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


          $title ('ERAQ: Erase File with Query')
1         eraseq:
          do;

          /*
            Revised:
              19 Jan  80  by Thomas Rolander
              20 July 81  by Doug Huskey
               6 Aug  81  by Danny Horovitz
              31 Jan  83  by Bill Fitler
          */

          $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 is_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';

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


 6   1       mon1:
               procedure (func,info) external;
 7   2           declare func byte;
 8   2           declare info address;
 9   2         end mon1;

10   1       mon2:
               procedure (func,info) byte external;
11   2           declare func byte;
12   2           declare info address;
13   2         end mon2;

14   1       mon3:
               procedure (func,info) address external;
15   2           declare func byte;
16   2           declare info address;
17   2         end mon3;



18   1       declare cmdrv     byte    external;   /* command drive      */
19   1       declare fcb (1)   byte    external;   /* 1st default fcb    */
20   1       declare fcb16 (1) byte    external;   /* 2nd default fcb    */
21   1       declare pass0     address external;   /* 1st password ptr   */
```

```
22  1   declare len0      byte    external;   /* 1st passwd length  */
23  1   declare pass1     address external;   /* 2nd password ptr   */
24  1   declare len1      byte    external;   /* 2nd passwd length  */
25  1   declare tbuff (1) byte    external;   /* default dma buffer */


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

26  1   read$console:
          procedure byte;
27  2       return mon2 (1,0);
28  2       end read$console;


29  1   printchar:
          procedure(char);
30  2       declare char byte;
31  2       call mon1(2,char);
32  2       end printchar;

33  1   conin:
          procedure byte;
34  2       return mon2(6,0fdh);
35  2       end conin;

36  1   print$buf:
          procedure (buffer$address);
37  2       declare buffer$address address;
38  2       call mon1 (9,buffer$address);
39  2       end print$buf;

40  1   check$con$stat:
          procedure byte;
41  2       return mon2(11,0);
42  2       end check$con$stat;

43  1   version: procedure address;
          /* returns current cp/m version # */
44  2       return mon3(12,0);
45  2       end version;

46  1   setdma: procedure(dma);
47  2       declare dma address;
48  2       call mon1(26,dma);
49  2       end setdma;

50  1   search$first:
          procedure (fcb$address) byte;
51  2       declare fcb$address address;
52  2       return mon2 (17,fcb$address);
53  2       end search$first;

54  1   search$next:
```

```
                  procedure byte;
55    2             return mon2 (18,0);
56    2           end search$next;

57    1         delete$file:
                  procedure (fcb$address) address;
58    2             declare fcb$address address;
59    2             return mon3 (19,fcb$address);
60    2           end delete$file;

61    1         get$user$code:
                  procedure byte;
62    2             return mon2 (32,0ffh);
63    2           end get$user$code;

                  /* 0ff => return BDOS errors */
64    1         return$errors:
                  procedure;
65    2             call mon1 (45,0ffh);
66    2           end return$errors;


67    1         terminate:
                  procedure;
68    2             call mon1 (143,0);
69    2           end terminate;

70    1         declare
                  parse$fn structure (
                    buff$adr  address,
                    fcb$adr   address);
71    1         declare (saveax,savecx) word external;  /* reg return vals, set in mon1 */

72    1         parse: procedure;
73    2           declare (retcode,errcode) word;

74    2           call mon1(152,.parse$fn);
75    2           retcode = saveax;
76    2           errcode = savecx;
77    2           if retcode = 0ffffh then       /* parse returned an error */
78    2              do;
79    3                call print$buf(.('Invalid Filespec$'));
80    3                if errcode = 23 then call print$buf(.(' (drive)$'));
82    3                else if errcode = 24 then call print$buf(.(' (filename)$'));
84    3                else if errcode = 25 then call print$buf(.(' (filetype)$'));
86    3                else if errcode = 38 then call print$buf(.(' (password)$'));
                         call print$buf(.('.',13,10,'$')); call terminate;
90    3              end;
91    2           end parse;



