tab,tab,'(Label Password)',0));
170   2         call print(.('set [protect  = on] ',tab,tab,tab,'(Password Protection)',0));
171   2         call print(.('set [update   = on] ',tab,tab,tab,'(Update Time Stamps - on or off)',0));
172   2         call print(.('set [create   = on] ',tab,tab,tab,'(Creation Time Stamps - on or off)',0));
173   2         call print(.('set [access   = on] ',tab,tab,tab,'(Access Time Stamps - on or off)',0));
174   2         call print(.(
                        'set [default  = xyz]',tab,tab,tab,'(Default Password)',0));
175   2         call print(.('set a:[rw],   b:[ro]',tab,tab,tab,'(Drive Status)',0));
176   2         end help;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                    /* invalid command error */
177   1     perror: proc(msg);
178   2         dcl msg addr;

179   2         call print(.error$msg);
180   2         if ibp = 0 then
181   2             call printx(parse$fn.buff$adr);
          else
182   2             call printx(last$buff$adr);
183   2         call printx(.(' ?',0));
184   2         call print(.invalid);
185   2         call printx(msg);
186   2         call terminate;
187   2         end perror;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

```
                        /* parsing error */
188   1     parse$error: proc;

189   2         if option then
190   2             call perror(.('Parameter',0));
              else
191   2             call perror(.('File',0));
192   2         end parse$error;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


              /* parse the next lexical item in the command line
                 parse$fn must filled in with input parameters */
193   1     parse: procedure address;
194   2         declare p address;
195   2         declare c based p byte;

196   2         p = mon3(152,.parse$fn);
197   2         if p = 0FFFFh then
198   2             call parse$error;
199   2         else if p <> 0 then do;
201   3             if c = '[' then
202   3                 optdel = true;
203   3             else if c = ']' then
204   3                 optdel = false;
                  p = p + 1;
206   3             if c = ',' then
207   3                 p = p + 1;
208   3             last$buff$adr = parse$fn.buff$adr - 1;
209   3             parse$fn.buff$adr = p;
210   3             end;
              else
211   2             optdel = false;
212   2         return p;
213   2         end parse;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* parse a option value */
214   1     parse$value: proc;

              /* test for end */
215   2         if ibp = 0 then
216   2             call parse$error;

              /* more to go */
217   2         ibp = parse;
218   2         end parse$value;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* fill string @ s for c bytes with f */
219   1     fill:   proc(s,f,c);
220   2         dcl s addr,
                  (f,c) byte,
                  a based s byte;
```

```
221   2             do while (c:=c-1)<>255;
222   3             d = f;
223   3             s = s+1;
224   3             end;
225   2         end fill;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                            /* copy c bytes from s to d */
226   1     copy:   proc(s,d,c);
227   2         dcl (s,d) addr, c byte;
228   2         dcl a based s byte, b based d byte;

229   2             do while (c:=c-1)<>255;
230   3             b=a; s=s+1; d=d+1;
233   3             end;
234   2         end copy;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                            /* upper case character from console */
235   1     ucase:   proc byte;
236   2         dcl c byte;

237   2         if (c:=conin) >= 'a' then
238   2            if c < '{' then
239   2               return(c-20h);
240   2         return c;
241   2         end ucase;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                            /* get password and place in passwd */
242   1     getpasswd:   proc;
243   2         dcl (i,c) byte;

244   2         call print(.('Password ? ',0));
245   2     retry:
              call fill(.passwd,' ',8);
246   2             do i = 0 to 7;
247   3     nxtchr:
              if (c:=ucase) >= ' ' then
248   3                 passwd(i)=c;
249   3             if c = cr then
250   3                 go to exit;
251   3             if c = ctrlx then
252   3                 goto retry;
253   3             if c = ctrlh then do;
255   4                 if i<1 then
256   4                     goto retry;
257   4                 else do;
258   5                     passwd(i:=i-1)=' ';
259   5                     goto nxtchr;
260   5                     end;
261   4                 end;
262   3             if c = ctrlc then
```

```
263   3                 call terminate;       /* end of program */
264   3             end;
265   2       exit:
                  c = check$con$stat;             /* clear raw I/O mode */
266   2         end getpasswd;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* print drive name */
267   1       printdrv: procedure;

268   2             call printchar(cdisk+'A');
269   2             call printchar(':');
270   2             end printdrv;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* print file name */
271   1       printfn: procedure;
272   2             declare k byte;

273   2             call printdrv;

274   2                 do k = 1 to fnam;
275   3                 if k = ftyp then
276   3                   call printchar('.');
277   3                   call printchar(fcbv(k) and 7fh);
278   3                   end;
279   2             end printfn;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                         /* error message routine */
280   1       bdos$error:   procedure;
281   2          declare
                     code byte;

282   2           if (code:=high(error$code)) < 3 then do;
284   3               call print(.error$msg);
285   3               call printdrv;
286   3               call printb;
287   3               if code = 1 then
288   3                   call printx(.('BDOS Bad Sector',0));
289   3               if code=2 then do;
291   4                   call printx(.('Drive ',0));
292   4                   call printx(.read$only);
293   4                   end;
294   3               call terminate;
295   3               end;
296   2           call printx(.error$msg);
297   2           if code = 3 then
298   2               call printx(.read$only);
299   2           if code = 5 then
300   2               call printx(.('Currently Opened',0));
301   2           if code = 7 then
302   2               call printx(.('Wrong Password',0));
303   2           end bdos$error;
```

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


                        /* get address of FCB in dirbuf */
304   1     set$up$file: procedure(dir$index);
305   2         dcl dir$index byte;

306   2         if dir$index <> 0ffh then do;
308   3             sav$dcnt = UDA.DCNT;
309   3             sav$searchl = UDA.SEARCHL;
310   3             sav$searcha = UDA.SEARCHA;
311   3             fcbp = shl(dir$index,5) + .dirbuf;
312   3             fcbv(0) = fcb(0);                   /* set drive byte */
313   3             end;
314   2         end set$up$file;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /*  match command from command string */
315   1     match: proc(commands$adr, last$cmd) byte;
316   2         dcl (i,j,matched,scase,last$cmd) byte;
317   2         dcl
                  commands$adr       address,
                  commands based commands$adr (1) byte;

318   2         j = 0;
319   2         do scase = 0 to last$cmd;
320   3             matched = true;
321   3                 do i = 1 to 2;
322   4                 if commands(j) <> cmd(i) then
323   4                     matched = false;
324   4                 j = j + 1;
325   4                 end;
326   3             if matched then
327   3                 return scase;
328   3             end;
329   2         call perror(.errATTRIB);
330   2         end match;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* return boolean option value */
331   1     bool: procedure byte;

332   2             if match(.boolean,1) then
333   2                 return true;
                  else
334   2                 return false;
335   2             end bool;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* print boolean option value */
336   1     pbool: procedure(value);
337   2             declare
                        value byte;
```

```
338   2             call printx(.option$set);
339   2             if value then
340   2                 call printx(.('ON',0));
                    else
341   2                 call printx(.('OFF',0));
342   2             end pbool;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* print command */
343   1       printcmd: procedure;

344   2             call printx(.set$to);
345   2             cmd(12)=0;
346   2             call printx(.cmd(1));
347   2             end printcmd;

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

                        F I L E   A T T R I B U T E S

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


                        /* print attribute set */
348   1       printatt: procedure;


              /* test if attribute fcbv(i) is on */
349   2           attribute: procedure(i) byte;
350   3              declare i byte;

351   3              if rol(fcbv(i),1) then
352   3                 return true;
353   3              return false;
354   3              end attribute;


              /* print character c if attribute(b) is true */
355   2           prnt$attrib: procedure(b,c);
356   3              declare (b,c) byte;

357   3              if attribute(b) then
358   3                 call printchar(c);
359   3              end prnt$attrib;

              /* display attributes: sys,ro,a,f1-f4 */

360   2           call printx(.set$to);
361   2           if attribute(infile) then
362   2               call printx(.('system (SYS)',0));
                  else
363   2               call printx(.('directory (DIR)',0));
364   2           call printx(.(', ',0));
365   2           if attribute(rofile) then do;
```

```
367   3             call printx(.read$only);
368   3             call printx(.ro);
369   3             end;
                  else
370   2             call printx(.read$write);

371   2         call printchar(tab);
372   2         call prnt$attrib(archiv,'A');
373   2         call prnt$attrib(attrb1,'1');
374   2         call prnt$attrib(attrb2,'2');
375   2         call prnt$attrib(attrb3,'3');
376   2         call prnt$attrib(attrb4,'4');
377   2         end print$att;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* read current file attributes */
378   1       rd$attributes: procedure;

379   2         if scase = -1 then
380   2             if not wild then do;
382   3                 call setdma(.dirbuf);
383   3                 call set$up$file(search$first(.fcb));
384   3                 end;
385   2         end rd$attributes;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* set up file attributes */
386   1       set$attributes: procedure;

          /*-------------------------------------------------------------

          scase ranges from 0 - 13 :

          0 - RW   3 - SYS       6 - F2 (on)  9 - not Archived
          1 - RO   4 - ARCHIVED  7 - F3 (on) 10 - F1 (off) 12 - F3 (off)
          2 - DIR  5 - F1 (on)   8 - F4 (on) 11 - F2 (off) 13 - F4 (off)

          -------------------------------------------------------------*/

387   2         call rd$attributes;
388   2         if (scase := match(.attributes,8)) > 3 then do;
390   3             call parse$value;
391   3             if not bool then
392   3                 scase = scase * 5;
393   3             end;
394   2         if attr$value(scase) then
395   2             fcbv(attr$byte(scase)) = fcbv(attr$byte(scase)) or 80h;
                  else
396   2             fcbv(attr$byte(scase)) = fcbv(attr$byte(scase)) and 7fh;

397   2         end set$attributes;


