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


            $compact
            $title ('ERA: Utility to Erase File for MP/M & CCP/M')
  1         erase:
            do;

            /*
              Revised:
                 19 Jan  80  by Thomas Rolander (MP/M 1.1)
                 19 July 81  by Doug Huskey     (MP/M II )
                  8 Aug  81  by Danny Horovitz  (MP/M-86 )
                 31 Jan  83  by Bill Fitler     (CCP/M-86 )
            */
            /* ERA checks if files are open by other users */

            $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.
=           ****/

            $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$Mask literally '0fdh';  /* mask out ls_network bit */
        =
  4   1 =   declare Ver$BDOS literally '30h';   /* minimal BDOS version rqd */
        =

  5   1     declare
            true    literally '1',
            false   literally '0',
            forever literally 'while true',
            lit     literally 'literally',
            proc    literally 'procedure',
            dcl     literally 'declare',
            addr    literally 'address',
            cr      literally '13',
            lf      literally '10',
            ctrlc   literally '3',
            ctrlx   literally '18h',
            bksp    literally '8';

        $include (:f2:proces.lit)
        =
        =   /*
        =       Proces Literals MP/M-8086 II
        =   */
        =
  6   1 =   declare pnamsiz literally '8';
        =
  7   1 =   declare pd$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';
        =
  8   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)';
        =
  9   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';
        =
 10   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';
        =
11  1   =     declare pcm$11             lit '00001h',
        =           pcm$ctls            lit '00002h',
        =           pcm$rout            lit '00004h',
        =           pcm$ctlc            lit '00008h',
        =           pcm$ctlo            lit '00080h',
        =           pcm$rsx             lit '00300h';

        $include (:f2:uda.lit)
        =
        =   /* MP/M-86 II  User Data Area format - August 8, 1981 */
        =
12  1   =   declare uda$structure lit 'structure (
        =       dparam          word,
        =       dma$ofst        word,
        =       dma$seg         word,
        =       func            byte,
        =       searchl         byte,
        =       searcha         word,
        =       searchabase     word,
        =       dcnt            word,
        =       dblk            word,
        =       error$mode      byte,
        =       mult$cnt        byte,
        =       df$password     (8) byte,
        =       pd$cnt          byte)';
        =


13  1       dcl stack$siz lit '16';
14  1       dcl int3 lit '0CCCCh';
15  1       dcl plmstack (stack$siz) word public initial(
                int3,int3,int3,int3, int3,int3,int3,int3,
                int3,int3,int3,int3, int3,int3,int3,int3);
16  1       dcl stack$size word public data(stack$siz + stack$siz);

          /****************************************
           *                                      *
           *       B D O S   INTERFACE            *
           *                                      *
           ****************************************/


17  1         bdos:
            procedure (func,info) external;
```

```
 18   2            declare func byte;
 19   2            declare info address;
 20   2          end mon1;

 21   1        mon2:
              procedure (func,info) byte external;
 22   2            declare func byte;
 23   2            declare info address;
 24   2          end mon2;

 25   1        mon3:
              procedure (func,info) address external;
 26   2            declare func byte;
 27   2            declare info address;
 28   2          end mon3;

 29   1        mon4:
              procedure (func,info) pointer external;
 30   2            declare func byte;
 31   2            declare info address;
 32   2          end mon4;


 33   1        declare cmdrv     byte    external;   /* command drive      */
 34   1        declare fcb (1)   byte    external;   /* 1st default fcb    */
 35   1        declare fcb16 (1) byte    external;   /* 2nd default fcb    */
 36   1        declare pass0     address external;   /* 1st password ptr   */
 37   1        declare len0      byte    external;   /* 1st passwd length  */
 38   1        declare pass1     address external;   /* 2nd password ptr   */
 39   1        declare len1      byte    external;   /* 2nd passwd length  */
 40   1        declare tbuff (1) byte    external;   /* default dma buffer */

              /****************************************
               *                                      *
               *       B D O S   Externals            *
               *                                      *
               ****************************************/

 41   1        read$console:
              procedure byte;
 42   2            return mon2 (1,0);
 43   2          end read$console;


 44   1        printchar:
              procedure(char);
 45   2          declare char byte;
 46   2          call mon1(2,char);
 47   2          end printchar;

 48   1        conin:
              procedure byte;
 49   2          return mon2(6,0fdh);
 50   2          end conin;
```

```
51   1        check$con$stat:
                procedure byte;
52   2          return mon2(11,0);
53   2        end check$con$stat;

54   1        print$buf:
                procedure (buffer$address);
55   2          declare buffer$address address;
56   2          call mon1 (9,buffer$address);
57   2        end print$buf;

58   1        version: procedure address;
                /* returns current cp/m version # */
59   2          return mon3(12,0);
60   2        end version;

61   1        setdma: procedure(dma);
62   2          declare dma address;
63   2          call mon1(26,dma);
64   2          end setdma;

65   1        search:
                procedure (fcb$address) byte;
66   2            declare fcb$address address;
67   2            return mon2 (17,fcb$address);
68   2          end search;

69   1        searchn:
                procedure byte;
70   2            return mon2 (18,0);
71   2          end searchn;

72   1        delete$file:
                procedure (fcb$address) address;
73   2            declare fcb$address address;
74   2            return mon3 (19,fcb$address);
75   2          end delete$file;