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

```
 92  1      declare xfcb     byte initial(0);
 93  1      declare successful lit '0FFh';

 94  1      declare dir$entries (128) structure (
                file (12) byte );

 95  1      declare dir$entry$adr address;
 96  1      declare dir$entry based dir$entry$adr (1) byte;

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

                        /* upper case character from console */
 97  1    crlf:   proc;
 98  2        call printchar(cr);
 99  2        call printchar(lf);
100  2        end crlf;
       /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* fill string @ s for c bytes with f */
101  1    fill:   proc(s,f,c);
102  2        dcl s addr,
                 (f,c) byte,
                 a based s byte;

103  2            do while (c:=c-1)<>255;
104  3            a = f;
105  3            s = s+1;
106  3            end;
107  2        end fill;
       /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* error message routine */
108  1    error:   proc(code);
109  2       declare
                  code byte;

110  2        call printchar(' ');
111  2        if code=1 then
112  2            call print$buf(.(cr,lf,'Disk I/O Error.$'));
113  2        if code=2 then
114  2            call print$buf(.(cr,lf,'Drive $'));
115  2        if code = 3 or code = 2 then
116  2            call print$buf(.('Read Only$'));
117  2        if code = 4 then
118  2            call print$buf(.(cr,lf,'Invalid Filespec (drive).$'));
119  2        if code = 5 then
120  2            call print$buf(.('Currently Opened$'));
121  2        if code = 7 then
122  2            call print$buf(.('Password Error$'));
```

```
123   2          if code < 3 or code = 4 then
124   2              call terminate;
125   2          end error;
                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                  /* try to delete fcb at fcb$address
                        return error code if unsuccessful */
126   1          delete:
                    procedure(fcb$address) byte;
127   2            declare
                       fcb$address address,
                       fcbv based  fcb$address (32) byte,
                       error$code  address,
                       code        byte;

128   2            if xfcb then
129   2                fcbv(5) = fcbv(5) or 80h;
130   2            call setdma(.fcb16);                    /* password */
131   2            fcbv(0) = fcb(0);                       /* drive */
132   2            error$code = delete$file(fcb$address);
133   2            fcbv(5) = fcbv(5) and 7fh;              /* reset xfcb bit */
134   2            if low(error$code) = 0FFh then do;
136   3                code = high(error$code);
137   3                if (code=1) or (code=2) then
138   3                    call error(code);
139   3                return code;
140   3                end;
141   2            return successful;
142   2            end delete;
                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                              /* upper case character from console */
143   1          ucase:   proc byte;
144   2              dcl c byte;

145   2              if (c:=conin) >= 'a' then
146   2                 if c < '{' then
147   2                    return(c-20h);
148   2              return c;
149   2              end ucase;
                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                              /* get password and place at fcb + 16 */
150   1          getpasswd:   proc;
151   2              dcl (i,c) byte;