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

D R I V E A T T R I B U T E S

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



```
                                                    /* set drive attributes */
398   1     setdrvstatus: procedure;
399   2             dcl code byte;

                                                    /* set the drive */
400   2         if (scase:=match(.attributes,1)) then
401   2             code = writeprot;                 /* RO */
              else
402   2             if ((cmd(1) = 'R') and (cmd(2) = 'W'))then
403   2             code = reset$drv(cdisk);          /* RW */
404   2             else do;                          /* Invalid drive option */
405   3                  call print(.fail);
406   3                  call terminate;
407   3                  end;

408   2         if code <> 0ffh then do;
410   3             call print(.('Drive ',0));
411   3             call printdrv;
412   3             call printb;
413   3             call printx(.set$to);
414   3             if scase then do;
416   4                 call printx(.read$only);
417   4                 call printx(.ro);
418   4                 end;
                  else
419   3                 call printx(.read$write);
420   3             end;
              else
421   2             call print(.failed);

422   2         scase = -1;
423   2         end setdrvstatus;
              /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* set default password */
424   1     defaultpass: procedure;

425   2             if (getlbl(cdisk) and 80h) <> 80h then /* Is drive password enabled? */
426   2                do;                                 /* If not, print and leave.   */
427   3                  call print(.errENAB);
428   3                  call terminate;
429   3                end;

430   2             call fill(.cmd(1),' ',8);
431   2             ibp = parse;              /* get password */
432   2             call mon1(106,.cmd(1));  /* set default password */
433   2             call print(.('Default Password ',0));
```

```
434   2             call printcmd;

435   2             CALL GET$PD$UDA;
436   2             PD$PARENT$PTR.SEGMENT = PD$PTR.SEGMENT;
437   2             PD$PARENT$PTR.OFF = PD.PARENT;
438   2             UDA$PARENT$PTR.SEGMENT = PD$PARENT.UDA;
439   2             UDA$PARENT$PTR.OFF = 0;
440   2             CALL MOVW(@UDA.DF$PASSWORD,@UDA$PARENT.DF$PASSWORD,4);

441   2             end defaultpass;


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

                        L A B E L   A T T R I B U T E S

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


                        /* read the directory label before
                           writing the label to preserve the
                           name, type, and stamps */
442   1       readlabel: procedure;
443   2           dcl (mode, dcnt) byte;


444   2       readlbl: proc;
445   3           dcl d byte data('?');

446   3           call setdma(.dirbuf);
447   3           dcnt = search$first(.d);
448   3               do while dcnt <> 0ffh;
449   4               if dirbuf(ror(dcnt,3) and 110$0000b)=20H then
450   4                   return;
451   4               dcnt = search$next;
452   4               end;

453   3           call print(.errRDLBL);
454   3           call terminate;
455   3           end readlbl;

456   2           if lblcmd then
457   2               return;
458   2           mode = getlbl(cdisk);
459   2           password = false;
460   2           if mode > 0 then do;
462   3               call readlbl;
463   3               fcbp = shl(dcnt,5) + .dirbuf;
464   3               fext = fext and 11110000b;      /* turn off set passwd */
465   3               if fcbv(16) <> ' ' then
466   3                   if fcbv(16) <> 0 then
467   3                       password = true;
468   3               end;
469   2           else do;
470   3               fcbp = .fcb;
471   3               call copy(.label$name,.fcb(1),length(label$name));
472   3               end;
```

```
473   2           if password then
474   2               call getpassw;
475   2           lblcmd = true;
476   2           end readlabel;


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

                        X F C B     A T T R I B U T E S

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


                        /* read xfcb into xfcb buffer */
477   1       set$up$xfcb: procedure;

478   2               if not xfcbcmd then do;
480   3                   call copy(.fcbv,.xfcb,12);
481   3                   password,xfcbmode = 0;
482   3                   call readxfcb(.xfcb);        /* read xfcb */
483   3                   if xfcbmode then
484   3                       password = true;
485   3                   xfcbcmd = true;
486   3                   end;
                     /* else
                            already done */
487   2           end set$up$xfcb;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* no directory label exists */
488   1       no$label: procedure(msg);
489   2           declare msg addr;

490   2           call crlf;
491   2           call print(.error$msg);
492   2           call printx(.(' First SET ',0));
493   2           call printdrv;
494   2           call printx(msg);
495   2           call terminate;
496   2           end no$label;


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

                    PASSWORD  AND  PASSWORD  MODE  ROUTINES

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


                        /* set file or label password */
497   1       set$password: procedure;
498   2           dcl (p,q) address;
499   2           dcl c based p byte;
500   2           dcl d based q byte;
```

```
501   2          if (getlbl(cdisk) and 80h) <> 80h then /* Is the drive passwd enabled? */
502   2             do;                                 /* if not, print and leave      */
503   3                call print(.errENAB);
504   3                call terminate;
505   3             end;
506   2              if fileref then do;
                        /* if getlbl(cdisk) = 0 then
                             call no$label(.set$prot);*/
508   3                  call set$up$xfcb;              /* read xfcb */
509   3                  xfcbmode = xfcbmode or 1;      /* set passwd */
510   3                  end;
511   2              else do;
512   3                  call readlabel;
513   3                  fext = fext or 1;
514   3                  end;
515   2              p = (q:=parse$fn.buff$adr) - 1;
516   2              if c = ',' or d = ']' then         /* null password */
517   2                  call fill(.passwd(8),' ',8);
518   2              else do;
519   3                  ibp = parse;                   /* parse password */
520   3                  call copy(.cmd(1),.passwd(8),8); /* copy it to fcb */
521   3                  password = true;
522   3                  end;
523   2          end set$password;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* set file or drive protection mode */
524   1      protect: procedure;
525   2          declare new$password byte;

526   2          zeropass: proc;
527   3              xfcbmode = 1;
528   3              call fill(.passwd(8),' ',8);
529   3              password = true;
530   3              end zeropass;


531   2          call parse$value;              /* protection value */
532   2          if fileref then  do;
534   3              if ((getlbl(cdisk) and 80h) <> 80h) then
535   3                  do;                    /* Must set protect=on first */
536   4                  call print(.errENAB);
537   4                  call terminate;
538   4                  end;
539   3              call set$up$xfcb;
540   3              if xfcbmode then           /* lsb    */
541   3                  new$password = true;   /* save   */
                    else
542   3                  new$password = false;


543   3                   do case match(.values,3);
544   4                    xfcbmode = 80h;      /* READ   */
545   4                    xfcbmode = 40h;      /* WRITE  */
546   4                    xfcbmode = 20h;      /* DELETE */
547   4                    call zeropass;       /* NONE   */
```

```
548   4                   end;

549   3                   if newpassword then       /* restore */
550   3                      xfcbmode = xfcbmode or 1;


551   3            end;
552   2           else do;            /* Not a file ref, do the label */
553   3              call readlabel;
554   3              if bool then
555   3                  fext = fext or 80h;         /* turn on passwords */
                     else
556   3                  fext = fext and 01111111b;  /* turn off passwords */
557   3              end;
558   2           end ;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


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

                          LABEL  ATTRIBUTE  ROUTINES

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


                          /* gets the label option boolean value */
559   1       getbool: procedure;

560   2           if fileref then
561   2              call parse$error;
562   2           call readlabel;          /* get label name */
563   2           call parse$value;           /* option value */
564   2           end getbool;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                          /* sets the label name */
565   1       lname: procedure;

566   2           call getbool;
567   2           call copy(.cmd(1),.fcbv(1),11);       /* copy label name */
568   2           end lname;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                          /* set access time stamping */
569   1       access: procedure;

570   2             call getbool;
571   2             if not bool then
572   2                fext = fext and 10111111b;        /* turn off access ts */
                    else
573   2                do;
574   3                 if mxstamp then                  /* Create has also been chosen */
575   3                    do;
576   4                      call print(.errCRAC);
```

```
577   4                    call terminate;
578   4                  end;
                       else
579   3                  do;
580   4                    fext = fext or 40h;          /* turn on access ts */
581   4                    fext = fext and 11101111b; /* turn off create ts */
582   4                    mxstamp = true;              /* Mark 1 of 2 as chosen */
583   4                  end;
584   3              end;
585   2          end access;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* set update time stamping */
586   1          update: procedure;

587   2              call getbool;
588   2              if not bool then
589   2                fext = fext and 11011111b;        /* turn off update ts */
                     else
590   2                fext = fext or 20h;               /* turn on update ts */
591   2          end update;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* set create time stamping */
592   1          create: procedure;

593   2              call getbool;
594   2              if not bool then
595   2                fext = fext and 11101111b;        /* turn off create ts*/
                     else
596   2                do;
597   3                  if mxstamp then                 /* Access has also been chosen */
598   3                  do;
599   4                    call print(.errCRAC);
600   4                    call terminate;
601   4                  end;
                       else
602   3                  do;
603   4                    fext = fext or 10h;           /* turn on create ts */
604   4                    fext = fext and 10111111b;    /* turn off access ts*/
605   4                    mxstamp = true;               /* Mark 1 of 2 mx stamps */
606   4                  end;                            /* as being chosen   */
607   3                end;
608   2          end create;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


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

                     S H O W   L A B E L   &   X F C B

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

```
                        /* display the new password */
609   1     show$passwd: procedure;

610   2         call printx(.('Password = ',0));
611   2         passwd(16) = 0;
612   2         call printx(.passwd(8));
613   2         end show$passwd;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


          /* HEADER for showlbl procedure */

614   1     dcl label1 (*) byte data (
          'Directory       Passwds    Stamp     Stamp     Stamp',cr,lf,
          'Label           Reqd       Create    Access    Update',cr,lf,
          '--------------  -------    --------  --------  --------',cr,lf,0);

                        /* show the label options */
615   1     showlbl: procedure;
616   2         declare (make,access) byte;

617   2         call print(.('Label for drive ',0));
618   2         call printdrv;
619   2         call crlf;
620   2         call print(.label1);
621   2         call printfn;

              /* PASSWORDS REQUIRED */
622   2         if (fext and 80h) = 80h then
623   2             call printx(.on);
              else
624   2             call printx(.off);

              /* STAMP CREATE */
625   2         if (fext and 10h) = 10h  then
626   2            call printx(.on);
              else
627   2            call printx(.off);

              /* STAMP ACCESS */
628   2         if (fext and 40h) = 40h then
629   2              call printx(.on);
              else
630   2             call printx(.off);

              /* STAMP UPDATE */
631   2         if (fext and 20h) = 20h then
632   2             call printx(.on);
              else
633   2             call printx(.off);

634   2         call crlf;
635   2         if fext then do;
637   3             call crlf;
```

```
638   3              call show$passwd;
639   3              end;
640   2          end showlbl;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* display xfcb attributes */
641   1       show$xfcb: procedure;

642   2           if xfcbmode = 0 then do;     /* xfcbmode = true status of file */
644   3              if not(password) then do;/* must first have a password     */
646   4                       call printx(.error$msg);
647   4                       call printx(.errNOPASS);
648   4                       return;         /* error condition */
649   4                       end;
650   3              else do;                 /* No protection at all. This = default */
651   4                   call printx(.('Protection = ',0));
652   4                   call printx(.nopasswd);
653   4                   end;
654   3               end;
655   2           else do;
656   3               call printx(.('Protection = ',0));
657   3               if (xfcbmode and 80h) = 80h then
658   3                   call printx(.readmode);
659   3               else if (xfcbmode and 40h) = 40h then
660   3                   call printx(.writemode);
661   3               else if (xfcbmode and 20h) = 20h then
662   3                   call printx(.deletemode);
                    /* else if (not xfcbmode) or (passwd(8) = ' ') then
                        call printx(.nopasswd);
                    else
                        call printx(.readmode);*/

                  /*  if xfcbmode then
                        do;
                          call printx(.comma);
                          call show$passwd;
                     end;*/
                    end;

664   2           end ;

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

                   WRITE  XFCB, LABEL  AND  FILE  ATTRIBUTES

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




                        /* display the file or xfcb */
665   1       put$file: procedure;

666   2           call crlf;
```

```
667   2          call printfn;
668   2          call printb;
669   2          call printo;
670   2          end put$file;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                           /* write file attributes */
671   1      put$attributes: procedure;
672   2          if (not filepf) then do;
674   3             call print(.err$nofile);
675   3             call terminate;
676   3             end;
677   2          error$code = setind(fcbp);
678   2          if low(error$code) = 0ffh then
679   2              if high(error$code) <> 0 then do;
681   3                  call put$file;
682   3                  call bdos$error;
683   3                  if high(error$code) = 7 then do;
685   4                      call crlf;
686   4                      call getpasswd;
687   4                      call crlf;
688   4                      error$code = setind(fcbp);
689   4                      if high(error$code) <> 0 then do;
691   5                          call put$file;
692   5                          call bdos$error;
693   5                          end;
694   4                      end;
695   3                  end;
                       else
696   2                      call printx(.not$found);
697   2          if low(error$code) <> 0ffh then
698   2              if fext <= extmsk then do;
700   3                  call put$file;
701   3                  call print$att;
702   3                  end;
703   2          scase = -1;
704   2          end put$attributes;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                           /* write new label */
705   1      write$label: procedure;
706   2          err: proc;
707   3              call print(.dirlabel);
708   3              call bdos$error;
709   3              end err;

710   2          error$code = wrlbl(fcbp);
711   2          if low(error$code) = 0ffh then
712   2              if high(error$code) <> 0 then do;
714   3                  call err;
715   3                  if high(error$code) = 7 then do;
717   4                      call crlf;
718   4                      call getpasswd;
719   4                      error$code = wrlbl(fcbp);
720   4                      if high(error$code) <> 0 then do;
```

```
722   5                     call err;
723   5                     call terminate;
724   5                     end;
725   4                 call crlf;
726   4                 end;
727   3             end;
728   2         else do;
729   3             call print(.errFORMAT);
730   3             call print(.errFORM2);
731   3             call terminate;
732   3             end;

          /* successful */
733   2         call showlbl;
734   2         lblcmd = false;
735   2         end write$label;
      /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                  /* write out new xfcb */
736   1     write$xfcb: procedure;

737   2         call put$file;
738   2         error$code = wrxfcb(.xfcb);
739   2         if low(error$code) = 0ffh then
740   2             if high(error$code) <> 0 then do;
742   3                 call bdos$error;
743   3                 if high(error$code) = 7 then do;
745   4                     call crlf;
746   4                     call getpasswd;
747   4                     call crlf;
748   4                     call put$file;
749   4                     error$code = wrxfcb(.xfcb);
750   4                     if high(error$code) <> 0 then
751   4                         call bdos$error;
752   4                     end;
753   3                 end;
754   2             else do;
755   3                 call printx(.not$found);
756   3                 call printx(.no$space);
757   3                 end;
758   2         if low(error$code) <> 0ffh then
759   2             call show$xfcb;
760   2         xfcbcmd = false;
761   2         end write$xfcb;