76   1        get$user$code:
                procedure byte;
77   2            return mon2 (32,0ffh);
78   2          end get$user$code;

                /* 0ff => return BDOS errors */
79   1        return$errors:
                procedure;
80   2            call mon1 (45,0ffh);
81   2          end return$errors;

82   1        terminate:
                procedure;
83   2            call mon1 (143,0);
84   2          end terminate;

85   1        declare
                parse$fn structure (
                  buff$adr  address,
```

```
                fcb$adr   address);
 86   1        declare (saveax,savecx) word external;  /* reg return vals, set in mon1 */

 87   1        parse: procedure;
 88   2          declare (retcode,errcode) word;

 89   2          call mon1(152,.parse$fn);
 90   2          retcode = saveax;
 91   2          errcode = savecx;
 92   2          if retcode = 0ffffh then       /* parse returned an error*/
 93   2             do;
 94   3               call print$buf(.('Invalid Filespec$'));
 95   3               if errcode = 23 then call print$buf(.(' (drive)$'));
 97   3               else if errcode = 24 then call print$buf(.(' (filename)$'));
 99   3               else if errcode = 25 then call print$buf(.(' (filetype)$'));
101   3               else if errcode = 38 then call print$buf(.(' (password)$'));
                      call print$buf(.('.',13,10,'$')); call terminate;
105   3             end;
106   2          end parse;

107   1      declare
                 pd$pointer pointer,
                 pd         based pd$pointer pd$structure;
108   1      declare
                 uda$pointer pointer,
                 uda$ptr structure (
                           offset word,
                           segment word) at (@uda$pointer),
                 uda based uda$pointer uda$structure;

109   1      get$uda: procedure;

110   2          pd$pointer = mon4(156,0);
111   2          uda$ptr.segment = pd.uda;
112   2          uda$ptr.offset = 0;
113   2          end get$uda;

              /****************************************
               *                                      *
               *         GLOBAL VARIABLES             *
               *                                      *
               ****************************************/

114   1        declare xfcb     byte initial(0);
115   1        declare successful lit '0FFh';

              /****************************************
               *                                      *
               *       S U B R O U T I N E S          *
               *                                      *
               ****************************************/


                          /* upper case character from console */
116   1      crlf:   proc;
117   2          call printchar(cr);
118   2          call printchar(lf);
```

```
119   2         end crlf;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* fill string @ s for c bytes with f */
120   1     fill:   proc(s,f,c);
121   2         dcl s addr,
                      (f,c) byte,
                      a based s byte;

122   2         do while (c:=c-1)<>255;
123   3         a = f;
124   3         s = s+1;
125   3         end;
126   2     end fill;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* error message routine */
127   1     error:   proc(code);
128   2        declare
                   code byte;

129   2         call printchar(' ');
130   2         if code=1 then
131   2             call print$buf(.(cr,lf,'Disk I/O Error.$'));
132   2         if code=2 then
133   2             call print$buf(.(cr,lf,'Drive $'));
134   2         if code = 3 or code = 2 then
135   2             call print$buf(.('Read Only$'));
136   2         if code = 4 then
137   2             call print$buf(.(cr,lf,'Invalid Filespec (drive).$'));
138   2         if code = 5 then
139   2             call print$buf(.('Currently Opened$'));
140   2         if code = 7 then
141   2             call print$buf(.('Password Error$'));
142   2         if code < 3 or code = 4 then
143   2             call terminate;
144   2         end error;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                       /* print file name */
145   1     print$file: procedure(fcbp);
146   2             declare k byte;
147   2             declare typ lit '9';        /* file type */
148   2             declare fnam lit '11';       /* file type */
149   2             declare
                        fcbp   addr,
                        fcbv   based fcbp (32) byte;