152   2              call print$buf(.('Password ? ','$'));
153   2          retry:
                    call fill(.fcb16,' ',8);
154   2                do i = 0 to 7;
155   3          nxtchr:
                        if (c:=ucase) >= ' ' then
156   3                      fcb16(i)=c;
157   3                  if c = cr then
```

```
158  3                goto exit;
159  3            if c = ctrlx then
160  3                goto retry;
161  3            if c = bksp then do;
163  4                if i<1 then
164  4                    goto retry;
165  4                else do;
166  5                    fcb16(i:=i-1)=' ';
167  5                    goto nxtchr;
168  5                    end;
169  4                end;
170  3            if c = 3 then
171  3                call terminate;
172  3            end;
173  2    exit:
         c = check$con$stat;
174  2    end getpasswd;
     /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                    /* error on deleting a file */
175  1    file$err: procedure(code);
176  2        declare code byte;

177  2        call crlf;
178  2        call print$buf(.('Not erased, $'));
179  2        call error(code);
180  2        call crlf;
181  2        end file$err;



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


182  1    declare (i,j,k,code,response,user,dcnt) byte;
183  1    declare ver address;

184  1    declare last$dseg$byte byte
       initial (0);

185  1    plm$start: procedure public;
186  2        ver = version;
187  2        if low(ver) < Ver$BDOS  or  (high(ver) and Ver$Mask) = 0 then do;
189  3          call print$buf (.(cr,lf,Ver$Needs$OS,'$'));
190  3          call mon1(0,0);
191  3          end;

192  2        if fcb(17) <> ' ' then
193  2            if fcb(17) = 'X' then
194  2                xfcb = true;
195  2            else do;
196  3                call print$buf (.(
```

```
                    'Invalid Command Option.$'));
197   3                 call terminate;
198   3                 end;

199   2           parse$fn.buff$adr = .tbuff(1);
200   2           parse$fn.fcb$adr = .fcb;
201   2           call parse;

202   2           if fcb(0) = 0 then
203   2             fcb(0) = low (mon2 (25,0)) + 1;
204   2           i = -1;
205   2           user = get$user$code;
206   2           call return$errors;
207   2           dcnt = search$first (.fcb);
208   2           do while dcnt <> 0ffh;
209   3             dir$entry$adr = .tbuff(ror(dcnt,3) and 110$0000b);
210   3             if dir$entry(0) = user then
211   3             do;
212   4               if (i:=i+1) = 128 then
213   4               do;
214   5                 call print$buf (.(
                          'Too many directory entries for query.','$'));
215   5                 call terminate;
216   5               end;
217   4               call move (12,.dir$entry(1),.dir$entries(i));
218   4             end;
219   3             dcnt = search$next;
220   3           end;
221   2           if i = -1 then
222   2           do;
223   3             call print$buf (.(
                        'File Not Found.','$'));
224   3           end;
                    else
225   2           do j = 0 to i;
226   3             call printchar ('A'+fcb(0)-1);
227   3             call printchar (':');
228   3             call printchar (' ');
229   3             do k = 0 to 10;
230   4               if k = 8
                          then call printchar ('.');
232   4               call printchar (dir$entries(j).file(k) and 07FH);
233   4             end;
234   3             call printchar (' ');
235   3             call printchar ('?');
236   3             response = read$console;