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

                C O M M A N D   P R O C E S S I N G

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

```
                        /* select the disk specified in cmd line */
762   1     setdisk: procedure;
763   2         if cmd(0) <> 0 then do;
765   3             cdisk = cmd(0)-1;
766   3             call select(cdisk);
767   3             call set$dpb;
768   3             end;
769   2         end setdisk;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* find the next file matching the wildcard */
770   1     getfile: procedure byte;
771   2         declare
                      dir$index byte;

772   2         call setdma(.dirbuf);
773   2         if wild then do;
775   3             UDA.DCNT = sav$dcnt;
776   3             UDA.SEARCHL = sav$searchl;
777   3             UDA.SEARCHA = sav$searcha;
778   3             dir$index = search$next;
779   3             end;
          /* else */
780   2             dir$index = search$first(.fcb);
781   2         if dir$index <> 0ffh then do;
783   3             call set$up$file(dir$index);
784   3             return true;
785   3             end;
          /* else */
786   2             return false;
787   2         end getfile;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* test if the file is a wildcard */
788   1     wildcard: procedure byte;
789   2         declare
                      i byte;

790   2             do i=1 to fnam;
791   3             if fcb(i) = '?' then
792   3                 return true;
793   3             end;
794   2         return false;
795   2         end wildcard;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* set up the next file or drive reference */
796   1     setup$fcb: procedure;

797   2         call setdisk;
798   2         call copy(.cmd,.fcb,12);        /* name */
799   2         call copy(.cmd(16),.passwd,8); /* password */
800   2         time$opt, option$found = false;
801   2         if fcb(1) <> ' ' or fcb(ftyp) <> ' ' then do;
```

```
803   3             fileref = true;
804   3             if wildcard then
805   3                 if getfile then do;
807   4                     wild = true;
808   4                     opt$adr = parse$fn.buff$adr;
809   4                     end;
810   3                 else do;
811   4                     call print(.not$found);
812   4                     call terminate;
813   4                     end;
                      else
814   3                     fcbp = .fcb;
815   3                 end;
                  else
816   2                 fileref = false;
817   2             end setup$fcb;
              /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                            /* parse next option */
818   1         parse$option: procedure;

819   2             if cmd(1) = 'A' then do;                    /* A */
821   3                 if cmd(2) = 'C' then
822   3                     call access;
823   3                 else if fileref then
824   3                     call set$attributes;
                      else
825   3                     call parse$error;
826   3                 end;
827   2             else if cmd(1) = 'C' then                   /* C */
828   2                 call create;
829   2             else if cmd(1) = 'D' then do;               /* D */
831   3                 if fileref then
832   3                     call set$attributes;
833   3                 else if cmd(2) = 'E' then
834   3                     call defaultpass;
                      else
835   3                     call parse$error;
836   3                 end;
837   2             else if cmd(1) = 'F' then                   /* F */
838   2                 call set$attributes;
839   2             else if cmd(1) = 'H' then                   /* H */
840   2                 call help;
841   2             else if cmd(1) = 'N' then                   /* N */
842   2                 call lname;
843   2             else if cmd(1) = 'P' then do;               /* P */
845   3                 if cmd(2) = 'R' then
846   3                     call protect;
847   3                 else if cmd(2) = 'A' then
848   3                     call set$password;
                      else
849   3                     call parse$error;
850   3                 end;
851   2             else if cmd(1) = 'R' then do;               /* R */
853   3                 if fileref then
854   3                     call set$attributes;
```

```
                else
855   3                 call setdrvstatus;
856   3             end;
857   2         else if cmd(1) = 'S' and fileref then     /* S */
858   2             call set$attributes;
859   2         else if cmd(1) = 'U' then                 /* U */
860   2             call update;
          2      else
861   2             call parse$error;
862   2         end parse$option;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* check for more to parse */
863   1     is$there$more: proc;

864   2         if ibp = 0 then do;
866   3             if not option$found then do;
868   4                 call print(.error$msg);
869   4                 call printx(.('Invalid Command Parameter, try SET [HELP].',0));
870   4                 call terminate;
871   4                 end;
872   3             if not wild then
873   3                 more = false;
874   3             end;
875   2         end is$there$more;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* check for SET HELP */
                        /* REMOVED FOR CONSISTANCY WITH SDIR
          help$check: proc;
              declare i byte;

                  do i=1 to 11;
                  if fcb(i) <> cmd(i) then
                      return;
                  end;
              call help;
              call terminate;
              end help$check;
                        */

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

                              M A I N   P R O G R A M

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



876   1     declare
                  i              byte   initial (1),
                  last$dseg$byte byte   initial (0);


877   1     PLMSTART:
```

```
                procedure public;
               /* process request */
878   2            ver = get$version;
879   2            if low(ver) < Ver$BDOS  or  (high(ver) and Ver$Mask) <> Ver$OS then
880   2                call print(.(Ver$Needs$OS,0));
                   else
881   2                do;
               /*         call help$check;  */
                           /* scan for global option */
882   3                    do while buff(i)=' ';
883   4                    i = i + 1;
884   4                    end;
885   3                if buff(i) = '[' then do;
887   4                    option, optdel, option$found = true;
888   4                    parse$fn.buff$adr = .buff(i+1);
889   4                    end;
                       else
890   3                    parse$fn.buff$adr = .buff(1);
891   3                last$buff$adr = .buff(1);       /* used by perror routine */
892   3                parse$fn.fcb$adr = .cmd;
893   3                user$code = getuser;
894   3                call GET$PD$UDA;        /* get process descriptor */
895   3                call set$dpb;                   /* get disk parameter blk */
896   3                cdisk=cselect;          /* get current disk       */
897   3                ibp = parse;
898   3                    do while more;
899   4                    call is$there$more;
900   4                    if option then
901   4                        call parse$option;
902   4                    else if more then
903   4                        call setup$fcb;         /* file or drive reference */

                           if optdel then
905   4                        option, option$found = true;
906   4                    else do;
907   5                        option = false;
908   5                        call return$errors(0FFh);  /* bdos return errors */
909   5                        if lblcmd then             /* label options */
910   5                            call write$label;
911   5                        if scase <> -1 then        /* file attributes */
912   5                            call put$attributes;
913   5                        if xfcbcmd then            /* xfcb attributes */
914   5                            call write$xfcb;
915   5                        call return$errors(0);
916   5                        if wild then
917   5                            if getfile then do;
919   6                                parse$fn.buff$adr = opt$adr;
920   6                                option, optdel = true;
921   6                                end;
                                   else
922   5                                wild = false;
923   5                        end;
924   4                    call is$there$more;
925   4                    ibp = parse;
926   4                    end;
927   3                end;
928   2            call terminate;
```

```
929   2       END PLMSTART;
930   1     end set;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES											
41	0004H	2	A.	WORD PARAMETER AUTOMATIC			42	43	44	45	45			
19	0000H	2	A.	WORD PARAMETER	20									
220	0000H	1	A.	BYTE BASED(S)	222									
56	0004H	2	A.	WORD PARAMETER AUTOMATIC			57	59						
22	0000H	2	A.	WORD PARAMETER	23									
28	0000H	2	A.	WORD PARAMETER	29									
228	0000H	1	A.	BYTE BASED(S)	230									
25	0000H	2	A.	WORD PARAMETER	26									
135	0021H	1	ABORT.	BYTE MEMBER(PD)										
135	0021H	1	ABORT.	BYTE MEMBER(PDPARENT)										
569	0AA9H	67	ACCESS	PROCEDURE STACK=0030H			822							
616	0133H	1	ACCESS	BYTE										
8			ADDR	LITERALLY	61	108	155	156	157	158	159	178	220	227	489
150			ARCHIV	LITERALLY	372									
150			ATTRB1	LITERALLY	373									
150			ATTRB2	LITERALLY	374									
150			ATTRB3	LITERALLY	375									
150			ATTRB4	LITERALLY	376									
160	0109H	14	ATTRBYTE	BYTE ARRAY(14) INITIAL				395	396					
349	066EH	32	ATTRIBUTE.	PROCEDURE BYTE STACK=0004H					357	361	365			
161	0271H	28	ATTRIBUTES	BYTE ARRAY(28) DATA	388			400						
160	0117H	14	ATTRVALUE.	BYTE ARRAY(14) INITIAL				394						
355	0006H	1	B.	BYTE PARAMETER AUTOMATIC				356	357					
228	0000H	1	B.	BYTE BASED(D)	230									
280	0487H	117	BDOSERROR.	PROCEDURE STACK=001AH				682	692	708	742	751		
112			BLKMSK	LITERALLY										
112			BLKSHF	LITERALLY										
112	0002H	1	BLS.	BYTE MEMBER(DPB)										
112	0003H	1	BMS.	BYTE MEMBER(DPB)										
331	05A3H	25	BOOL	PROCEDURE BYTE STACK=0028H					391	554	571	588	594	
161	0296H	25	BOOLEAN.	BYTE ARRAY(25) DATA	332									
31	0002H	14	BOOT	PROCEDURE STACK=0008H										
17	0000H	128	BUFF	BYTE ARRAY(129) EXTERNAL(2)					882	885	888	890	891	
17			BUFFA.	LITERALLY										
159	0000H	2	BUFFADR.	WORD MEMBER(PARSEFN)			181	208	209	515	808	888	890	919
219	0004H	1	C.	BYTE PARAMETER AUTOMATIC				220	221					
243	0127H	1	C.	BYTE	247	248	249	251	253	262	265			
499	0000H	1	C.	BYTE BASED(P)	516									
226	0004H	1	C.	BYTE PARAMETER AUTOMATIC				227	229					
355	0004H	1	C.	BYTE PARAMETER AUTOMATIC				356	358					
236	0125H	1	C.	BYTE	237	238	239	240						
195	0000H	1	C.	BYTE BASED(P)	201	203	206							
157	0108H	1	CDISK.	BYTE	268	403	425	458	501	534	765	766	896	
34	0004H	1	CHAR	BYTE PARAMETER AUTOMATIC				35	36					
49	0047H	15	CHECKCONSTAT . . .	PROCEDURE BYTE STACK=0008H					265					
112			CHKSIZ	LITERALLY										
112	000BH	2	CKS.	WORD MEMBER(DPB)										
153	004FH	27	CMD.	BYTE ARRAY(27) INITIAL				322	345	346	402	430	432	520	567
				763	765	798	799	819	821	827	829	833	837	839	841	843	845	

```
                                        847  851  857  859  892
135  0020H     1  CNS. . . . . .       BYTE MEMBER(PDPARENT)
135  0020H     1  CNS. . . . . .       BYTE MEMBER(PD)
281  0129H     1  CODE . . . . . .     BYTE          283  287  289  297  299  301
399  012EH     1  CODE . . . . . .     BYTE          401  403  408
  9  00E0H     3  COMMA. . . . . .     BYTE ARRAY(3) DATA
317  0000H     1  COMMANDS . . . . .   BYTE BASED(COMMANDSADR) ARRAY(1)          322
315  0006H     2  COMMANDSADR. . . .   WORD PARAMETER AUTOMATIC          317  322
 64  0086H    15  CONIN. . . . . .     PROCEDURE BYTE STACK=0008H               237
135  0022H     2  CONMODE. . . . . .   WORD MEMBER(PD)
135  0022H     2  CONMODE. . . . . .   WORD MEMBER(PDPARENT)
226  0373H    37  COPY . . . . . .     PROCEDURE STACK=0008H          471  480  520  567  798  799
  5  0020H    38  COPYRIGHT. . . . .   BYTE ARRAY(38) DATA
  8             CR . . . . . .        LITERALLY          53  164  169  249  614
592  080FH    67  CREATE . . . . . .   PROCEDURE STACK=0030H          828
 52  0056H    17  CRLF . . . . . .     PROCEDURE STACK=000EH           58  147  490  619  634  637  666  685
                                        687  717  725  745  747
 82  00D7H    15  CSELECT. . . . . .   PROCEDURE BYTE STACK=0008H               896
  8             CTRLC. . . . . .      LITERALLY          262
  8             CTRLH. . . . . .      LITERALLY          253
  8             CTRLX. . . . . .      LITERALLY          251
500  0000H     1  D. . . . . .         BYTE BASED(Q)           516
226  0006H     2  D. . . . . .         WORD PARAMETER AUTOMATIC         227  228  230  232
 67  0004H     1  D. . . . . .         BYTE PARAMETER AUTOMATIC          68   69
445  02AFH     1  D. . . . . .         BYTE DATA          447
121  0004H     1  D. . . . . .         BYTE PARAMETER AUTOMATIC         122  123
112  0009H     2  DBL. . . . . .       WORD MEMBER(DPB)
136  000EH     2  DBLK . . . . . .     WORD MEMBER(UDAPARENT)
136  000EH     2  DBLK . . . . . .     WORD MEMBER(UDA)
  8             DCL. . . . . .        LITERALLY
136  000CH     2  DCNT . . . . . .     WORD MEMBER(UDA)          308  775
443  0130H     1  DCNT . . . . . .     BYTE          447  448  449  451  463
136  000CH     2  DCNT . . . . . .     WORD MEMBER(UDAPARENT)
424  07ACH   130  DEFAULTPASS. . . .   PROCEDURE STACK=0028H          834
  9  00FEH     7  DELETEMODE . . . .   BYTE ARRAY(7) DATA        662
136  0012H     8  DFPASSWORD . . . .   BYTE ARRAY(8) MEMBER(UDA)        440
136  0012H     8  DFPASSWORD . . . .   BYTE ARRAY(8) MEMBER(UDAPARENT)           440
112             DIRBLK . . . . . .    LITERALLY
156  0088H   128  DIRBUF . . . . . .   BYTE ARRAY(128)       311  382  446  449  463  772
304  0004H     1  DIRINDEX . . . . .   BYTE PARAMETER AUTOMATIC          305  306  311
771  0134H     1  DIRINDEX . . . . .   BYTE          778  780  781  783
  9  0093H    17  DIRLABEL . . . . .   BYTE ARRAY(17) DATA        707
112             DIRMAX . . . . . .    LITERALLY
 85  0004H     2  DMA. . . . . .       WORD PARAMETER AUTOMATIC          86   87
136  0002H     2  DMAOFST. . . . . .   WORD MEMBER(UDAPARENT)
136  0002H     2  DMAOFST. . . . . .   WORD MEMBER(UDA)
136  0004H     2  DMASEG . . . . . .   WORD MEMBER(UDAPARENT)
136  0004H     2  DMASEG . . . . . .   WORD MEMBER(UDA)
112  0007H     2  DMX. . . . . .       WORD MEMBER(DPB)
136  0000H     2  DPARAM . . . . . .   WORD MEMBER(UDA)
136  0000H     2  DPARAM . . . . . .   WORD MEMBER(UDAPARENT)
112  0000H    15  DPB. . . . . .       STRUCTURE BASED(DPBPTR)          698
112  0000H     4  DPBPTR . . . . . .   POINTER          112  114  698
142  0004H     1  DRV. . . . . .       BYTE PARAMETER AUTOMATIC         143  144
135  0012H     1  DSK. . . . . .       BYTE MEMBER(PDPARENT)
135  0012H     1  DSK. . . . . .       BYTE MEMBER(PD)
135  0018H     2  DVRACT . . . . . .   WORD MEMBER(PD)
```