150   2                 do k = 1 to fnam;
151   3                 if k = typ then
152   3                call printchar('.');
153   3                 call printchar(fcbv(k) and 7fh);
154   3                 end;
155   2             end print$file;
```

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


                  /* try to delete fcb at fcb$address
                        return error code if unsuccessful */
156   1           delete:
                    procedure(fcb$address) byte;
157   2           declare
                       fcb$address address,
                       fcbv based  fcb$address (32) byte,
                       error$code  address,
                       code        byte;

158   2           if xfcb then
159   2               fcbv(5) = fcbv(5) or 80h;
160   2           call setdma(.fcb16);                    /* password */
161   2           fcbv(0) = fcb(0);                       /* drive */
162   2           error$code = delete$file(fcb$address);
163   2           fcbv(5) = fcbv(5) and 7fh;              /* reset xfcb bit */
164   2           if low(error$code) = 0FFh then do;
166   3               code = high(error$code);
167   3               if (code=1) or (code=2) or (code=4) then
168   3                   call error(code);
169   3               return code;
170   3               end;
171   2           return successful;
172   2           end delete;
                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                            /* upper case character from console */
173   1       ucase:   proc byte;
174   2           dcl c byte;

175   2           if (c:=conin) >= 'a' then
176   2              if c < '{' then
177   2                 return(c-20h);
178   2           return c;
179   2           end ucase;
                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                            /* get password and place at fcb + 16 */
180   1       getpasswd:   proc;
181   2           dcl (i,c) byte;

182   2           call crlf;
183   2           call print$buf(.('Password ? ','$'));
184   2       retry:
                    call fill(.fcb16,' ',8);
185   2               do i = 0 to 7;
186   3       nxtchr:
                        if (c:=ucase) >= ' ' then
187   3                      fcb16(i)=c;
188   3                  if c = cr then do;
190   4                      call crlf;
191   4                      goto exit;
```

```
192   4                 end;
193   3               if c = ctrlx then
194   3                   goto retry;
195   3               if c = bksp then do;
197   4                   if i<1 then
198   4                       goto retry;
199   4                   else do;
200   5                       fcb16(i:=i-1)=' ';
201   5                       goto nxtchr;
202   5                       end;
203   4                   end;
204   3               if c = 3 then
205   3                   call terminate;
206   3               end;
207   2       exit:
                    c = check$con$stat;
208   2          end getpasswd;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                            /* try deleting files one at a time */
209   1         single$file:
                  procedure;
210   2           declare (code,dcnt,sav$searchl) byte;
211   2           declare (fcba,sav$dcnt) addr;

212   2           file$err: procedure;
213   3               call crlf;
214   3               call print$buf(.('Not erased: $'));
215   3               call print$file(fcba);
216   3               call error(code);
217   3               end file$err;

218   2           call setdma(.tbuff);
219   2           dcnt = search(.fcb);
220   2               do while dcnt <> 0ffh;
221   3               fcba = shl(dcnt,5) + .tbuff;
222   3               sav$dcnt = uda.dcnt;
223   3               sav$searchl = uda.searchl;
224   3               if (code:=delete(fcba)) = 7 then do;
226   4                   call file$err;
227   4                   call getpasswd;
228   4                   code = delete(fcba);
229   4                   end;
230   3               if code <> successful then
231   3                   call file$err;
232   3               call setdma(.tbuff);
                        /* restore dcnt and search length of 11 */
233   3               uda.dcnt = sav$dcnt;
234   3               uda.searchl = sav$searchl;
235   3               dcnt = searchn;
236   3               end;
237   2         end single$file;

                  /****************************************
                   *                                      *
                   *       M A I N   P R O G R A M        *
```

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


238   1       declare (i,response,user,code) byte;
239   1       declare ver address;
240   1       declare last$dseg$byte byte
                initial (0);

241   1       plm$start: procedure public;

242   2           ver = version;
243   2           if low(ver) < Ver$BDOS  or  (high(ver) and Ver$Mask) = 0 then do;
245   3             call print$buf (.(cr,lf,Ver$Needs$OS,'$'));
246   3             call mon1(0,0);
247   3             end;

248   2           parse$fn.buff$adr = .tbuff(1);
249   2           parse$fn.fcb$adr = .fcb;
250   2           user = get$user$code;
251   2           call getuda;                /* get uda address */
252   2           call return$errors;
253   2           if fcb(17) <> ' ' then
254   2               if fcb(17) = 'X' then
255   2                   xfcb = true;
256   2               else do;
257   3                   call print$buf (.(
                        'Invalid Command Option.$'));
258   3                   call terminate;
259   3                   end;

260   2           i = 0;
261   2             do while fcb(i:=i+1) = '?';
262   3             ;
263   3             end;
264   2           if i > 11 then
265   2             if not xfcb then
266   2               do;
267   3               call print$buf (.(
                          'Confirm delete all user files (Y/N)?',"$"));
268   3               response = read$console;
269   3               if not ((response = 'y') or
                             (response = 'Y'))
                        then call terminate;
271   3               end;
272   2           call parse;
273   2           if (code:=delete(.fcb)) <> successful then do;
275   3               if code = 0 then
276   3                   call print$buf (.(cr,lf,
                            'File Not Found.',"$"));
277   3               else if code < 3 or code = 4 then
278   3                   call error(code);             /* fatal errors */
                        else
279   3                   call single$file;       /* single file error */
280   3               end;
281   2             call terminate;
282   2         end plm$start;
```