237   3             call printchar (0dh);
238   3             call printchar (0ah);
239   3             if (response = 'y') or
                         (response = 'Y') then
240   3             do;
241   4               call move (12,.dir$entries(j),.fcb(1));
242   4               if (code:=delete(.fcb)) <> successful then do;
244   5                   if code < 3 or code = 4 then
245   5                       call error(code);           /* fatal errors */
246   5                   else if code = 7 then do;
248   6                       call file$err(code);
```

```
249   6                     call getpassword;
250   6                     code = delete(.fcb);
251   6                     end;
                        if code <> successful then
253   5                     call file$err(code);
254   5                   call crlf;
255   5                   end;
256   4             end;
257   3           end;
258   2           call terminate;
259   2         end plm$start;

260   1       end eraseq;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
102	0000H	1	A. . . . . BYTE BASED(S) 104
5			ADDR . . . . . LITERALLY 102
5			BKSP . . . . . LITERALLY 161
70	0000H	2	BUFFADR. . . . . WORD MEMBER(PARSEFN) 199
36	0004H	2	BUFFERADDRESS. . . WORD PARAMETER AUTOMATIC 37 38
151	0612H	1	C. . . . . BYTE 155 156 157 159 161 170 173
101	0004H	1	C. . . . . BYTE PARAMETER AUTOMATIC 102 103
144	0610H	1	C. . . . . BYTE 145 146 147 148
29	0004H	1	CHAR . . . . . BYTE PARAMETER AUTOMATIC 30 31
40	0041H	15	CHECKCONSTAT . . . PROCEDURE BYTE STACK=0008H 173
18	0000H	1	CMDRV. . . . . BYTE EXTERNAL(3)
175	0004H	1	CODE . . . . . BYTE PARAMETER AUTOMATIC 176 179
108	0004H	1	CODE . . . . . BYTE PARAMETER AUTOMATIC 109 111 113 115 117 119 121 123
127	060FH	1	CODE . . . . . BYTE 136 137 138 139
182	0616H	1	CODE . . . . . BYTE 242 244 245 246 248 250 252 253
33	0022H	15	CONIN. . . . . PROCEDURE BYTE STACK=0008H 145
5			CR . . . . . LITERALLY 98 112 114 118 157 189
97	0130H	17	CRLF . . . . . PROCEDURE STACK=000EH 177 180 254
5			CTRLC. . . . . LITERALLY
5			CTRLX. . . . . LITERALLY 159
5			DCL. . . . . LITERALLY
182	0619H	1	DCNT . . . . . BYTE 207 208 209 219
126	01D1H	87	DELETE . . . . . PROCEDURE BYTE STACK=0016H 242 250
57	008EH	16	DELETEFILE . . . . PROCEDURE WORD STACK=000AH 132
94	0008H	1536	DIRENTRIES . . . . STRUCTURE ARRAY(128) 217 232 241
96	0000H	1	DIRENTRY . . . . . BYTE BASED(DIRENTRYADR) ARRAY(1) 210 217
95	0608H	2	DIRENTRYADR. . . . WORD 96 209 210 217
46	0004H	2	DMA. . . . . WORD PARAMETER AUTOMATIC 47 48
1	0000H		ERASEQ . . . . . PROCEDURE STACK=0000H
73	0006H	2	ERRCODE. . . . . WORD 76 80 82 84 86
108	0161H	112	ERROR. . . . . PROCEDURE STACK=0010H 138 179 245
127	060AH	2	ERRORCODE. . . . . WORD 132 134 136
173	02C1H		EXIT . . . . . LABEL 158
101	0006H	1	F. . . . . BYTE PARAMETER AUTOMATIC 102 104
5			FALSE. . . . . LITERALLY
19	0000H	1	FCB. . . . . BYTE ARRAY(1) EXTERNAL(4) 131 192 193 200 202 203 207 226 241 242 250
20	0000H	1	FCB16. . . . . BYTE ARRAY(1) EXTERNAL(5) 130 153 156 166
50	0004H	2	FCBADDRESS . . . . WORD PARAMETER AUTOMATIC 51 52
126	0004H	2	FCBADDRESS . . . . WORD PARAMETER AUTOMATIC 127 129 131 132 133
57	0004H	2	FCBADDRESS . . . . WORD PARAMETER AUTOMATIC 58 59
70	0002H	2	FCBADR . . . . . WORD MEMBER(PARSEFN) 200
127	0000H	32	FCBV . . . . . BYTE BASED(FCBADDRESS) ARRAY(32) 129 131 133
94	0000H	12	FILE . . . . . BYTE ARRAY(12) MEMBER(DIRENTRIES) 232
175	02C9H	26	FILEERR. . . . . PROCEDURE STACK=0016H 248 253
101	0141H	32	FILL . . . . . PROCEDURE STACK=0008H 153
5			FOREVER. . . . . LITERALLY
6	0000H	1	FUNC . . . . . BYTE PARAMETER 7