```
135  0018H     2  DVRACT . . . . . .  WORD MEMBER(PDPARENT)
706  0D66H    15  ERR. . . . . .      PROCEDURE STACK=001EH                714  722
  9  0084H    12  ERRATTRIB. . . . .  BYTE ARRAY(12) DATA          329
  9  01D2H    48  ERRCRAC. . . . . .  BYTE ARRAY(48) DATA          576  599
  9  0198H    55  ERRENAB. . . . . .  BYTE ARRAY(55) DATA          427  503  536
  9  023CH    19  ERRFORM2 . . . . .  BYTE ARRAY(19) DATA          730
  9  0202H    58  ERRFORMAT. . . . .  BYTE ARRAY(58) DATA          729
  9  024FH    34  ERRNOFILE. . . . .  BYTE ARRAY(34) DATA          674
  9  017BH    32  ERRNOPASS. . . . .  BYTE ARRAY(32) DATA          647
158  0020H     2  ERRORCODE. . . . .  WORD        282  677  678   679  683  688  689  697  710  711  712  715
                                       719  720  738  739  740   743  749  750  758
136  0010H     1  ERRORMODE. . . . .  BYTE MEMBER(UDAPARENT)
136  0010H     1  ERRORMODE. . . . .  BYTE MEMBER(UDA)
  9  00E8H     8  ERRORMSG . . . . .  BYTE ARRAY(8) DATA          179  284  296  491  646  868
  9  015BH    32  ERRRDLBL . . . . .  BYTE ARRAY(32) DATA          453
265  0431H        EXIT . . . . . .    LABEL       250
112  0004H     1  EXM. . . . . .      BYTE MEMBER(DPB)           698
112             EXTMSK . . . . . .   LITERALLY       698
219  0006H     1  F. . . . . .        BYTE PARAMETER AUTOMATIC          220  222
 28  0000H     1  F. . . . . .        BYTE PARAMETER         29
 25  0000H     1  F. . . . . .        BYTE PARAMETER         26
 22  0000H     1  F. . . . . .        BYTE PARAMETER         23
 19  0000H     1  F. . . . . .        BYTE PARAMETER         20
  9  011EH    25  FAIL . . . . . .    BYTE ARRAY(25) DATA          405
  9  0137H    31  FAILED . . . . . .  BYTE ARRAY(31) DATA          421
  8             FALSE. . . . . .     LITERALLY       154  204  211   323  334  353  459  542  734  760  786
                                       794  800  816  873  907  922
 99  0004H     2  FCB. . . . . .      WORD PARAMETER AUTOMATIC          100  101
130  0004H     2  FCB. . . . . .      WORD PARAMETER AUTOMATIC          131  133
116  0004H     2  FCB. . . . . .      WORD PARAMETER AUTOMATIC          117  119
107  0004H     2  FCB. . . . . .      WORD PARAMETER AUTOMATIC          108  110
 75  0004H     2  FCB. . . . . .      WORD PARAMETER AUTOMATIC           76   77
 71  0004H     2  FCB. . . . . .      WORD PARAMETER AUTOMATIC           72   73
 17  0000H    33  FCB. . . . . .      BYTE ARRAY(33) EXTERNAL(1)              312  383  470  471  780  791  798
                                       801  814
125  0004H     2  FCB. . . . . .      WORD PARAMETER AUTOMATIC          126  128
 17             FCBA . . . . . .     LITERALLY
159  0002H     2  FCBADR . . . . . .  WORD MEMBER(PARSEFN)       892
151  0014H     2  FCBP . . . . . .    WORD        151  277  311   312  351  395  396  463  464  465  466  470
                                       480  513  555  556  567   572  580  581  589  590  595  603  604  622
                                       625  628  631  635  677   688  698  710  719  814
151  0000H    32  FCBV . . . . . .    BYTE BASED(FCBP) ARRAY(32)              277  312  351  395  396  464  465
                                       466  480  513  555  556   567  572  580  581  589  590  595  603  604
                                       622  625  628  631  635   698
150             FDL. . . . . .       LITERALLY
150             FDM. . . . . .       LITERALLY
151             FEXT . . . . . .     LITERALLY       464  513  555   556  572  580  581  589  590  595  603
                                       604  622  625  628  631   635  698
154  007CH     1  FILEREF. . . . . .  BYTE INITIAL          506  532   560  672  803  816  823  831  853  857
219  0353H    32  FILL . . . . . .    PROCEDURE STACK=0008H            245  430  517  528
135  0006H     2  FLAG . . . . . .    WORD MEMBER(PD)
135  0006H     2  FLAG . . . . . .    WORD MEMBER(PDPARENT)
150             FLN. . . . . .       LITERALLY
150             FMOD . . . . . .     LITERALLY
150             FNAM . . . . . .     LITERALLY       274  790
  8             FOREVER. . . . . .   LITERALLY
150             FRC. . . . . .       LITERALLY
```

```
150                FTYP . . . . . LITERALLY        275  801
136  0006H     1   FUNC . . . . . BYTE MEMBER(UDA)
136  0006H     1   FUNC . . . . . BYTE MEMBER(UDAPARENT)
559  0A7AH    21   GETBOOL. . . . . PROCEDURE STACK=002EH            566  570  587  593
770  00FFH    79   GETFILE. . . . . PROCEDURE BYTE STACK=000EH            805  917
 99  0127H    16   GETFILESIZE. . . PROCEDURE STACK=000AH
121  018FH    19   GETLBL . . . . . PROCEDURE BYTE STACK=000AH            425  458  501  534
242  0388H   129   GETPASSWD. . . . PROCEDURE STACK=001AH            474  686  718  746
137  01D0H    40   GETPDUDA . . . . PROCEDURE STACK=0008H            435  894
 92  0105H    15   GETUSER. . . . . PROCEDURE BYTE STACK=0008H            893
 61  0077H    15   GETVERSION . . . PROCEDURE WORD STACK=0008H            878
162  0221H    96   HELP . . . . . PROCEDURE STACK=001AH            840
                   HIGH . . . . . BUILTIN          282  679  683  689  712  715  720  740  743  750  879
316  012AH     1   I. . . . . BYTE        321  322
789  0135H     1   I. . . . . BYTE        790  791
349  0004H     1   I. . . . . BYTE PARAMETER AUTOMATIC        350  351
243  0126H     1   I. . . . . BYTE        246  248  255  258
876  0136H     1   I. . . . . BYTE INITIAL         882  883  885  888
155  0018H     2   IBP. . . . . WORD        180  215  217  431  519  854  897  925
150                INFILE . . . . . LITERALLY        361
  9  007DH     9   INVALID. . . . . BYTE ARRAY(9) DATA         184
863  0FABH    52   ISTHEREMORE. . . PROCEDURE STACK=001AH            899  924
316  012BH     1   J. . . . . BYTE        318  322  324
272  0128H     1   K. . . . . BYTE        274  275  277
614  0280H   160   LABEL1 . . . . . BYTE ARRAY(160) DATA       620
  9  0156H     5   LABELNAME. . . . BYTE ARRAY(5) DATA         471
159  0026H     2   LASTBUFFADR. . . WORD        182  208  891
315  0004H     1   LASTCMD. . . . . BYTE PARAMETER AUTOMATIC        316  319
876  0137H     1   LASTDSEGBYTE . . BYTE INITIAL
154  007DH     1   LBLCMD . . . . . BYTE INITIAL         456  475  734  909
135  0014H     1   LDSK . . . . . BYTE MEMBER(PD)
135  0014H     1   LDSK . . . . . BYTE MEMBER(PDPARENT)
                   LENGTH . . . . . BUILTIN          471
  8                LF . . . . . LITERALLY         54  164  169  614
135  0000H     2   LINK . . . . . WORD MEMBER(PD)
135  0000H     2   LINK . . . . . WORD MEMBER(PDPARENT)
  8                LIT. . . . . LITERALLY         13   14   15   16
565  0A8FH    26   LNAME. . . . . PROCEDURE STACK=0030H            842
                   LOW. . . . . BUILTIN          678  697  711  739  758  879
135  0024H     1   LST. . . . . BYTE MEMBER(PD)
135  0024H     1   LST. . . . . BYTE MEMBER(PDPARENT)
135  0015H     1   LUSER. . . . . BYTE MEMBER(PD)
135  0015H     1   LUSER. . . . . BYTE MEMBER(PDPARENT)
616  0132H     1   MAKE . . . . . BYTE
315  0539H   106   MATCH. . . . . PROCEDURE BYTE STACK=0024H            332  388  400  543
316  012CH     1   MATCHED. . . . . BYTE        320  323  326
112                MAXALL . . . . . LITERALLY
 17  0000H     2   MAXB . . . . . WORD EXTERNAL(0)
135  0016H     2   MEM. . . . . WORD MEMBER(PD)
135  0016H     2   MEM. . . . . WORD MEMBER(PDPARENT)
103  0004H     1   MODE . . . . . BYTE PARAMETER AUTOMATIC        104  105
443  012FH     1   MODE . . . . . BYTE        458  460
 19  0000H         MON1 . . . . . PROCEDURE EXTERNAL(3) STACK=0000H          32   36   69   87   97  101
                     105  128  148  432
 22  0000H         MON2 . . . . . PROCEDURE BYTE EXTERNAL(4) STACK=0000H          50   65   73   77   80
                      83   90   93  123  144
 25  0000H         MON3 . . . . . PROCEDURE WORD EXTERNAL(5) STACK=0000H          62  110  119  133  196
```

```
 28  0000H        MON4 . . . . . .  PROCEDURE POINTER EXTERNAL(6) STACK=0000H          114  138
155  0086H     1  MORE . . . . . .  BYTE INITIAL          373  893  902
                 MOVW . . . . . .  BUILTIN         440
177  0004H     2  MSG. . . . . .  WORD PARAMETER AUTOMATIC        178  185
488  0004H     2  MSG. . . . . .  WORD PARAMETER AUTOMATIC        469  494
136  0011H     1  MULTCNT. . . . .  BYTE MEMBER(UDAPARENT)
136  0011H     1  MULTCNT. . . . .  BYTE MEMBER(UDA)
112  0005H     2  MXA. . . . . .  WORD MEMBER(DPB)
154  0085H     1  MXSTAMP. . . . .  BYTE INITIAL          574  582  597  605
135  0008H     8  NAME . . . . . .  BYTE ARRAY(8) MEMBER(PD)
135  0008H     8  NAME . . . . . .  BYTE ARRAY(8) MEMBER(PDPARENT)
135  001DH     1  NET. . . . . .  BYTE MEMBER(PD)
135  001DH     1  NET. . . . . .  BYTE MEMBER(PDPARENT)
525  0131H     1  NEWPASSWORD. . . .  BYTE        541  542  549
488  0923H    36  NOLABEL. . . . .  PROCEDURE STACK=001CH
  9  0105H     5  NOPASSWD . . . . .  BYTE ARRAY(5) DATA        652
  9  0066H    23  NOSPACE. . . . .  BYTE ARRAY(23) DATA        756
  9  0056H    16  NOTFOUND . . . . .  BYTE ARRAY(16) DATA       696  755  811
247  03D8H        NXTCHR . . . . .  LABEL       259
135  0000H     2  OFF. . . . . .  WORD MEMBER(PDPTR)
  9  0114H    10  OFF. . . . . .  BYTE ARRAY(10) DATA        624  627  630  633
136  0000H     2  OFF. . . . . .  WORD MEMBER(UDAPARENTPTR)       439
136  0000H     2  OFF. . . . . .  WORD MEMBER(UDAPTR)        139
135  0000H     2  OFF. . . . . .  WORD MEMBER(PDPARENTPTR)       437
112                OFFSET . . . . .  LITERALLY
112  000DH     2  OFS. . . . . .  WORD MEMBER(DPB)
  9  010AH    10  ON . . . . . .  BYTE ARRAY(10) DATA        623  626  629  632
 71  00A8H    16  OPEN . . . . . .  PROCEDURE BYTE STACK=000AH
155  0016H     2  OPTADR . . . . .  WORD        808  919
154  0080H     1  OPTDEL . . . . .  BYTE INITIAL          202  204  211  887  904  920
154  0084H     1  OPTION . . . . .  BYTE INITIAL          189  887  900  905  907  920
154  0081H     1  OPTIONFOUND. . . .  BYTE INITIAL          800  866  887  905
  9  00A4H    16  OPTIONSET. . . . .  BYTE ARRAY(16) DATA       338
135  001CH     1  ORG. . . . . .  BYTE MEMBER(PD)
135  001CH     1  ORG. . . . . .  BYTE MEMBER(PDPARENT)
194  0028H     2  P. . . . . .  WORD        195  196  197  199  201  203  205  206  207  209  212
498  002AH     2  P. . . . . .  WORD        499  515  516
135  001EH     2  PARENT . . . . .  WORD MEMBER(PD)       437
135  001EH     2  PARENT . . . . .  WORD MEMBER(PDPARENT)
193  02D2H   108  PARSE. . . . . .  PROCEDURE WORD STACK=0024H          217  431  519  897  925
188  02BAH    24  PARSEERROR . . . .  PROCEDURE STACK=0020H          198  216  561  825  835  849  861
159  0022H     4  PARSEFN. . . . .  STRUCTURE        181  196  208  209  515  808  888  890  892  919
818  0EEDH   190  PARSEOPTION. . . .  PROCEDURE STACK=0034H          901
214  033EH    21  PARSEVALUE . . . .  PROCEDURE STACK=0028H          390  531  563
153  006AH    17  PASSWD . . . . . .  BYTE ARRAY(17)        109  118  127  132  245  248  258  517  520  528
                                       611  612  799
154  0083H     1  PASSWORD . . . . .  BYTE INITIAL          459  467  473  481  484  521  529  644
336  05BCH    33  PBOOL. . . . . .  PROCEDURE STACK=0016H
 15                PCM11. . . . . .  LITERALLY
 15                PCMCTLC. . . . .  LITERALLY
 15                PCMCTLO. . . . .  LITERALLY
 15                PCMCTLS. . . . .  LITERALLY
 15                PCMROUT. . . . .  LITERALLY
 15                PCMRSX . . . . .  LITERALLY
135  0000H    48  PD . . . . . .  STRUCTURE BASED(PDPOINTER)          140  437
136  001AH     1  PDCNT. . . . . .  BYTE MEMBER(UDAPARENT)
136  001AH     1  PDCNT. . . . . .  BYTE MEMBER(UDA)
```