```
283  1      end erase;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
121	0000H	1	A. . . . . BYTE BASED(S) 123
107	0021H	1	ABORT. . . . . BYTE MEMBER(PD)
5			ADDR . . . . . LITERALLY 121 149 211
5			BKSP . . . . . LITERALLY 195
85	0000H	2	BUFFADR. . . . . WORD MEMBER(PARSEFN) 248
54	0004H	2	BUFFERADDRESS. . . WORD PARAMETER AUTOMATIC 55 56
181	003DH	1	C. . . . . BYTE 186 187 188 193 195 204 207
120	0004H	1	C. . . . . BYTE PARAMETER AUTOMATIC 121 122
174	003BH	1	C. . . . . BYTE 175 176 177 178
44	0004H	1	CHAR . . . . . BYTE PARAMETER AUTOMATIC 45 46
51	0033H	15	CHECKCONSTAT . . . PROCEDURE BYTE STACK=0008H 207
33	0000H	1	CMDRV. . . . . BYTE EXTERNAL(4)
107	0020H	1	CNS. . . . . BYTE MEMBER(PD)
157	003AH	1	CODE . . . . . BYTE 166 167 168 169
238	0044H	1	CODE . . . . . BYTE 273 275 277 278
127	0004H	1	CODE . . . . . BYTE PARAMETER AUTOMATIC 128 130 132 134 136 138 140 142
210	003EH	1	CODE . . . . . BYTE 216 224 228 230
48	0024H	15	CONIN. . . . . PROCEDURE BYTE STACK=0008H 175
107	0022H	2	CONMODE. . . . . WORD MEMBER(PD)
5			CR . . . . . LITERALLY 117 131 133 137 188 245 276
116	015AH	17	CRLF . . . . . PROCEDURE STACK=000EH 182 190 213
5			CTRLC. . . . . LITERALLY
5			CTRLX. . . . . LITERALLY 193
108	000EH	2	DBLK . . . . . WORD MEMBER(UDA)
5			DCL. . . . . LITERALLY
210	003FH	1	DCNT . . . . . BYTE 219 220 221 235
108	000CH	2	DCNT . . . . . WORD MEMBER(UDA) 222 233
156	0233H	91	DELETE . . . . . PROCEDURE BYTE STACK=0016H 224 228 273
72	0090H	16	DELETEFILE . . . . PROCEDURE WORD STACK=000AH 162
108	0012H	8	DFPASSWORD . . . . BYTE ARRAY(8) MEMBER(UDA)
61	0004H	2	DMA. . . . . WORD PARAMETER AUTOMATIC 62 63
108	0002H	2	DMADFST. . . . . WORD MEMBER(UDA)
108	0004H	2	DMASEG . . . . . WORD MEMBER(UDA)
108	0000H	2	DPARAM . . . . . WORD MEMBER(UDA)
107	0012H	1	DSK. . . . . BYTE MEMBER(PD)
107	0018H	2	DVRACT . . . . . WORD MEMBER(PD)
1	0002H		ERASE. . . . . PROCEDURE STACK=0000H
88	0026H	2	ERRCODE. . . . . WORD 91 95 97 99 101
127	018BH	112	ERROR. . . . . PROCEDURE STACK=0010H 168 216 278
157	0030H	2	ERRORCODE. . . . . WORD 162 164 166
108	0010H	1	ERRORMODE. . . . . BYTE MEMBER(UDA)
207	032FH		EXIT . . . . . LABEL 191
120	0006H	1	F. . . . . BYTE PARAMETER AUTOMATIC 121 123
5			FALSE. . . . . LITERALLY
34	0000H	1	FCB. . . . . BYTE ARRAY(1) EXTERNAL(5) 161 219 249 253 254 261 273
35	0000H	1	FCB16. . . . . BYTE ARRAY(1) EXTERNAL(6) 160 184 187 200
211	0032H	2	FCBA . . . . . WORD 215 221 224 228
156	0004H	2	FCBADDRESS . . . . WORD PARAMETER AUTOMATIC 157 159 161 162 163

```
 65  0004H     2  FCBADDRESS . . . .  WORD PARAMETER AUTOMATIC          66   67
 72  0004H     2  FCBADDRESS . . . .  WORD PARAMETER AUTOMATIC          73   74
 85  0002H     2  FCBADR . . . . . .  WORD MEMBER(PARSEFN)       249
145  0004H     2  FCBP . . . . . .  WORD PARAMETER AUTOMATIC         149  153
149  0000H    32  FCBV . . . . . .  BYTE BASED(FCBP) ARRAY(32)           153
157  0000H    32  FCBV . . . . . .  BYTE BASED(FCBADDRESS) ARRAY(32)          159  161  163
212  038AH    29  FILEERR. . . . .  PROCEDURE STACK=0014H           226  231
120  016BH    32  FILL . . . . . .  PROCEDURE STACK=0008H           184
107  0006H     2  FLAG . . . . . .  WORD MEMBER(PD)
148             FNAM . . . . . .  LITERALLY       150
  5             FOREVER. . . . .  LITERALLY
 29  0000H     1  FUNC . . . . . .  BYTE PARAMETER         30
 17  0000H     1  FUNC . . . . . .  BYTE PARAMETER         18
108  0006H     1  FUNC . . . . . .  BYTE MEMBER(UDA)
 21  0000H     1  FUNC . . . . . .  BYTE PARAMETER         22
 25  0000H     1  FUNC . . . . . .  BYTE PARAMETER         26
180  02AEH   137  GETPASSWD. . . . .  PROCEDURE STACK=0012H           227
109  0132H    40  GETUDA . . . . . .  PROCEDURE STACK=0008H           251
 76  00A0H    15  GETUSERCODE. . . .  PROCEDURE BYTE STACK=0008H           250
                 HIGH . . . . . .  BUILTIN         166  243
238  0041H     1  I. . . . . .  BYTE        260  261  264
181  003CH     1  I. . . . . .  BYTE        185  187  197  200