```
 14  0000H     1  FUNC . . . . . .  BYTE PARAMETER          15
 10  0000H     1  FUNC . . . . . .  BYTE PARAMETER          11
150  0248H   129  GETPASSWD. . . .  PROCEDURE STACK=0010H            249
 61  009EH    15  GETUSERCODE. . .  PROCEDURE BYTE STACK=0008H             205
                  HIGH . . . . . .  BUILTIN           136  187
182  0613H     1  I. . . . . . . .  BYTE         204  212  217  221  225
151  0611H     1  I. . . . . . . .  BYTE         154  156  163  166
 14  0000H     2  INFO . . . . . .  WORD PARAMETER          16
 10  0000H     2  INFO . . . . . .  WORD PARAMETER          12
  6  0000H     2  INFO . . . . . .  WORD PARAMETER           8
182  0614H     1  J. . . . . . . .  BYTE         225  232  241
182  0615H     1  K. . . . . . . .  BYTE         229  230  232
184  061AH     1  LASTDSEGBYTE . .  BYTE INITIAL
 22  0000H     1  LEN0 . . . . . .  BYTE EXTERNAL(7)
 24  0000H     1  LEN1 . . . . . .  BYTE EXTERNAL(9)
  5             LF . . . . . . .  LITERALLY          99  112  114  118  189
  5             LIT. . . . . . .  LITERALLY          93
                LOW. . . . . . .  BUILTIN           134  187  203
  6  0000H        MON1 . . . . . .  PROCEDURE EXTERNAL(0) STACK=0000H          31   38   48   65   68   74
                                    190
 10  0000H        MON2 . . . . . .  PROCEDURE BYTE EXTERNAL(1) STACK=0000H          27   34   41   52   55
                                    62  203
 14  0000H        MON3 . . . . . .  PROCEDURE WORD EXTERNAL(2) STACK=0000H          44   59
                MOVE . . . . . .  BUILTIN           217  241
155  026BH        NXTCHR . . . . .  LABEL       167
 72  00CBH   101  PARSE. . . . . .  PROCEDURE STACK=000EH            201
 70  0000H     4  PARSEFN. . . . .  STRUCTURE          74  199  200
 21  0000H     2  PASS0. . . . . .  WORD EXTERNAL(6)
 23  0000H     2  PASS1. . . . . .  WORD EXTERNAL(8)
185  02E3H   500  PLMSTART . . . .  PROCEDURE PUBLIC STACK=001AH
 36  0031H    16  PRINTBUF . . . .  PROCEDURE STACK=000AH            79   81   83   85   87   88   112  114
                                    116  118  120  122  152  178  189  196  214  223
 29  000FH    19  PRINTCHAR. . . .  PROCEDURE STACK=000AH            98   99  110  226  227  228  231  232
                                    234  235  237  238
  5             PROC . . . . . .  LITERALLY          97  101  108  143  150
 26  0000H    15  READCONSOLE. . .  PROCEDURE BYTE STACK=0008H             236
182  0617H     1  RESPONSE . . . .  BYTE         236  239
 73  0004H     2  RETCODE. . . . .  WORD          75   77
153  0252H        RETRY. . . . . .  LABEL       160  164
 64  00ADH    15  RETURNERRORS . .  PROCEDURE STACK=0008H            206
                  ROR. . . . . . .  BUILTIN           209
101  0008H     2  S. . . . . . . .  WORD PARAMETER AUTOMATIC       102  104  105
 71  0000H     2  SAVEAX . . . . .  WORD EXTERNAL(11)          75
 71  0000H     2  SAVECX . . . . .  WORD EXTERNAL(12)          76
 50  006FH    16  SEARCHFIRST. . .  PROCEDURE BYTE STACK=000AH             207
 54  007FH    15  SEARCHNEXT . . .  PROCEDURE BYTE STACK=0008H             219
 46  005FH    16  SETDMA . . . . .  PROCEDURE STACK=000AH            130
 93             SUCCESSFUL . . .  LITERALLY          141  242  252
 25  0000H     1  TBUFF. . . . . .  BYTE ARRAY(1) EXTERNAL(10)             199  209
 67  00BCH    15  TERMINATE. . . .  PROCEDURE STACK=0008H             89  124  171  197  215  258
  5             TRUE . . . . . .  LITERALLY          194
143  0228H    32  UCASE. . . . . .  PROCEDURE BYTE STACK=000CH             155
182  0618H     1  USER . . . . . .  BYTE         205  210
183  060CH     2  VER. . . . . . .  WORD         186  187
  4             VERBDOS. . . . .  LITERALLY          187
  3             VERMASK. . . . .  LITERALLY          187
  2             VERNEEDSOS . . .  LITERALLY          189
```