```
 11                 PDHDR. . . . . LITERALLY        135
135  0000H    48    PDPARENT . . . . STRUCTURE BASED(PDPARENTPTR)          438
135  0008H     4    PDPARENTPOINTER. . POINTER         135  438
135  0008H     4    PDPARENTPTR. . . . STRUCTURE AT         436  437
135  0004H     4    PDPOINTER. . . . . POINTER         135  138  140  437
135  0004H     4    PDPTR. . . . . . . STRUCTURE AT         436
 12                 PDSTRUCTURE. . . . LITERALLY       135
177  0281H    57    PERROR . . . . . . PROCEDURE STACK=001CH          190  191  329
 14                 PFACTIVE . . . . . LITERALLY
 14                 PFCHILDABORT . . . LITERALLY
 14                 PFCTLC . . . . . . LITERALLY
 14                 PFCTLD . . . . . . LITERALLY
 14                 PFKEEP . . . . . . LITERALLY
 14                 PFKERNAL . . . . . LITERALLY
 14                 PFNOCTLS . . . . . LITERALLY
 14                 PFPURE . . . . . . LITERALLY
 14                 PFRAW. . . . . . . LITERALLY
 14                 PFRESOURCE . . . . LITERALLY
 14                 PFSYS. . . . . . . LITERALLY
 14                 PFTABLE. . . . . . LITERALLY
 14                 PFTEMPKEEP . . . . LITERALLY
877  0FDFH   274    PLMSTART . . . . . PROCEDURE PUBLIC STACK=0038H
 10                 PNAMSIZ. . . . . . LITERALLY
135  002CH     2    PRET . . . . . . . WORD MEMBER(PDPARENT)
135  002CH     2    PRET . . . . . . . WORD MEMBER(PD)
 56  0067H    16    PRINT. . . . . . . PROCEDURE STACK=0016H          163  164  165  166  167  168  169  170
                                        171  172  173  174  175  179  184  244  284  405  410  421  427  433
                                        453  491  503  536  576  599  617  620  674  707  729  730  811  868
                                        880
348  05F5H   121    PRINTATT . . . . . PROCEDURE STACK=0016H          701
 38  0023H    11    PRINTB . . . . . . PROCEDURE STACK=000EH          286  412  668  669
 34  0010H    19    PRINTCHAR. . . . . PROCEDURE STACK=000AH           39   45   53   54  268  269  276  277
                                        358  371
343  050DH    24    PRINTCMD . . . . . PROCEDURE STACK=0014H          434
267  0439H    20    PRINTDRV . . . . . PROCEDURE STACK=000EH          273  285  411  493  618
271  0440H    58    PRINTFN. . . . . . PROCEDURE STACK=0012H          621  667
 41  002EH    25    PRINTX . . . . . . PROCEDURE STACK=0010H           59  181  182  183  185  288  291  292
                                        296  298  300  302  338  340  341  344  346  360  362  363  364  367
                                        368  370  413  416  417  419  492  494  610  612  623  624  626  627
                                        629  630  632  633  646  647  651  652  656  658  660  662  696  755
                                        756  869
135  0005H     1    PRIOR. . . . . . . BYTE MEMBER(PD)
135  0005H     1    PRIOR. . . . . . . BYTE MEMBER(PDPARENT)
355  068EH    23    PRNTATTRIB . . . . PROCEDURE STACK=0012H          372  373  374  375  376
  8                 PROC . . . . . . . LITERALLY       177  188  214  219  226  235  242  315  444  526  706
                                        863
524  09C1H   157    PROTECT. . . . . . PROCEDURE STACK=002CH          846
 13                 PSCIOWAIT. . . . . LITERALLY
 13                 PSDELAY. . . . . . LITERALLY
 13                 PSDQ . . . . . . . LITERALLY
 13                 PSFLAGWAIT . . . . LITERALLY
 13                 PSNQ . . . . . . . LITERALLY
 13                 PSPOLL . . . . . . LITERALLY
 13                 PSRUN. . . . . . . LITERALLY
 13                 PSSLEEP. . . . . . LITERALLY
 13                 PSSWAP . . . . . . LITERALLY
 13                 PSTERM . . . . . . LITERALLY
```

```
671  0C6FH  143  PUTATTRIBUTES. . .  PROCEDURE STACK=001FH                912
665  0C5EH   17  PUTFILE. . . . . .  PROCEDURE STACK=0016H                681  691  760  737  748
498  002CH    2  Q. . . . . . . . .  WORD          500  515  516
378  06A5H   39  RDATTRIBUTES . . .  PROCEDURE STACK=000EH                387
442  082EH  114  READLABEL. . . . .  PROCEDURE STACK=001EH                512  553  562
444  08A0H   66  READLBL. . . . . .  PROCEDURE STACK=001AH                462
  9  00F3H    5  READMODE . . . . .  BYTE ARRAY(5) DATA          658
  9  00C0H   10  READONLY . . . . .  BYTE ARRAY(10) DATA         292  298  367  416
  9  00D0H   16  READWRITE. . . . .  BYTE ARRAY(16) DATA         370  419
125  01A2H   23  READXFCB . . . . .  PROCEDURE STACK=0010H                482
135  0028H    4  RESERVD. . . . . .  BYTE ARRAY(4) MEMBER(PD)
135  0028H    4  RESERVD. . . . . .  BYTE ARRAY(4) MEMBER(PDPARENT)
142  01F8H   24  RESETDRV . . . . .  PROCEDURE BYTE STACK=000AH           403
 18  0000H   32  RESETMASK. . . . .  WORD ARRAY(16) DATA         144
245  03C2H       RETRY. . . . . .  LABEL         252  256
103  0137H   19  RETURNERRORS . . .  PROCEDURE STACK=000AH                908  915
  9  00CAH    6  RO . . . . . . . .  BYTE ARRAY(6) DATA          368  417
150             ROFILE . . . . . .  LITERALLY        365
                ROL. . . . . . . .  BUILTIN          351
                ROR. . . . . . . .  BUILTIN          449
226  0008H    2  S. . . . . . . . .  WORD PARAMETER AUTOMATIC          227  228  230  231
219  0008H    2  S. . . . . . . . .  WORD PARAMETER AUTOMATIC          220  222  223
 43  0000H    1  S. . . . . . . . .  BYTE BASED(A)          44   45
156  001AH    2  SAVDCNT. . . . . .  WORD          308  775
156  001CH    2  SAVSEARCHA . . . .  WORD          310  777
156  0087H    1  SAVSEARCHL . . . .  BYTE          309  776
316  012DH    1  SCASE. . . . . .  BYTE          319  327
154  007BH    1  SCASE. . . . . .  BYTE INITIAL          379  388  392  394  395  396  400  414  422  703
                                     911
112             SCPTRK . . . . . .  LITERALLY
135  002EH    2  SCRATCH. . . . . .  WORD MEMBER(PDPARENT)
135  002EH    2  SCRATCH. . . . . .  WORD MEMBER(PD)
136  0008H    2  SEARCHA. . . . . .  WORD MEMBER(CDAPARENT)
136  0008H    2  SEARCHA. . . . . .  WORD MEMBER(CDA)          310  777
136  000AH    2  SEARCHABASE. . . .  WORD MEMBER(CDAPARENT)
136  000AH    2  SEARCHABASE. . . .  WORD MEMBER(CDA)
 75  0088H   16  SEARCHFIRST. . . .  PROCEDURE BYTE STACK=000AH           383  447  780
136  0007H    1  SEARCHL. . . . . .  BYTE MEMBER(CDAPARENT)
136  0007H    1  SEARCHL. . . . . .  BYTE MEMBER(CDA)          309  776
 79  00C8H   15  SEARCHNEXT . . . .  PROCEDURE BYTE STACK=0008H           451  778
 17             SECTORLEN. . . . .  LITERALLY
135  0002H    2  SEGMENT. . . . . .  WORD MEMBER(PDPTR)          436
136  0002H    2  SEGMENT. . . . . .  WORD MEMBER(CDAPARENTPTR)        438
136  0002H    2  SEGMENT. . . . . .  WORD MEMBER(CDAPTR)          140
135  0002H    2  SEGMENT. . . . . .  WORD MEMBER(PDPARENTPTR)        436
 67  0095H   19  SELECT . . . . . .  PROCEDURE STACK=000AH                766
  1  0002H       SET. . . . . . . .  PROCEDURE STACK=0000H
386  06CCH   94  SETATTRIBUTES. . .  PROCEDURE STACK=002CH                824  832  838  854  858
762  0DE4H   27  SETDISK. . . . . .  PROCEDURE STACK=000EH                797
 85  00E6H   16  SETDMA . . . . . .  PROCEDURE STACK=000AH                109  118  127  132  382  446  772
113  0161H   23  SETDPB . . . . . .  PROCEDURE STACK=0008H                767  895
398  072AH  130  SETDRVSTATUS . . .  PROCEDURE STACK=0028H                855
107  014AH   23  SETIND . . . . . .  PROCEDURE WORD STACK=0010H           677  688
497  0947H  122  SETPASSWORD. . . .  PROCEDURE STACK=0028H                848
  9  0086H   13  SETPROT. . . . . .  BYTE ARRAY(13) DATA
  9  00E3H    8  SETTD. . . . . .  BYTE ARRAY(8) DATA          344  360  413
796  0E78H  117  SETUPFCB . . . . .  PROCEDURE STACK=001AH                903
```

```
304  04FCH    61  SETUPFILE. . . . .  PROCEDURE STACK=0004H                  363  783
477  08E2H    65  SETUPXFCB. . . . .  PROCEDURE STACK=0014H                  508  533
 95  0114H    19  SETUSER. . . . . .  PROCEDURE STACK=000AH
135  0025H     1  SF3. . . . . . . .  BYTE MEMBER(PD)
135  0025H     1  SF3. . . . . . . .  BYTE MEMBER(PDPARENT)
135  0026H     1  SF4. . . . . . . .  BYTE MEMBER(PD)
135  0026H     1  SF4. . . . . . . .  BYTE MEMBER(PDPARENT)
135  0027H     1  SF5. . . . . . . .  BYTE MEMBER(PD)
135  0027H     1  SF5. . . . . . . .  BYTE MEMBER(PDPARENT)
                  SHL. . . . . . . .  BUILTIN          311  463
615  0B6AH   148  SHOWLBL. . . . . .  PROCEDURE STACK=001AH                  733
609  0B52H    24  SHOWPASSWD . . . .  PROCEDURE STACK=0014H                  638
641  0BFEH    96  SHOWXFCB . . . . .  PROCEDURE STACK=0014H                  759
112  0000H     2  SPT. . . . . . . .  WORD MEMBER(DPB)
135  0004H     1  STAT . . . . . . .  BYTE MEMBER(PD)
135  0004H     1  STAT . . . . . . .  BYTE MEMBER(PDPARENT)
  8                 TAB. . . . . . . .  LITERALLY        163  164  167  168  169  170  171  172  173  174  175
                    371
146  0210H    17  TERMINATE. . . . .  PROCEDURE STACK=0012H                  186  263  294  406  428  454  495  504
                    537  577  600  675  723  731  812  870  928
135  0002H     2  THREAD . . . . . .  WORD MEMBER(PD)
135  0002H     2  THREAD . . . . . .  WORD MEMBER(PDPARENT)
154  0082H     1  TIMEOPT. . . . . .  BYTE INITIAL          800
  8                 TRUE . . . . . . .  LITERALLY        155  202  320  333  352  467  475  484  485  521  529
                    541  582  605  784  792  803  807  887  905  920
235  0398H    32  UCASE. . . . . . .  PROCEDURE BYTE STACK=000CH                247
135  0010H     2  UDA. . . . . . . .  WORD MEMBER(PD)        140
136  0000H    27  UDA. . . . . . . .  STRUCTURE BASED(UDAPOINTER)            308  309  310  440  775  776  777
135  0010H     2  UDA. . . . . . . .  WORD MEMBER(PDPARENT)              438
136  0000H    27  UDAPARENT. . . . .  STRUCTURE BASED(UDAPARENTPOINTER)          440
136  0010H     4  UDAPARENTPOINTER .  POINTER          136  440
136  0010H     4  UDAPARENTPTR . . .  STRUCTURE AT           438  439
136  000CH     4  UDAPOINTER . . . .  POINTER          136  308  309  310  440  775  776  777
136  000CH     4  UDAPTR . . . . . .  STRUCTURE AT           139  140
 16                 UDASTRUCTURE . . .  LITERALLY        136
586  0AECH    35  UPDATE . . . . . .  PROCEDURE STACK=0030H                  860
135  0013H     1  USER . . . . . . .  BYTE MEMBER(PD)
 95  0004H     1  USER . . . . . . .  BYTE PARAMETER AUTOMATIC            96   97
135  0013H     1  USER . . . . . . .  BYTE MEMBER(PDPARENT)
 17  002EH     1  USERCODE . . . . .  BYTE         893
336  0004H     1  VALUE. . . . . . .  BYTE PARAMETER AUTOMATIC            337  339
161  028DH     9  VALUES . . . . . .  BYTE ARRAY(9) DATA          543
157  001EH     2  VER. . . . . . . .  WORD         878  879
  4                 VERBDOS. . . . . .  LITERALLY        879
  3                 VERMASK. . . . . .  LITERALLY        879
  2                 VERNEEDSOS . . . .  LITERALLY        880
  2                 VEROS. . . . . . .  LITERALLY        879
  7  004EH     8  VERSION. . . . . .  BYTE ARRAY(8) DATA
  6  0046H     8  VERSIONDATE. . . .  BYTE ARRAY(8) DATA
135  001AH     2  WAIT . . . . . . .  WORD MEMBER(PD)
135  001AH     2  WAIT . . . . . . .  WORD MEMBER(PDPARENT)
154  007FH     1  WILD . . . . . . .  BYTE INITIAL          380  773  807  872  916  922
788  0E4EH    42  WILDCARD . . . . .  PROCEDURE BYTE STACK=0002H                804
705  0CFEH   104  WRITELABEL . . . .  PROCEDURE STACK=0022H                  910
  9  00F8H     6  WRITEMODE. . . . .  BYTE ARRAY(6) DATA          660
 89  00F6H    15  WRITEPROT. . . . .  PROCEDURE BYTE STACK=0008H                401
736  0D75H   111  WRITEXFCB. . . . .  PROCEDURE STACK=001EH                  914
```