 21  0000H     2  INFO . . . . . .  WORD PARAMETER         23
 17  0000H     2  INFO . . . . . .  WORD PARAMETER         19
 25  0000H     2  INFO . . . . . .  WORD PARAMETER         27
 29  0000H     2  INFO . . . . . .  WORD PARAMETER         31
 14             INT3 . . . . . .  LITERALLY        15
146  0039H     1  K. . . . . .  BYTE        150  151  153
240  0045H     1  LASTDSEGBYTE . . .  BYTE INITIAL
107  0014H     1  LDSK . . . . . .  BYTE MEMBER(PD)
 37  0000H     1  LEN0 . . . . . .  BYTE EXTERNAL(8)
 39  0000H     1  LEN1 . . . . . .  BYTE EXTERNAL(10)
  5             LF . . . . . .  LITERALLY       118  131  133  137  245  276
107  0000H     2  LINK . . . . . .  WORD MEMBER(PD)
  5             LIT. . . . . .  LITERALLY         9   10   11   12   13   14  115  147  148
                 LOW. . . . . .  BUILTIN         164  243
107  0024H     1  LST. . . . . .  BYTE MEMBER(PD)
107  0015H     1  LUSER. . . . . .  BYTE MEMBER(PD)
107  0016H     2  MEM. . . . . .  WORD MEMBER(PD)
 17  0000H        MON1 . . . . . .  PROCEDURE EXTERNAL(0) STACK=0000H          46   56   63   80   83   89
                   246
 21  0000H        MON2 . . . . . .  PROCEDURE BYTE EXTERNAL(1) STACK=0000H          42   49   52   67   70
                   77
 25  0000H        MON3 . . . . . .  PROCEDURE WORD EXTERNAL(2) STACK=0000H          59   74
 29  0000H        MON4 . . . . . .  PROCEDURE POINTER EXTERNAL(3) STACK=0000H          110
108  0011H     1  MULTCNT. . . . . .  BYTE MEMBER(UDA)
107  0008H     8  NAME . . . . . .  BYTE ARRAY(8) MEMBER(PD)
107  001DH     1  NET. . . . . .  BYTE MEMBER(PD)
186  02D4H        NXTCHR . . . . . .  LABEL       201
108  0000H     2  OFFSET . . . . . .  WORD MEMBER(UDAPTR)       112
107  001CH     1  ORG. . . . . .  BYTE MEMBER(PD)
107  001EH     2  PARENT . . . . . .  WORD MEMBER(PD)
 87  00CDH   101  PARSE. . . . . .  PROCEDURE STACK=000EH           272
 85  0020H     4  PARSEFN. . . . . .  STRUCTURE         89  248  249
 36  0000H     2  PASS0. . . . . .  WORD EXTERNAL(7)
 38  0000H     2  PASS1. . . . . .  WORD EXTERNAL(9)
 11             PCM11. . . . . .  LITERALLY
```

```
  11                    PCMCTLC. . . . . LITERALLY
  11                    PCMCTLO. . . . . LITERALLY
  11                    PCMCTLS. . . . . LITERALLY
  11                    PCMROUT. . . . . LITERALLY
  11                    PCMRSX . . . . . LITERALLY
 107  0000H     48      PD . . . . . STRUCTURE BASED(PDPOINTER)                 111
 108  001AH      1      PDCNT. . . . . BYTE MEMBER(UDA)
   7                    PDHDR. . . . . LITERALLY        107
 107  0028H      4      PDPOINTER. . . . POINTER        107  110  111
   8                    PDSTRUCTURE. . . LITERALLY      107
  10                    PFACTIVE . . . . LITERALLY
  10                    PFCHILDABORT . . LITERALLY
  10                    PFCTLC . . . . . LITERALLY
  10                    PFCTLD . . . . . LITERALLY
  10                    PFKEEP . . . . . LITERALLY
  10                    PFKERNAL . . . . LITERALLY
  10                    PFNOCTLS . . . . LITERALLY
  10                    PFPURE . . . . . LITERALLY
  10                    PFRAW. . . . . LITERALLY
  10                    PFRESOURCE . . . LITERALLY
  10                    PFSYS. . . . . LITERALLY
  10                    PFTABLE. . . . . LITERALLY
  10                    PFTEMPKEEP . . . LITERALLY
  15  0000H     32      PLMSTACK . . . . WORD ARRAY(16) PUBLIC INITIAL
 241  03D7H    246      PLMSTART . . . . PROCEDURE PUBLIC STACK=001EH
   6                    PNAMSIZ. . . . . LITERALLY
 107  002CH      2      PRET . . . . . WORD MEMBER(PD)
  54  0042H     16      PRINTBUF . . . . PROCEDURE STACK=000AH          94   96   98  100  102  103  131  133
                                          135  137  139  141  183  214  245  257  267  276
  44  0011H     19      PRINTCHAR. . . . PROCEDURE STACK=000AH          117  118  129  152  153
 145  01FBH     56      PRINTFILE. . . . PROCEDURE STACK=0010H          215