```
     2                VEROS. . . . . .   LITERALLY
    43  0050H    15   VERSION. . . . . .   PROCEDURE WORD STACK=0008H          136
    92  060EH     1   XFCB . . . . . .   BYTE INITIAL          128  194
```

MODULE INFORMATION:

```
     CODE AREA SIZE      = 0407H   1239D
     CONSTANT AREA SIZE  = 0128H    296D
     VARIABLE AREA SIZE  = 0618H   1563D
     MAXIMUM STACK SIZE  = 001AH     26D
     463 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

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

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

BASE   OFFSET TYPE SYMBOL              BASE   OFFSET TYPE SYMBOL

0000H  03E3H  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  RD
1000H  0080H  PUB  BUFF                1000H  0080H  PUB  TBUFF
1000H  0080H  PUB  BUFFA               1000H  005CH  PUB  FCBA
1000H  0100H  PUB  SAVEAX              1000H  0102H  PUB  SAVEBX
1000H  0104H  PUB  SAVECX              1000H  0106H  PUB  SAVEDX

ERASEQ:  SYMBOLS AND LINES
1000H  0890H  SYM  MEMORY             0000H  0100H  SYM  READCONSOLE
0000H  010FH  SYM  PRINTCHAR           STACK  0004H  SYM  CHAR
0000H  0122H  SYM  CONIN               0000H  0131H  SYM  PRINTBUF
STACK  0004H  SYM  BUFFERADDRESS       0000H  0141H  SYM  CHECKCONSTAT
0000H  0150H  SYM  VERSION             0000H  015FH  SYM  SETDMA
STACK  0004H  SYM  DMA                 0000H  016FH  SYM  SEARCHFIRST
STACK  0004H  SYM  FCBADDRESS          0000H  017FH  SYM  SEARCHNEXT
0000H  018EH  SYM  DELETEFILE          STACK  0004H  SYM  FCBADDRESS
0000H  019EH  SYM  GETUSERCODE         0000H  01ADH  SYM  RETURNERRORS
0000H  01BCH  SYM  TERMINATE           1000H  0108H  SYM  PARSEFN
0000H  01C8H  SYM  PARSE               1000H  010CH  SYM  RETCODE
1000H  010EH  SYM  ERRCODE             1000H  0716H  SYM  XFCB
1000H  0110H  SYM  DIRENTRIES          1000H  0710H  SYM  DIRENTRYADR
1000H  0710H  BAS  DIRENTRY            0000H  0230H  SYM  CRLF
0000H  0241H  SYM  FILL                STACK  0008H  SYM  S
STACK  0006H  SYM  F                   STACK  0004H  SYM  C
STACK  0008H  BAS  A                   0000H  0261H  SYM  ERROR
STACK  0004H  SYM  CODE                0000H  02D1H  SYM  DELETE
STACK  0004H  SYM  FCBADDRESS          STACK  0004H  BAS  FCBV
1000H  0712H  SYM  ERRORCODE           1000H  0717H  SYM  CODE
0000H  0328H  SYM  UCASE               1000H  0718H  SYM  C
0000H  0348H  SYM  GETPASSWD           1000H  0719H  SYM  I
1000H  071AH  SYM  C                   0000H  0352H  SYM  RETRY
0000H  0368H  SYM  NXTCHR              0000H  03C1H  SYM  EXIT
0000H  03C9H  SYM  FILEERR             STACK  0004H  SYM  CODE
1000H  0718H  SYM  I                   1000H  071CH  SYM  J
1000H  071DH  SYM  K                   1000H  071EH  SYM  CODE
1000H  071FH  SYM  RESPONSE            1000H  0720H  SYM  USER
1000H  0721H  SYM  DCNT                1000H  0714H  SYM  VER
1000H  0722H  SYM  LASTDSEGBYTE        0000H  03E3H  SYM  PLMSTART
0000H  0100H  LIN      26              0000H  0103H  LIN      27
0000H  010FH  LIN      28              0000H  010FH  LIN      29
0000H  0112H  LIN      31              0000H  011EH  LIN      32
0000H  0122H  LIN      33              0000H  0125H  LIN      34
0000H  0131H  LIN      35              0000H  0131H  LIN      36
0000H  0134H  LIN      38              0000H  013DH  LIN      39
```