```
  116  0178H    23  WRLBL. . . . . .   PROCEDURE WORD STACK=0010H          710  719
  130  0189H    23  WRXFCB . . . . . .  PROCEDURE WORD STACK=0010H          736  749
  152  002FH    32  XFCB . . . . . .   BYTE ARRAY(32)          152  480  482  738  749
  154  007EH     1  XFCBCMD. . . . . .  BYTE INITIAL           473  435  760  913
  152  0038H     1  XFCBMODE . . . . .  BYTE AT          481  483  509  527  540  544  545  546  550  642  657
                                        659  661
  526  0A5EH    28  ZEROPASS . . . . .  PROCEDURE STACK=006CH          547
```

MODULE INFORMATION:

```
     CODE AREA SIZE     = 10F1H    4337D
     CONSTANT AREA SIZE = 06C3H    1731D
     VARIABLE AREA SIZE = 0138H     312D
     MAXIMUM STACK SIZE = 0038H      56D
     1783 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

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

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

BASE	OFFSET	TYPE	SYMBOL	BASE	OFFSET	TYPE	SYMBOL
0000H	10DFH	PUB	PLMSTART	0000H	005DH	PUB	XDOS
0000H	005DH	PUB	MON1	0000H	005DH	PUB	MON2
0000H	005DH	PUB	MON3	0000H	005DH	PUB	MON4
1000H	0004H	PUB	RDISK	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	TBUFF
1000H	0080H	PUB	BUFFA	1000H	005CH	PUB	FCBA
1000H	0100H	PUB	SAVEAX	1000H	0102H	PUB	SAVEBX
1000H	0104H	PUB	SAVECX	1000H	0106H	PUB	SAVEDX

SET: SYMBOLS AND LINES

1000H	0960H	SYM	MEMORY	1000H	0288H	SYM	COPYRIGHT
1000H	02DEH	SYM	VERSIONDATE	1000H	02E6H	SYM	VERSION
1000H	02EEH	SYM	NOTFOUND	1000H	02FEH	SYM	NOSPACE
1000H	0315H	SYM	INVALID	1000H	031EH	SYM	SETPROT
1000H	032BH	SYM	DIRLABEL	1000H	033CH	SYM	OPTIONSET
1000H	034CH	SYM	ERRATTRIB	1000H	0358H	SYM	READONLY
1000H	0362H	SYM	RO	1000H	0368H	SYM	READWRITE
1000H	0378H	SYM	COMMA	1000H	0378H	SYM	SETTO
1000H	0383H	SYM	ERRORMSG	1000H	038BH	SYM	READMODE
1000H	0390H	SYM	WRITEMODE	1000H	0396H	SYM	DELETEMODE
1000H	039DH	SYM	NOPASSWD	1000H	03A2H	SYM	ON
1000H	03ACH	SYM	OFF	1000H	03B6H	SYM	FAIL
1000H	03CFH	SYM	FAILED	1000H	03EEH	SYM	LABELNAME
1000H	03F3H	SYM	ERRDLBL	1000H	0413H	SYM	ERRNOPASS
1000H	0433H	SYM	ERRENAB	1000H	046AH	SYM	ERRCRAC
1000H	049AH	SYM	ERRFORMAT	1000H	04D4H	SYM	ERRFORM2
1000H	04E7H	SYM	ERRNOFILE	1000H	0136H	SYM	USERCODE
1000H	0298H	SYM	RESETMASK	0000H	0102H	SYM	BOOT
0000H	0110H	SYM	PRINTCHAR	STACK	0004H	SYM	CHAR
0000H	0123H	SYM	PRINTB	0000H	012EH	SYM	PRINTX
STACK	0004H	SYM	A	STACK	0004H	BAS	S
0000H	0147H	SYM	CHECKCONSTAT	0000H	0156H	SYM	CRLF
0000H	0167H	SYM	PRINT	STACK	0004H	SYM	A
0000H	0177H	SYM	GETVERSION	0000H	0186H	SYM	CONIN
0000H	0195H	SYM	SELECT	STACK	0004H	SYM	D
0000H	01A8H	SYM	OPEN	STACK	0004H	SYM	FCB
0000H	01B8H	SYM	SEARCHFIRST	STACK	0004H	SYM	FCB
0000H	01C8H	SYM	SEARCHNEXT	0000H	01D7H	SYM	CSELECT
0000H	01E6H	SYM	SETDMA	STACK	0004H	SYM	DMA
0000H	01F6H	SYM	WRITEPROT	0000H	0205H	SYM	GETUSER
0000H	0214H	SYM	SETUSER	STACK	0004H	SYM	USER
0000H	0227H	SYM	GETFILESIZE	STACK	0004H	SYM	FCB
0000H	0237H	SYM	RETURNERRORS	STACK	0004H	SYM	MODE
0000H	024AH	SYM	SETIND	STACK	0004H	SYM	FCB
1000H	0108H	SYM	DPBPTR	1000H	0108H	BAS	DPB
0000H	0261H	SYM	SETDPB	0000H	0278H	SYM	WRLBL