 107  0005H      1      PRIOR. . . . . BYTE MEMBER(PD)
   5                    PROC . . . . . LITERALLY        116  120  127  173  180
   9                    PSCIOWAIT. . . . LITERALLY
   9                    PSDELAY. . . . . LITERALLY
   9                    PSDQ . . . . . LITERALLY
   9                    PSFLAGWAIT . . . LITERALLY
   9                    PSNQ . . . . . LITERALLY
   9                    PSPOLL . . . . . LITERALLY
   9                    PSRUN. . . . . LITERALLY
   9                    PSSLEEP. . . . . LITERALLY
   9                    PSSWAP . . . . . LITERALLY
   9                    PSTERM . . . . . LITERALLY
  41  0002H     15      READCONSOLE. . . PROCEDURE BYTE STACK=0008H             268
 107  0028H      4      RESERVD. . . . . BYTE ARRAY(4) MEMBER(PD)
 238  0042H      1      RESPONSE . . . . BYTE       268  269
  88  0024H      2      RETCODE. . . . . WORD        90   92
 184  02BBH             RETRY. . . . . . LABEL      194  198
  79  00AFH     15      RETURNERRORS . . . PROCEDURE STACK=0008H          252
 120  0008H      2      S. . . . . WORD PARAMETER AUTOMATIC       121  123  124
 211  0034H      2      SAVDCNT. . . . . WORD       222  233
  86  0000H      2      SAVEAX . . . . . WORD EXTERNAL(12)          90
  86  0000H      2      SAVECX . . . . . WORD EXTERNAL(13)          91
 210  0040H      1      SAVSEARCHL . . . . BYTE       223  234
 107  002EH      2      SCRATCH. . . . . WORD MEMBER(PD)
  65  0071H     16      SEARCH . . . . . PROCEDURE BYTE STACK=000AH             219
 108  0008H      2      SEARCHA. . . . . WORD MEMBER(UDA)
```

```
 108  000AH     2  SEARCHABASE. . . .   WORD MEMBER(UDA)
 108  0007H     1  SEARCHL. . . . . .   BYTE MEMBER(UDA)          223  234
  69  0081H    15  SEARCHN. . . . . .   PROCEDURE BYTE STACK=0008H          235
 108  0002H     2  SEGMENT. . . . . .   WORD MEMBER(UDAPTR)       111
  61  0061H    16  SETDMA . . . . . .   PROCEDURE STACK=000AH          160  218  232
 107  0025H     1  SF3. . . . . . . .   BYTE MEMBER(PD)
 107  0026H     1  SF4. . . . . . . .   BYTE MEMBER(PD)
 107  0027H     1  SF5. . . . . . . .   BYTE MEMBER(PD)
                   SHL. . . . . . . .   BUILTIN        221
 209  0337H   131  SINGLEFILE . . . .   PROCEDURE STACK=001AH          279
  13                STACKSIZ . . . . .   LITERALLY       15   16
  16  0000H     2  STACKSIZE. . . . .   WORD PUBLIC DATA
 107  0004H     1  STAT . . . . . . .   BYTE MEMBER(PD)
 115                SUCCESSFUL . . . .   LITERALLY       171  230  273
  40  0000H     1  TBUFF. . . . . . .   BYTE ARRAY(1) EXTERNAL(11)          218  221  232  248
  82  00BEH    15  TERMINATE. . . . .   PROCEDURE STACK=0008H          104  143  205  258  270  281
 107  0002H     2  THREAD . . . . . .   WORD MEMBER(PD)
   5                TRUE . . . . . . .   LITERALLY       255
 147                TYP. . . . . . . .   LITERALLY       151
 173  028EH    32  UCASE. . . . . . .   PROCEDURE BYTE STACK=000CH          186
 108  0000H    27  UDA. . . . . . . .   STRUCTURE BASED(UDAPOINTER)         222  223  233  234
 107  0010H     2  UDA. . . . . . . .   WORD MEMBER(PD)      111
 108  002CH     4  UDAPOINTER . . . .   POINTER         108  222  223  233  234
 108  002CH     4  UDAPTR . . . . . .   STRUCTURE AT         111  112
  12                UDASTRUCTURE . . .   LITERALLY       108
 238  0043H     1  USER . . . . . . .   BYTE       250
 107  0013H     1  USER . . . . . . .   BYTE MEMBER(PD)
 239  0036H     2  VER. . . . . . . .   WORD       242  243
   4                VERBDOS. . . . . .   LITERALLY       243
   3                VERMASK. . . . . .   LITERALLY       243
   2                VERNEEDSOS . . . .   LITERALLY       245
   2                VEROS. . . . . . .   LITERALLY
  58  0052H    15  VERSION. . . . . .   PROCEDURE WORD STACK=0008H          242
 107  001AH     2  WAIT . . . . . . .   WORD MEMBER(PD)
 114  0038H     1  XFCB . . . . . . .   BYTE INITIAL         158  255  265
```