```
0000H  0141H  LIN     40
0000H  0150H  LIN     42
0000H  0153H  LIN     44
0000H  015FH  LIN     46
0000H  016BH  LIN     49
0000H  0172H  LIN     52
0000H  017FH  LIN     54
0000H  018EH  LIN     56
0000H  0191H  LIN     59
0000H  019EH  LIN     61
0000H  01ADH  LIN     63
0000H  01B0H  LIN     65
0000H  01BCH  LIN     67
0000H  01C9H  LIN     69
0000H  01CEH  LIN     74
0000H  01DEH  LIN     76
0000H  01E8H  LIN     79
0000H  01F9H  LIN     81
0000H  0205H  LIN     83
0000H  0211H  LIN     85
0000H  021DH  LIN     87
0000H  0228H  LIN     89
0000H  0230H  LIN     97
0000H  0239H  LIN     99
0000H  0241H  LIN    101
0000H  0250H  LIN    104
0000H  0258H  LIN    106
0000H  0261H  LIN    108
0000H  026AH  LIN    111
0000H  0277H  LIN    113
0000H  0284H  LIN    115
0000H  0297H  LIN    117
0000H  02A4H  LIN    119
0000H  02B1H  LIN    121
0000H  02BEH  LIN    123
0000H  02CDH  LIN    125
0000H  02D4H  LIN    128
0000H  02E2H  LIN    130
0000H  02F1H  LIN    132
0000H  02FFH  LIN    134
0000H  030EH  LIN    137
0000H  031DH  LIN    139
0000H  0328H  LIN    142
0000H  032BH  LIN    145
0000H  033CH  LIN    147
0000H  0348H  LIN    149
0000H  034BH  LIN    152
0000H  035FH  LIN    154
0000H  0375H  LIN    156
0000H  0389H  LIN    159
0000H  0397H  LIN    163
0000H  03AFH  LIN    167
0000H  03B8H  LIN    171
0000H  03C1H  LIN    173
0000H  03C9H  LIN    175
0000H  03CFH  LIN    178
0000H  03DCH  LIN    180
0000H  03E3H  LIN    185
0000H  03ECH  LIN    187
0000H  0403H  LIN    190