STACK	0004H	SYM	FCB
STACK	0004H	SYM	D
STACK	0004H	SYM	FCB
STACK	0004H	SYM	FCB
1000H	010CH	SYM	PDPTR
1000H	0110H	SYM	PDPARENTPOINTER
1000H	0110H	BAS	PDPARENT
1000H	0114H	SYM	UDAPTR
1000H	0118H	SYM	UDAPARENTPOINTER
1000H	0118H	BAS	UDAPARENT
0000H	02F8H	SYM	RESETDRV
0000H	0310H	SYM	TERMINATE
1000H	011CH	BAS	FCBV
1000H	0143H	SYM	XFCBMODE
1000H	0172H	SYM	PASSWD
1000H	0184H	SYM	FILEREF
1000H	0186H	SYM	XFCBCMD
1000H	0188H	SYM	OPTDEL
1000H	018AH	SYM	TIMEOPT
1000H	018CH	SYM	OPTION
1000H	018EH	SYM	MORE
1000H	0120H	SYM	IBP
1000H	0124H	SYM	SAVSEARCHA
1000H	0190H	SYM	DIRBUF
1000H	0126H	SYM	VER
1000H	012AH	SYM	PARSEFN
1000H	0211H	SYM	ATTRBYTE
1000H	0509H	SYM	ATTRIBUTES
1000H	052EH	SYM	BOOLEAN
0000H	0381H	SYM	PERROR
0000H	03BAH	SYM	PARSEERROR
1000H	0130H	SYM	P
0000H	043EH	SYM	PARSEVALUE
STACK	0008H	SYM	S
STACK	0004H	SYM	C
0000H	0473H	SYM	COPY
STACK	0006H	SYM	D
STACK	0008H	BAS	A
0000H	0498H	SYM	UCASE
0000H	04B8H	SYM	GETPASSWD
1000H	022FH	SYM	C
0000H	04D8H	SYM	NXTCHR
0000H	0539H	SYM	PRINTDRV
1000H	0230H	SYM	K
1000H	0231H	SYM	CODE
STACK	0004H	SYM	DIRINDEX
STACK	0006H	SYM	COMMANDSADR
1000H	0232H	SYM	I
1000H	0234H	SYM	MATCHED
STACK	0006H	BAS	COMMANDS
0000H	06BCH	SYM	PBOOL
0000H	06DDH	SYM	PRINTCMD
0000H	076EH	SYM	ATTRIBUTE
0000H	078EH	SYM	PRNTATTRIB
STACK	0004H	SYM	C
0000H	07CCH	SYM	SETATTRIBUTES
1000H	0236H	SYM	CODE
0000H	092EH	SYM	READLABEL
1000H	0238H	SYM	DCNT
1000H	0547H	SYM	D
0000H	028FH	SYM	GETCTL
0000H	02A2H	SYM	RLBDXFCB
0000H	0269H	SYM	WRXFCB
1000H	010CH	SYM	PDPOINTER
1000H	010CH	BAS	PD
1000H	0110H	SYM	PDPARENTPTR
1000H	0114H	SYM	UDAPOINTER
1000H	0114H	BAS	UDA
1000H	0118H	SYM	UDAPARENTPTR
0000H	02D0H	SYM	GETPDUDA
STACK	0004H	SYM	DRV
1000H	011CH	SYM	FCBP
1000H	0137H	SYM	XFCB
1000H	0157H	SYM	CMD
1000H	0183H	SYM	SCASE
1000H	0185H	SYM	LBLCMD
1000H	0187H	SYM	WILD
1000H	0189H	SYM	OPTIONFOUND
1000H	018BH	SYM	PASSWORD
1000H	018DH	SYM	MXSTAMP
1000H	011EH	SYM	OPTADR
1000H	0122H	SYM	SAVDCNT
1000H	018FH	SYM	SAVSEARCHL
1000H	0210H	SYM	CDISK
1000H	0128H	SYM	ERRORCODE
1000H	012EH	SYM	LASTBUFFADR
1000H	021FH	SYM	ATTRVALUE
1000H	0525H	SYM	VALUES
0000H	0321H	SYM	HELP
STACK	0004H	SYM	MSG
0000H	0302H	SYM	PARSE
1000H	0130H	BAS	C
0000H	0453H	SYM	FILL
STACK	0006H	SYM	F
STACK	0008H	BAS	A
STACK	0008H	SYM	S
STACK	0004H	SYM	C
STACK	0006H	BAS	B
1000H	022DH	SYM	C
1000H	022EH	SYM	I
0000H	04C2H	SYM	RETRY
0000H	0531H	SYM	EXIT
0000H	0540H	SYM	PRINTFN
0000H	0587H	SYM	BDOSERROR
0000H	05FCH	SYM	SETUPFILE
0000H	0639H	SYM	MATCH
STACK	0004H	SYM	LASTCMD
1000H	0233H	SYM	J
1000H	0235H	SYM	SCASE
0000H	06A3H	SYM	BOOL
STACK	0004H	SYM	VALUE
0000H	06F5H	SYM	PRINTATT
STACK	0004H	SYM	I
STACK	0006H	SYM	B
0000H	07A5H	SYM	RDATTRIBUTES
0000H	082AH	SYM	SETDRVSTATUS
0000H	08ACH	SYM	DEFAULTPASS
1000H	0237H	SYM	MODE
0000H	094DH	SYM	READLBL
0000H	09E2H	SYM	SETUPXFCB

```
0000H  0A23H  SYM  NOLABEL                STACK  0004H  SYM  MSG
0000H  0A47H  SYM  SETPASSWORD            1000H  0132H  SYM  P
1000H  0134H  SYM  Q                      1000H  0132H  BAS  C
1000H  0134H  BAS  D                      0000H  0AC1H  SYM  PROTECT
1000H  0239H  SYM  NEWPASSWORD            0000H  0B5EH  SYM  ZERODPASS
0000H  0B7AH  SYM  GETDDOL                0000H  0B8FH  SYM  LNAME
0000H  0BA9H  SYM  ACCESS                 0000H  0BECH  SYM  UPDATE
0000H  0C0FH  SYM  CREATE                 0000H  0C52H  SYM  SHOWPASSWD
1000H  0548H  SYM  LABEL1                 0000H  0C6AH  SYM  SHOWLBL
1000H  023AH  SYM  MAKE                   1000H  023BH  SYM  ACCESS
0000H  0CFEH  SYM  SHOWXFCB               0000H  0D5EH  SYM  PUTFILE
0000H  0D6FH  SYM  PUTATTRIBUTES          0000H  0DFEH  SYM  WRITELABEL
0000H  0E66H  SYM  ERR                    0000H  0E75H  SYM  WRITEXFCB
0000H  0EE4H  SYM  SETDISK                0000H  0EFFH  SYM  GETFILE
1000H  023CH  SYM  DIRINDEX               0000H  0F4EH  SYM  MALOCARD
1000H  023DH  SYM  I                      0000H  0F78H  SYM  SETUPFCB
0000H  0FEDH  SYM  PARSEOPTION            0000H  10ABH  SYM  ISTHEREMORE
1000H  023EH  SYM  I                      1000H  023FH  SYM  LASTDSEGBYTE
0000H  100FH  SYM  PLMSTART               0000H  0102H  LIN    31
0000H  0105H  LIN    32                   0000H  010EH  LIN    33
0000H  0110H  LIN    34                   0000H  0113H  LIN    36
0000H  011FH  LIN    37                   0000H  0123H  LIN    38
0000H  0126H  LIN    39                   0000H  012CH  LIN    40
0000H  012EH  LIN    41                   0000H  0131H  LIN    44
0000H  0139H  LIN    45                   0000H  013EH  LIN    46
0000H  0141H  LIN    47                   0000H  0143H  LIN    48
0000H  0147H  LIN    49                   0000H  014AH  LIN    50
0000H  0156H  LIN    51                   0000H  0156H  LIN    52
0000H  0159H  LIN    53                   0000H  015FH  LIN    54
0000H  0165H  LIN    55                   0000H  0167H  LIN    56
0000H  016AH  LIN    58                   0000H  016DH  LIN    59
0000H  0173H  LIN    60                   0000H  0177H  LIN    61
0000H  017AH  LIN    62                   0000H  0186H  LIN    63
0000H  0186H  LIN    64                   0000H  0189H  LIN    65
0000H  0195H  LIN    66                   0000H  0195H  LIN    67
0000H  0198H  LIN    69                   0000H  01A4H  LIN    70
0000H  01A8H  LIN    71                   0000H  01ABH  LIN    73
0000H  01B8H  LIN    74                   0000H  01B8H  LIN    75
0000H  01BBH  LIN    77                   0000H  01C8H  LIN    78
0000H  01C8H  LIN    79                   0000H  01CBH  LIN    80
0000H  01D7H  LIN    81                   0000H  01D7H  LIN    82
0000H  01DAH  LIN    83                   0000H  01E6H  LIN    84
0000H  01E6H  LIN    85                   0000H  01E9H  LIN    87
0000H  01F2H  LIN    88                   0000H  01F6H  LIN    89
0000H  01F9H  LIN    90                   0000H  0205H  LIN    91
0000H  0205H  LIN    92                   0000H  0208H  LIN    93
0000H  0214H  LIN    94                   0000H  0214H  LIN    95
0000H  0217H  LIN    97                   0000H  0223H  LIN    98
0000H  0227H  LIN    99                   0000H  022AH  LIN   101
0000H  0233H  LIN   102                   0000H  0237H  LIN   103
0000H  023AH  LIN   105                   0000H  0246H  LIN   106
0000H  024AH  LIN   107                   0000H  024DH  LIN   109
0000H  0254H  LIN   110                   0000H  0261H  LIN   111
0000H  0261H  LIN   113                   0000H  0264H  LIN   114
0000H  0276H  LIN   115                   0000H  0278H  LIN   116
0000H  027BH  LIN   118                   0000H  0282H  LIN   119
0000H  028FH  LIN   120                   0000H  028FH  LIN   121
0000H  0292H  LIN   123                   0000H  02A2H  LIN   124
0000H  02A2H  LIN   125                   0000H  02A5H  LIN   127
0000H  02ACH  LIN   128                   0000H  02B5H  LIN   129
```

```
0000H  02B9H  LIN   130
0000H  02C3H  LIN   133
0000H  02D0H  LIN   137
0000H  02E5H  LIN   139
0000H  02F6H  LIN   141
0000H  02FBH  LIN   144
0000H  0310H  LIN   146
0000H  0316H  LIN   148
0000H  0321H  LIN   162
0000H  0326H  LIN   164
0000H  0339H  LIN   166
0000H  0347H  LIN   168
0000H  0355H  LIN   170
0000H  0363H  LIN   172
0000H  0371H  LIN   174
0000H  037FH  LIN   176
0000H  0384H  LIN   179
0000H  0392H  LIN   181
0000H  039FH  LIN   183
0000H  03ADH  LIN   185
0000H  03B6H  LIN   187
0000H  03BDH  LIN   189
0000H  03C9H  LIN   191
0000H  03D2H  LIN   193
0000H  03E2H  LIN   197
0000H  03EEH  LIN   199
0000H  03FEH  LIN   202
0000H  040EH  LIN   204
0000H  041AH  LIN   206
0000H  042BH  LIN   208
0000H  0432H  LIN   210
0000H  0439H  LIN   212
0000H  043EH  LIN   214
0000H  0448H  LIN   216
0000H  0451H  LIN   218
0000H  0456H  LIN   221
0000H  046AH  LIN   223
0000H  046FH  LIN   225
0000H  0476H  LIN   229
0000H  048CH  LIN   231
0000H  0492H  LIN   233
0000H  0498H  LIN   235
0000H  04A5H  LIN   238
0000H  04B3H  LIN   240
0000H  04B8H  LIN   242
0000H  04C2H  LIN   245
0000H  04DBH  LIN   247
0000H  04F2H  LIN   249
0000H  0500H  LIN   253
0000H  050EH  LIN   258
0000H  0521H  LIN   262
0000H  052BH  LIN   264
0000H  0537H  LIN   266
0000H  053CH  LIN   268
0000H  054BH  LIN   270
0000H  0550H  LIN   273
0000H  055FH  LIN   275
0000H  056CH  LIN   277
0000H  0585H  LIN   279
0000H  058AH  LIN   282
```

```
0000H  02BCH  LIN   132
0000H  02D0H  LIN   134
0000H  02D3H  LIN   138
0000H  02E8H  LIN   140
0000H  02F8H  LIN   142
0000H  0310H  LIN   145
0000H  0313H  LIN   147
0000H  031FH  LIN   149
0000H  0324H  LIN   163
0000H  0332H  LIN   165
0000H  0340H  LIN   167
0000H  034EH  LIN   169
0000H  035CH  LIN   171
0000H  036AH  LIN   173
0000H  0378H  LIN   175
0000H  0381H  LIN   177
0000H  038BH  LIN   180
0000H  0398H  LIN   182
0000H  03A6H  LIN   184
0000H  03B3H  LIN   186
0000H  03BAH  LIN   188
0000H  03C4H  LIN   190
0000H  03D0H  LIN   192
0000H  03D5H  LIN   196
0000H  03E9H  LIN   198
0000H  03F5H  LIN   201
0000H  0405H  LIN   203
0000H  0413H  LIN   205
0000H  0421H  LIN   207
0000H  042CH  LIN   209
0000H  0434H  LIN   211
0000H  043EH  LIN   213
0000H  0441H  LIN   215
0000H  044BH  LIN   217
0000H  0453H  LIN   219
0000H  0462H  LIN   222
0000H  046DH  LIN   224
0000H  0473H  LIN   226
0000H  0482H  LIN   230
0000H  048FH  LIN   232
0000H  0494H  LIN   234
0000H  049BH  LIN   237
0000H  04ACH  LIN   239
0000H  04BBH  LIN   241
0000H  04BBH  LIN   244
0000H  04CFH  LIN   246
0000H  04E5H  LIN   248
0000H  04F9H  LIN   251
0000H  0507H  LIN   255
0000H  051FH  LIN   259
0000H  0528H  LIN   263
0000H  0531H  LIN   265
0000H  0539H  LIN   267
0000H  0545H  LIN   269
0000H  054DH  LIN   271
0000H  0553H  LIN   274
0000H  0566H  LIN   276
0000H  057FH  LIN   278
0000H  0587H  LIN   280
0000H  0596H  LIN   284
```

```
0000H  059DH  LIN  285
0000H  05A3H  LIN  287
0000H  05B1H  LIN  289
0000H  05BFH  LIN  292
0000H  05C9H  LIN  296
0000H  05D7H  LIN  298
0000H  05E5H  LIN  300
0000H  05F3H  LIN  302
0000H  05FCH  LIN  304
0000H  0605H  LIN  308
0000H  0617H  LIN  310
0000H  0620H  LIN  312
0000H  0639H  LIN  315
0000H  0641H  LIN  319
0000H  0653H  LIN  321
0000H  0677H  LIN  323
0000H  0680H  LIN  325
0000H  068DH  LIN  327
0000H  0698H  LIN  329
0000H  06A3H  LIN  331
0000H  06B4H  LIN  333
0000H  06BCH  LIN  336
0000H  06C6H  LIN  339
0000H  06D2H  LIN  341
0000H  06DDH  LIN  343
0000H  06E7H  LIN  345
0000H  06F3H  LIN  347
0000H  076EH  LIN  349
0000H  0784H  LIN  352
0000H  078EH  LIN  354
0000H  0791H  LIN  357
0000H  07A1H  LIN  359
0000H  06FFH  LIN  361
0000H  070EH  LIN  363
0000H  071CH  LIN  365
0000H  072DH  LIN  368
0000H  0732H  LIN  370
0000H  073FH  LIN  372
0000H  0751H  LIN  374
0000H  0763H  LIN  376
0000H  07A5H  LIN  378
0000H  07AFH  LIN  380
0000H  07BFH  LIN  383
0000H  07CCH  LIN  386
0000H  07D2H  LIN  388
0000H  07E6H  LIN  391
0000H  07F4H  LIN  394
0000H  0813H  LIN  396
0000H  082AH  LIN  398
0000H  083EH  LIN  401
0000H  0854H  LIN  403
0000H  0864H  LIN  406
0000H  086EH  LIN  410
0000H  0878H  LIN  412
0000H  0882H  LIN  414
0000H  0890H  LIN  417
0000H  0899H  LIN  419
0000H  089EH  LIN  421
0000H  08AAH  LIN  423
0000H  08AFH  LIN  425

0000H  05A0H  LIN  286
0000H  05AAH  LIN  288
0000H  05B8H  LIN  291
0000H  05C6H  LIN  294
0000H  05D0H  LIN  297
0000H  05DEH  LIN  299
0000H  05ECH  LIN  301
0000H  05FAH  LIN  303
0000H  05FFH  LIN  306
0000H  0610H  LIN  309
0000H  061EH  LIN  311
0000H  0635H  LIN  314
0000H  063CH  LIN  318
0000H  064EH  LIN  320
0000H  065FH  LIN  322
0000H  067CH  LIN  324
0000H  0686H  LIN  326
0000H  0692H  LIN  328
0000H  069FH  LIN  330
0000H  06A6H  LIN  332
0000H  06B8H  LIN  334
0000H  06BFH  LIN  338
0000H  06CDH  LIN  340
0000H  06D9H  LIN  342
0000H  06E0H  LIN  344
0000H  06ECH  LIN  346
0000H  06F5H  LIN  348
0000H  0771H  LIN  351
0000H  0788H  LIN  353
0000H  078EH  LIN  355
0000H  079BH  LIN  358
0000H  06F8H  LIN  360
0000H  0709H  LIN  362
0000H  0715H  LIN  364
0000H  0726H  LIN  367
0000H  0732H  LIN  369
0000H  0739H  LIN  371
0000H  0748H  LIN  373
0000H  075AH  LIN  375
0000H  076CH  LIN  377
0000H  07A8H  LIN  379
0000H  07B8H  LIN  382
0000H  07CAH  LIN  385
0000H  07CFH  LIN  387
0000H  07E3H  LIN  390
0000H  07EFH  LIN  392
0000H  0802H  LIN  395
0000H  0828H  LIN  397
0000H  0820H  LIN  400
0000H  0846H  LIN  402
0000H  085DH  LIN  405
0000H  0867H  LIN  408
0000H  0875H  LIN  411
0000H  087BH  LIN  413
0000H  0889H  LIN  416
0000H  0897H  LIN  418
0000H  089EH  LIN  420
0000H  08A5H  LIN  422
0000H  08ACH  LIN  424
0000H  08BCH  LIN  427
```