MODULE INFORMATION:

```
     CODE AREA SIZE     = 04C0H   1229D
     CONSTANT AREA SIZE = 012BH    299D
     VARIABLE AREA SIZE = 0046H     70D
     MAXIMUM STACK SIZE = 001EH     30D
     566 LINES READ
     0 PROGRAM ERROR(S)
```

END OF PL/M-86 COMPILATION

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

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

BASE   OFFSET TYPE SYMBOL              BASE   OFFSET TYPE SYMBOL

0000H  04D7H  PUB  PLMSTART            0000H  005DH  PUB  XDOS
0000H  005DH  PUB  MON1                0000H  005DH  PUB  MON2
0000H  005DH  PUB  MON3                0000H  005DH  PUB  MON4
1000H  0004H  PUB  BDISK               1000H  0006H  PUB  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
1000H  0108H  PUB  PLMSTACK            1000H  018CH  PUB  STACKSIZE

ERASE:  SYMBOLS AND LINES
1000H  02C0H  SYM  MEMORY              1000H  0108H  SYM  PLMSTACK
1000H  018CH  SYM  STACKSIZE           0000H  0102H  SYM  READCONSOLE
0000H  0111H  SYM  PRINTCHAR           STACK  0004H  SYM  CHAR
0000H  0124H  SYM  CONIN               0000H  0133H  SYM  CHECKCONSTAT
0000H  0142H  SYM  PRINTBUF            STACK  0004H  SYM  BUFFERADDRESS
0000H  0152H  SYM  VERSION             0000H  0161H  SYM  SETDMA
STACK  0004H  SYM  DMA                 0000H  0171H  SYM  SEARCH
STACK  0004H  SYM  FCBADDRESS          0000H  0181H  SYM  SEARCHN
0000H  0190H  SYM  DELETEFILE          STACK  0004H  SYM  FCBADDRESS
0000H  01A0H  SYM  GETUSERCODE         0000H  01AFH  SYM  RETURNERRORS
0000H  01BEH  SYM  TERMINATE           1000H  0128H  SYM  PARSEFN
0000H  01CDH  SYM  PARSE               1000H  012CH  SYM  RETCODE
1000H  012EH  SYM  ERRCODE             1000H  0130H  SYM  PDPOINTER
1000H  0130H  BAS  PD                  1000H  0134H  SYM  UDAPOINTER
1000H  0134H  SYM  UDAPTR              1000H  0134H  BAS  UDA
0000H  0232H  SYM  GETUDA              1000H  0140H  SYM  XFCB
0000H  025AH  SYM  CRLF                0000H  026BH  SYM  FILL
STACK  0008H  SYM  S                   STACK  0006H  SYM  F
STACK  0004H  SYM  C                   STACK  0008H  BAS  A
0000H  028BH  SYM  ERROR               STACK  0004H  SYM  CODE
0000H  02FBH  SYM  PRINTFILE           STACK  0004H  SYM  FCBP
1000H  0141H  SYM  K                   STACK  0004H  BAS  FCBV
0000H  0333H  SYM  DELETE              STACK  0004H  SYM  FCBADDRESS
STACK  0004H  BAS  FCBV                1000H  0138H  SYM  ERRORCODE
1000H  0142H  SYM  CODE                0000H  038EH  SYM  UCASE
1000H  0143H  SYM  C                   0000H  03AEH  SYM  GETPASSWD
1000H  0144H  SYM  I                   1000H  0145H  SYM  C
0000H  03BBH  SYM  RETRY               0000H  03D4H  SYM  NXTCHR
0000H  042FH  SYM  EXIT                0000H  0437H  SYM  SINGLEFILE
1000H  0146H  SYM  CODE                1000H  0147H  SYM  DCNT
1000H  0148H  SYM  SAVSEARCHL          1000H  013AH  SYM  FCBA
1000H  013CH  SYM  SAVDCNT             0000H  04BAH  SYM  FILEERR
1000H  0149H  SYM  I                   1000H  014AH  SYM  RESPONSE
1000H  0148H  SYM  USER                1000H  014CH  SYM  CODE
1000H  013EH  SYM  VER                 1000H  014DH  SYM  LASTDSEGBYTE
```

```
0000H  04D7H  SYM  PLMSTART
0000H  0105H  LIN     42
0000H  0111H  LIN     44
0000H  0120H  LIN     47
0000H  0127H  LIN     49
0000H  0133H  LIN     51
0000H  0142H  LIN     53
0000H  0145H  LIN     56
0000H  0152H  LIN     58
0000H  0161H  LIN     60
0000H  0164H  LIN     63
0000H  0171H  LIN     65
0000H  0181H  LIN     68
0000H  0184H  LIN     70
0000H  0190H  LIN     72
0000H  01A0H  LIN     75
0000H  01A3H  LIN     77
0000H  01AFH  LIN     79
0000H  01BCH  LIN     81
0000H  01C1H  LIN     83
0000H  01CDH  LIN     87
0000H  01DAH  LIN     90
0000H  01E6H  LIN     92
0000H  01F4H  LIN     95
0000H  0200H  LIN     97
0000H  020CH  LIN     99
0000H  0218H  LIN    101
0000H  0226H  LIN    103
0000H  0230H  LIN    106
0000H  0235H  LIN    110
0000H  0252H  LIN    112
0000H  025AH  LIN    116
0000H  0263H  LIN    118
0000H  026BH  LIN    120
0000H  027AH  LIN    123
0000H  0285H  LIN    125
0000H  028BH  LIN    127
0000H  0294H  LIN    130
0000H  02A1H  LIN    132
0000H  02AEH  LIN    134
0000H  02C1H  LIN    136
0000H  02CEH  LIN    138
0000H  02D8H  LIN    140
0000H  02E8H  LIN    142
0000H  02F7H  LIN    144
0000H  02FEH  LIN    150
0000H  0311H  LIN    152
0000H  0329H  LIN    154
0000H  0333H  LIN    156
0000H  033DH  LIN    159
0000H  0348H  LIN    161
0000H  035AH  LIN    163
0000H  0368H  LIN    166
0000H  037CH  LIN    168
0000H  0388H  LIN    171
0000H  038EH  LIN    173
0000H  039BH  LIN    176
0000H  03A9H  LIN    178
0000H  03AEH  LIN    180
0000H  03B4H  LIN    183