0000H  0144H  LIN     41
0000H  0150H  LIN     43
0000H  015FH  LIN     45
0000H  0162H  LIN     48
0000H  016FH  LIN     50
0000H  017FH  LIN     53
0000H  0182H  LIN     55
0000H  018EH  LIN     57
0000H  019EH  LIN     60
0000H  01A1H  LIN     62
0000H  01ADH  LIN     64
0000H  01BAH  LIN     66
0000H  01BFH  LIN     68
0000H  01CBH  LIN     72
0000H  01D8H  LIN     75
0000H  01E4H  LIN     77
0000H  01F2H  LIN     80
0000H  01FEH  LIN     82
0000H  020AH  LIN     84
0000H  0216H  LIN     86
0000H  0224H  LIN     88
0000H  022EH  LIN     91
0000H  0233H  LIN     98
0000H  023FH  LIN    100
0000H  0244H  LIN    103
0000H  0258H  LIN    105
0000H  025DH  LIN    107
0000H  0264H  LIN    110
0000H  0270H  LIN    112
0000H  027DH  LIN    114
0000H  0290H  LIN    116
0000H  029DH  LIN    118
0000H  02AAH  LIN    120
0000H  02B7H  LIN    122
0000H  02CAH  LIN    124
0000H  02D1H  LIN    126
0000H  02DBH  LIN    129
0000H  02E9H  LIN    131
0000H  02F8H  LIN    133
0000H  0306H  LIN    136
0000H  0316H  LIN    138
0000H  0322H  LIN    141
0000H  0328H  LIN    143
0000H  0335H  LIN    146
0000H  0343H  LIN    148
0000H  0348H  LIN    150
0000H  0352H  LIN    153
0000H  036BH  LIN    155
0000H  0382H  LIN    157
0000H  0390H  LIN    161
0000H  039EH  LIN    166
0000H  03B1H  LIN    170
0000H  03BBH  LIN    172
0000H  03C7H  LIN    174
0000H  03CCH  LIN    177
0000H  03D6H  LIN    179
0000H  03DFH  LIN    181
0000H  03E6H  LIN    186
0000H  03FCH  LIN    189
0000H  040CH  LIN    192
```

```
0000H  0413H  LIN    193          0000H  041AH  LIN    194
0000H  0421H  LIN    196          0000H  0428H  LIN    197
0000H  042BH  LIN    199          0000H  0431H  LIN    200
0000H  0437H  LIN    201          0000H  043AH  LIN    202
0000H  0441H  LIN    203          0000H  0452H  LIN    204
0000H  0457H  LIN    205          0000H  045DH  LIN    206
0000H  0460H  LIN    207          0000H  046AH  LIN    208
0000H  0471H  LIN    209          0000H  0485H  LIN    210
0000H  048FH  LIN    212          0000H  049BH  LIN    214
0000H  04A2H  LIN    215          0000H  04A5H  LIN    217
0000H  04C0H  LIN    219          0000H  04C5H  LIN    220
0000H  04C5H  LIN    221          0000H  04CCH  LIN    223
0000H  04D3H  LIN    224          0000H  04D6H  LIN    225
0000H  04E4H  LIN    226          0000H  04EFH  LIN    227
0000H  04F5H  LIN    228          0000H  04F8H  LIN    229
0000H  0507H  LIN    230          0000H  050EH  LIN    231
0000H  0514H  LIN    232          0000H  052FH  LIN    233
0000H  0535H  LIN    234          0000H  053BH  LIN    235
0000H  0541H  LIN    236          0000H  0547H  LIN    237
0000H  054DH  LIN    238          0000H  0553H  LIN    239
0000H  0561H  LIN    241          0000H  0578H  LIN    242
0000H  0586H  LIN    244          0000H  0594H  LIN    245
0000H  059DH  LIN    246          0000H  05A4H  LIN    248
0000H  05ABH  LIN    249          0000H  05AEH  LIN    250
0000H  05B8H  LIN    252          0000H  05BFH  LIN    253
0000H  05C6H  LIN    254          0000H  05C9H  LIN    257
0000H  05D2H  LIN    258          0000H  05D5H  LIN    259
0000H  0100H  LIN    260
```


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

SEGMENT MAP

START	STOP	LENGTH	ALIGN	NAME	CLASS
00000H	005D6H	05D7H	G	CODE	CODE
10000H	10107H	0108H	G	DATS	DATA
10108H	10722H	061BH	W	DATA	DATA
10724H	1075DH	003AH	W	STACK	STACK
1075EH	10885H	0128H	W	CONST	CONST
10890H	10890H	0000H	G	??SEG	
10890H	10890H	0000H	W	MEMORY	MEMORY

GROUP MAP

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