SER. #_____

```
0000H  08C3H  LIN   428
0000H  08D3H  LIN   431
0000H  08E3H  LIN   433
0000H  08EDH  LIN   435
0000H  08F6H  LIN   437
0000H  090CH  LIN   439
0000H  092CH  LIN   441
0000H  09A0H  LIN   444
0000H  09AAH  LIN   447
0000H  09B8H  LIN   449
0000H  09D1H  LIN   451
0000H  09D6H  LIN   453
0000H  09E0H  LIN   455
0000H  0938H  LIN   457
0000H  0944H  LIN   459
0000H  0950H  LIN   462
0000H  0962H  LIN   464
0000H  096EH  LIN   466
0000H  0979H  LIN   468
0000H  0981H  LIN   471
0000H  0996H  LIN   474
0000H  099EH  LIN   476
0000H  09E5H  LIN   478
0000H  09FFH  LIN   481
0000H  0A10H  LIN   483
0000H  0A1CH  LIN   485
0000H  0A23H  LIN   488
0000H  0A29H  LIN   491
0000H  0A37H  LIN   493
0000H  0A40H  LIN   495
0000H  0A47H  LIN   497
0000H  0A57H  LIN   503
0000H  0A61H  LIN   506
0000H  0A68H  LIN   509
0000H  0A72H  LIN   512
0000H  0A7DH  LIN   515
0000H  0A97H  LIN   517
0000H  0AACH  LIN   520
0000H  0ABFH  LIN   523
0000H  0B5EH  LIN   526
0000H  0B66H  LIN   528
0000H  0B78H  LIN   530
0000H  0AC7H  LIN   532
0000H  0AD8H  LIN   536
0000H  0AE5H  LIN   539
0000H  0AEFH  LIN   541
0000H  0AF8H  LIN   543
0000H  0B17H  LIN   545
0000H  0B25H  LIN   547
0000H  0B32H  LIN   549
0000H  0B3EH  LIN   551
0000H  0B43H  LIN   554
0000H  0B54H  LIN   556
0000H  0B7AH  LIN   559
0000H  0B84H  LIN   561
0000H  0B8AH  LIN   563
0000H  0B8FH  LIN   565
0000H  0B95H  LIN   567
0000H  0BA9H  LIN   569
0000H  0BAFH  LIN   571

0000H  08C6H  LIN   429
0000H  08D9H  LIN   432
0000H  08E4H  LIN   434
0000H  08F0H  LIN   436
0000H  0901H  LIN   438
0000H  0912H  LIN   440
0000H  092EH  LIN   442
0000H  09A3H  LIN   446
0000H  09B4H  LIN   448
0000H  09CFH  LIN   450
0000H  09D6H  LIN   452
0000H  09DDH  LIN   454
0000H  0931H  LIN   456
0000H  093AH  LIN   458
0000H  0949H  LIN   460
0000H  0953H  LIN   463
0000H  0968H  LIN   465
0000H  0974H  LIN   467
0000H  0978H  LIN   470
0000H  098FH  LIN   473
0000H  0999H  LIN   475
0000H  09E2H  LIN   477
0000H  09EEH  LIN   480
0000H  0A09H  LIN   482
0000H  0A17H  LIN   484
0000H  0A21H  LIN   487
0000H  0A26H  LIN   490
0000H  0A30H  LIN   492
0000H  0A3AH  LIN   494
0000H  0A43H  LIN   496
0000H  0A4AH  LIN   501
0000H  0A5EH  LIN   504
0000H  0A68H  LIN   508
0000H  0A70H  LIN   510
0000H  0A75H  LIN   513
0000H  0A87H  LIN   516
0000H  0AA6H  LIN   519
0000H  0ABAH  LIN   521
0000H  0AC1H  LIN   524
0000H  0B61H  LIN   527
0000H  0B73H  LIN   529
0000H  0AC4H  LIN   531
0000H  0ACEH  LIN   534
0000H  0AE2H  LIN   537
0000H  0AE8H  LIN   540
0000H  0AF6H  LIN   542
0000H  0B10H  LIN   544
0000H  0B1EH  LIN   546
0000H  0B2AH  LIN   548
0000H  0B39H  LIN   550
0000H  0B40H  LIN   553
0000H  0B4AH  LIN   555
0000H  0B5CH  LIN   558
0000H  0B7DH  LIN   560
0000H  0B87H  LIN   562
0000H  0B8DH  LIN   564
0000H  0B92H  LIN   566
0000H  0BA7H  LIN   568
0000H  0BACH  LIN   570
0000H  0BB8H  LIN   572
```

```
0000H  0BC2H  LIN   574
0000H  0BD0H  LIN   577
0000H  0BD5H  LIN   580
0000H  0BE5H  LIN   582
0000H  0BECH  LIN   586
0000H  0BF2H  LIN   588
0000H  0C05H  LIN   590
0000H  0C0FH  LIN   592
0000H  0C15H  LIN   594
0000H  0C28H  LIN   597
0000H  0C36H  LIN   600
0000H  0C38H  LIN   603
0000H  0C4BH  LIN   605
0000H  0C52H  LIN   609
0000H  0C5CH  LIN   611
0000H  0C68H  LIN   613
0000H  0C6DH  LIN   617
0000H  0C77H  LIN   619
0000H  0C81H  LIN   621
0000H  0C91H  LIN   623
0000H  0C9DH  LIN   625
0000H  0CAFH  LIN   627
0000H  0CC3H  LIN   629
0000H  0CCFH  LIN   631
0000H  0CE1H  LIN   633
0000H  0CE8H  LIN   635
0000H  0CF9H  LIN   638
0000H  0CFEH  LIN   641
0000H  0D08H  LIN   644
0000H  0D18H  LIN   647
0000H  0D1DH  LIN   651
0000H  0D29H  LIN   654
0000H  0D30H  LIN   657
0000H  0D3EH  LIN   659
0000H  0D4CH  LIN   661
0000H  0D5CH  LIN   664
0000H  0D61H  LIN   666
0000H  0D67H  LIN   668
0000H  0D6DH  LIN   670
0000H  0D72H  LIN   672
0000H  0D82H  LIN   675
0000H  0D8FH  LIN   678
0000H  0D9FH  LIN   681
0000H  0DA5H  LIN   683
0000H  0DB1H  LIN   686
0000H  0DB7H  LIN   688
0000H  0DCAH  LIN   691
0000H  0DD0H  LIN   695
0000H  0DD9H  LIN   697
0000H  0DF1H  LIN   700
0000H  0DF7H  LIN   703
0000H  0DFEH  LIN   705
0000H  0E69H  LIN   707
0000H  0E73H  LIN   709
0000H  0E08H  LIN   711
0000H  0E18H  LIN   714
0000H  0E27H  LIN   717
0000H  0E2DH  LIN   719
0000H  0E40H  LIN   722
0000H  0E46H  LIN   725

0000H  0BC9H  LIN   576
0000H  0BD3H  LIN   579
0000H  0BE1H  LIN   581
0000H  0BEAH  LIN   585
0000H  0BFAH  LIN   587
0000H  0BFBH  LIN   589
0000H  0C00H  LIN   591
0000H  0C12H  LIN   593
0000H  0C1EH  LIN   595
0000H  0C2FH  LIN   599
0000H  0C39H  LIN   601
0000H  0C47H  LIN   604
0000H  0C50H  LIN   608
0000H  0C55H  LIN   610
0000H  0C61H  LIN   612
0000H  0C6AH  LIN   615
0000H  0C74H  LIN   618
0000H  0C7AH  LIN   620
0000H  0C84H  LIN   622
0000H  0C96H  LIN   624
0000H  0CAAH  LIN   626
0000H  0CB6H  LIN   628
0000H  0CC8H  LIN   630
0000H  0CDCH  LIN   632
0000H  0CE8H  LIN   634
0000H  0CF6H  LIN   637
0000H  0CFCH  LIN   640
0000H  0D01H  LIN   642
0000H  0D11H  LIN   646
0000H  0D1DH  LIN   649
0000H  0D24H  LIN   652
0000H  0D29H  LIN   656
0000H  0D39H  LIN   658
0000H  0D47H  LIN   660
0000H  0D55H  LIN   662
0000H  0D5EH  LIN   665
0000H  0D64H  LIN   667
0000H  0D6AH  LIN   669
0000H  0D6FH  LIN   671
0000H  0D7BH  LIN   674
0000H  0D85H  LIN   677
0000H  0D96H  LIN   679
0000H  0DA2H  LIN   682
0000H  0DAEH  LIN   685
0000H  0DB4H  LIN   687
0000H  0DC1H  LIN   689
0000H  0DCDH  LIN   692
0000H  0DD2H  LIN   696
0000H  0DE0H  LIN   698
0000H  0DF4H  LIN   701
0000H  0DFCH  LIN   704
0000H  0E66H  LIN   706
0000H  0E70H  LIN   708
0000H  0E01H  LIN   710
0000H  0E12H  LIN   712
0000H  0E1EH  LIN   715
0000H  0E2AH  LIN   718
0000H  0E37H  LIN   720
0000H  0E43H  LIN   723
0000H  0E49H  LIN   727
```

```
0000H  0E4BH  LIN    729
0000H  0E59H  LIN    731
0000H  0E5FH  LIN    734
0000H  0E75H  LIN    736
0000H  0E7BH  LIN    738
0000H  0E8CH  LIN    740
0000H  0E98H  LIN    743
0000H  0EA4H  LIN    746
0000H  0EAAH  LIN    748
0000H  0EB7H  LIN    750
0000H  0EC3H  LIN    753
0000H  0ECCH  LIN    756
0000H  0EDAH  LIN    759
0000H  0EE2H  LIN    761
0000H  0EE7H  LIN    763
0000H  0EF6H  LIN    766
0000H  0EFDH  LIN    769
0000H  0F02H  LIN    772
0000H  0F10H  LIN    775
0000H  0F22H  LIN    777
0000H  0F2EH  LIN    779
0000H  0F38H  LIN    781
0000H  0F46H  LIN    784
0000H  0F4EH  LIN    787
0000H  0F51H  LIN    790
0000H  0F6AH  LIN    792
0000H  0F74H  LIN    794
0000H  0F78H  LIN    796
0000H  0F7EH  LIN    798
0000H  0F9AH  LIN    800
0000H  0FB2H  LIN    803
0000H  0FBEH  LIN    805
0000H  0FCAH  LIN    808
0000H  0FD2H  LIN    811
0000H  0FDCH  LIN    813
0000H  0FE4H  LIN    815
0000H  0FE8H  LIN    817
0000H  0FF0H  LIN    819
0000H  0FFEH  LIN    822
0000H  100DH  LIN    824
0000H  1010H  LIN    827
0000H  101CH  LIN    829
0000H  102AH  LIN    833
0000H  1036H  LIN    836
0000H  103DH  LIN    839
0000H  1049H  LIN    841
0000H  1055H  LIN    843
0000H  1063H  LIN    846
0000H  106FH  LIN    848
0000H  1074H  LIN    851
0000H  1082H  LIN    855
0000H  1087H  LIN    857
0000H  109AH  LIN    859
0000H  10A6H  LIN    861
0000H  10ABH  LIN    863
0000H  10B5H  LIN    866
0000H  10C5H  LIN    869
0000H  10CFH  LIN    872
0000H  10DDH  LIN    875
0000H  10E2H  LIN    878
0000H  0E52H  LIN    730
0000H  0E5CH  LIN    733
0000H  0E64H  LIN    735
0000H  0E78H  LIN    737
0000H  0E85H  LIN    739
0000H  0E95H  LIN    742
0000H  0EA1H  LIN    745
0000H  0EA7H  LIN    747
0000H  0EADH  LIN    749
0000H  0EC0H  LIN    751
0000H  0EC5H  LIN    755
0000H  0ED3H  LIN    758
0000H  0EDDH  LIN    760
0000H  0EE4H  LIN    762
0000H  0EE8H  LIN    765
0000H  0EFAH  LIN    767
0000H  0EFFH  LIN    770
0000H  0FD9H  LIN    773
0000H  0F1BH  LIN    776
0000H  0F29H  LIN    778
0000H  0F2EH  LIN    780
0000H  0F3FH  LIN    783
0000H  0F4AH  LIN    786
0000H  0F4EH  LIN    788
0000H  0F5DH  LIN    791
0000H  0F6EH  LIN    793
0000H  0F78H  LIN    795
0000H  0F7BH  LIN    797
0000H  0F8CH  LIN    799
0000H  0FA4H  LIN    801
0000H  0FB7H  LIN    804
0000H  0FC5H  LIN    807
0000H  0FD0H  LIN    809
0000H  0FD9H  LIN    812
0000H  0FDEH  LIN    814
0000H  0FE6H  LIN    816
0000H  0FEDH  LIN    818
0000H  0FF7H  LIN    821
0000H  1003H  LIN    823
0000H  1010H  LIN    826
0000H  1017H  LIN    828
0000H  1023H  LIN    831
0000H  1031H  LIN    834
0000H  1036H  LIN    837
0000H  1044H  LIN    840
0000H  1050H  LIN    842
0000H  105CH  LIN    845
0000H  1068H  LIN    847
0000H  1074H  LIN    850
0000H  107BH  LIN    853
0000H  1085H  LIN    856
0000H  1095H  LIN    858
0000H  10A1H  LIN    860
0000H  10A9H  LIN    862
0000H  10AEH  LIN    864
0000H  10BEH  LIN    868
0000H  10CCH  LIN    870
0000H  10D8H  LIN    873
0000H  10DFH  LIN    877
0000H  10E8H  LIN    879
```

```
0000H  10FAH  LIN     880          0000H  1106H  LIN     882
0000H  1113H  LIN     883          0000H  1117H  LIN     884
0000H  1119H  LIN     885          0000H  1126H  LIN     887
0000H  1131H  LIN     888          0000H  1136H  LIN     889
0000H  113AH  LIN     890          0000H  1140H  LIN     891
0000H  1146H  LIN     892          0000H  114DH  LIN     893
0000H  1152H  LIN     894          0000H  1155H  LIN     895
0000H  1158H  LIN     896          0000H  115EH  LIN     897
0000H  1164H  LIN     898          0000H  1168H  LIN     899
0000H  116EH  LIN     900          0000H  1175H  LIN     901
0000H  117AH  LIN     902          0000H  1181H  LIN     903
0000H  1184H  LIN     904          0000H  1188H  LIN     905
0000H  1197H  LIN     907          0000H  119CH  LIN     909
0000H  11A2H  LIN     909          0000H  11A9H  LIN     910
0000H  11ACH  LIN     911          0000H  11B3H  LIN     912
0000H  11B6H  LIN     913          0000H  11BDH  LIN     914
0000H  11C0H  LIN     915          0000H  11C6H  LIN     916
0000H  11CDH  LIN     917          0000H  11D4H  LIN     919
0000H  11DAH  LIN     920          0000H  11E4H  LIN     921
0000H  11E6H  LIN     922          0000H  11E8H  LIN     924
0000H  11EEH  LIN     925          0000H  0102H  LIN     930

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

SEGMENT MAP

START     STOP      LENGTH ALIGN NAME            CLASS

00000H    011F0H    11F1H   G    CODE            CODE
10000H    10107H    0108H   G    DATS            DATA
10108H    1023FH    0138H   W    DATA            DATA
10240H    10297H    0058H   W    STACK           STACK
10298H    1095AH    06C3H   W    CONST           CONST
10960H    10960H    0000H   G    ??SEG
10960H    10960H    0000H   W    MEMORY          MEMORY


GROUP MAP

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