0000H  0102H  LIN     41
0000H  0111H  LIN     43
0000H  0114H  LIN     46
0000H  0124H  LIN     48
0000H  0133H  LIN     50
0000H  0136H  LIN     52
0000H  0142H  LIN     54
0000H  014EH  LIN     57
0000H  0155H  LIN     59
0000H  0161H  LIN     61
0000H  016DH  LIN     64
0000H  0174H  LIN     67
0000H  0181H  LIN     69
0000H  0190H  LIN     71
0000H  0193H  LIN     74
0000H  01A0H  LIN     76
0000H  01AFH  LIN     78
0000H  01B2H  LIN     80
0000H  01BEH  LIN     82
0000H  01CBH  LIN     84
0000H  01D0H  LIN     89
0000H  01E0H  LIN     91
0000H  01EDH  LIN     94
0000H  01FBH  LIN     96
0000H  0207H  LIN     98
0000H  0213H  LIN    100
0000H  021FH  LIN    102
0000H  022DH  LIN    104
0000H  0232H  LIN    109
0000H  0247H  LIN    111
0000H  0258H  LIN    113
0000H  025DH  LIN    117
0000H  0269H  LIN    119
0000H  026EH  LIN    122
0000H  0282H  LIN    124
0000H  0287H  LIN    126
0000H  028EH  LIN    129
0000H  029AH  LIN    131
0000H  02A7H  LIN    133
0000H  02BAH  LIN    135
0000H  02C7H  LIN    137
0000H  02D4H  LIN    139
0000H  02E1H  LIN    141
0000H  02F4H  LIN    143
0000H  02FBH  LIN    145
0000H  030AH  LIN    151
0000H  0317H  LIN    153
0000H  032FH  LIN    155
0000H  0336H  LIN    158
0000H  0344H  LIN    160
0000H  0353H  LIN    162
0000H  0361H  LIN    164
0000H  0370H  LIN    167
0000H  0383H  LIN    169
0000H  038EH  LIN    172
0000H  0391H  LIN    175
0000H  03A2H  LIN    177
0000H  03AEH  LIN    179
0000H  03B1H  LIN    182
0000H  03BBH  LIN    184
```

```
0000H  03C8H  LIN    185
0000H  03DEH  LIN    187
0000H  03F2H  LIN    190
0000H  03F7H  LIN    193
0000H  0405H  LIN    197
0000H  041DH  LIN    201
0000H  0426H  LIN    205
0000H  042FH  LIN    207
0000H  0437H  LIN    209
0000H  04BDH  LIN    213
0000H  04C7H  LIN    215
0000H  04D5H  LIN    217
0000H  0441H  LIN    219
0000H  0452H  LIN    221
0000H  046DH  LIN    223
0000H  0480H  LIN    226
0000H  0486H  LIN    228
0000H  0497H  LIN    231
0000H  04A1H  LIN    233
0000H  04B3H  LIN    235
0000H  04B8H  LIN    237
0000H  04DAH  LIN    242
0000H  04F0H  LIN    245
0000H  0500H  LIN    248
0000H  050CH  LIN    250
0000H  0515H  LIN    252
0000H  051FH  LIN    254
0000H  052DH  LIN    257
0000H  0537H  LIN    260
0000H  054FH  LIN    264
0000H  055FH  LIN    267
0000H  056CH  LIN    269
0000H  058DH  LIN    272
0000H  059EH  LIN    275
0000H  05AEH  LIN    277
0000H  05C5H  LIN    279
0000H  05CBH  LIN    282
0000H  03D4H  LIN    186
0000H  03EBH  LIN    188
0000H  03F5H  LIN    191
0000H  03FEH  LIN    195
0000H  040CH  LIN    200
0000H  041FH  LIN    204
0000H  0429H  LIN    206
0000H  0435H  LIN    208
0000H  04BAH  LIN    212
0000H  04C0H  LIN    214
0000H  04CEH  LIN    216
0000H  043AH  LIN    218
0000H  044BH  LIN    220
0000H  0461H  LIN    222
0000H  0475H  LIN    224
0000H  0483H  LIN    227
0000H  0490H  LIN    230
0000H  049AH  LIN    232
0000H  04ACH  LIN    234
0000H  04B8H  LIN    236
0000H  04D7H  LIN    241
0000H  04E0H  LIN    243
0000H  04F7H  LIN    246
0000H  0506H  LIN    249
0000H  0512H  LIN    251
0000H  0518H  LIN    253
0000H  0526H  LIN    255
0000H  0534H  LIN    258
0000H  053CH  LIN    261
0000H  0556H  LIN    265
0000H  0566H  LIN    268
0000H  058AH  LIN    270
0000H  0590H  LIN    273
0000H  05A5H  LIN    276
0000H  05BCH  LIN    278
0000H  05C8H  LIN    281
0000H  0102H  LIN    283
```


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

SEGMENT MAP

START	STOP	LENGTH	ALIGN	NAME	CLASS
00000H	005CCH	05CDH	G	CODE	CODE
10000H	10107H	0108H	G	DATS	DATA
10108H	10140H	0046H	W	DATA	DATA
1014EH	10188H	003EH	W	STACK	STACK
10188H	102B6H	012BH	W	CONST	CONST
102C0H	102C0H	0000H	G	??SEG	
102C0H	102C0H	0000H	W	MEMORY	MEMORY

GROUP MAP

ADDRESS	GROUP OR SEGMENT NAME
00000H	CGROUP

```
        CODE
10000H  DGROUP
        DATS
        STACK
        CONST
        DATA
```