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



```
                $compact
                $title ('REN:  Rename File')
  1             ren:
                do;

                /*
                  Revised:
                     19 Jan  80  by Thomas Rolander
                     31 July 81  by Doug Huskey
                      6 Aug  81  by Danny Horovitz
                     23 Jun  82  by Bill Fitler
                */

                $include (:f1:copyrt.lit)
              =
              =   /*
              =     Copyright (C) 1983
              =     Digital Research
              =     P.O. Box 579
              =     Pacific Grove, CA 93950
              =   */
              =

                $include (:f1: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.
              =   ****/

                $include (:f1:vermpm.lit)
              =
              =   /*   This utility requires MP/M or Concurrent function calls */
              =
              =   /****** commented out for CCP/M-86 :
              =   declare Ver$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$86OS literally '30h';   /* minimal 8OOS version rqd */
=

5   1       declare
            true    literally '0FFh',
            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';

            $include (:f1:proces.lit)
=
=       /*
=           Proces Literals MP/M-8086 II
=       */
=
6   1   =   declare pnamsiz literally '8';
=
7   1   =   declare pd$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$noctls           lit '01000h';
                =
11    1         =     declare pcm$ll            lit '00001h',
                =           pcm$ctls            lit '00002h',
                =           pcm$rout            lit '00004h',
                =           pcm$ctlc            lit '00008h',
                =           pcm$ctlo            lit '00080h',
                =           pcm$rsx             lit '00300h';

                    $include (:f1:uda.lit)
                =
                =   /* MP/M-86 II  User Data Area format - August 8, 1981 */
                =
12    1         =   declare uda$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)';
                =


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


13    1            mon1:
                     procedure (func,info) external;
14    2                declare func byte;
15    2                declare info address;
16    2              end mon1;

17    1            mon2:
                     procedure (func,info) byte external;
18    2                declare func byte;
19    2                declare info address;
```

```
20   2          end mon2;

21   1        mon3:
              procedure (func,info) address external;
22   2          declare func byte;
23   2          declare info address;
24   2          end mon3;

25   1        mon4:
              procedure (func,info) pointer external;
26   2          declare func byte;
27   2          declare info address;
28   2          end mon4;


29   1        declare cmdrv     byte    external;   /* command drive      */
30   1        declare fcb (1)   byte    external;   /* 1st default fcb    */
31   1        declare fcb16 (1) byte    external;   /* 2nd default fcb    */
32   1        declare pass0     address external;   /* 1st password ptr   */
33   1        declare len0      byte    external;   /* 1st passwd length  */
34   1        declare pass1     address external;   /* 2nd password ptr   */
35   1        declare len1      byte    external;   /* 2nd passwd length  */
36   1        declare tbuff (1) byte    external;   /* default dma buffer */

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

37   1        read$console:
              procedure byte;
38   2          return mon2 (1,0);
39   2          end read$console;

40   1        conin:
              procedure byte;
41   2          return mon2(6,0fdh);
42   2          end conin;

43   1        printchar:
              procedure (char);
44   2          declare char byte;
45   2          call mon1 (2,char);
46   2          end printchar;

47   1        check$con$stat:
              procedure byte;
48   2          return mon2(11,0);
49   2          end check$con$stat;

50   1        print$buf:
              procedure (buffer$address);
51   2          declare buffer$address address;
52   2          call mon1 (9,buffer$address);
```

```
53   2          end print$buf;

54   1        version: procedure address;
                /* returns current cp/m version # */
55   2          return mon3(12,0);
56   2          end version;

57   1        search$first:
                procedure (fcb$address) byte;
58   2            declare fcb$address address;
59   2            return mon2 (17,fcb$address);
60   2          end search$first;

61   1        search$next:
                procedure byte;
62   2            return mon2 (18,0);
63   2          end search$next;

64   1        delete$file:
                procedure (fcb$address);
65   2            declare fcb$address address;
66   2            call mon1 (19,fcb$address);
67   2          end delete$file;

68   1        rename$file:
                procedure (fcb$address) address;
69   2            declare fcb$address address;
70   2            return mon3 (23,fcb$address);
71   2          end rename$file;

72   1        setdma: procedure(dma);
73   2          declare dma address;
74   2          call mon1(26,dma);
75   2          end setdma;

                /* Off => return BDOS errors */
76   1        return$errors:
                procedure(mode);
77   2          declare mode byte;
78   2            call mon1 (45,mode);
79   2          end return$errors;

80   1        terminate:
                procedure;
81   2            call mon1 (143,0);
82   2          end terminate;

83   1        declare
                parse$fn structure (
                  buff$adr  address,
                  fcb$adr   address);

84   1        parse: procedure address;
85   2          return mon3(152,.parse$fn);
86   2          end parse;

87   1        declare
```

```
                pd$pointer      pointer,
                pd              based pd$pointer pd$structure;
 88   1       declare
                uda$pointer pointer,
                uda$ptr structure (
                  offset word,
                  segment word) at (@uda$pointer),
                uda based uda$pointer uda$structure;

 89   1       get$uda: procedure;
 90   2         pd$pointer = mon4(156,0);
 91   2         uda$ptr.segment = pd.uda;
 92   2         uda$ptr.offset = 0;
 93   2         end get$uda;


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

               /* Note: there are three fcbs used by
                  this program:

                      1) new$fcb: the new file name
                         (this can be a wildcard if it
                         has the same pattern of question
                         marks as the old file name)
                         Any question marks are replaced
                         with the corresponding filename
                         character in the old$fcb before
                         doing the rename function.

                      2) cur$fcb: the file to be renamed
                         specified in the rename command.
                         (any question marks must correspond
                         to question marks in new$fcb).

                      3) old$fcb: a fcb in the directory
                         matching the cur$fcb and used in
                         the bdos rename function.  This
                         cannot contain any question marks.
               */

 94   1       declare successful lit '0FFh';
 95   1       declare failed       (*) byte data(cr,lf,'Not renamed: $'),
                      read$only    (*) byte data(cr,lf,'Drive Read Only$'),
                      bad$wildcard (*) byte data('Invalid Wildcard$');
 96   1       declare passwd (8) byte;
 97   1       declare
                  new$fcb$adr address,       /* new name */
                  new$fcb based  new$fcb$adr (32) byte;
 98   1       declare cur$fcb (33) byte;      /* current fcb (old name) */

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

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


                                /* upper case character from console */
 99   1       crlf:   proc;
100   2           call printchar(cr);
101   2           call printchar(lf);
102   2           end crlf;
                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                                /* fill string a s for c bytes with f */
103   1       fill:   proc(s,f,c);
104   2           dcl s addr,
                        (f,c) byte,
                        a based s byte;

105   2           do while (c:=c-1)<>255;
106   3           a = f;
107   3           s = s+1;
108   3           end;
109   2           end fill;
                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                                /* error message routine */
110   1       error:   proc(code);
111   2          declare
                     code byte;

112   2           if code = 0 then do;
114   3               call print$buf(.('No such file to rename$'));
115   3               call terminate;
116   3               end;
117   2           if code=1 then do;
119   3               call print$buf(.(cr,lf,'BDOS Bad Sector$'));
120   3               call terminate;
121   3               end;
122   2           if code=2 then do;
124   3               call print$buf(.read$only);
125   3               call terminate;
126   3               end;
127   2           if code = 3 then
128   2               call print$buf(.read$only(8));
129   2           if code = 5 then
130   2               call print$buf(.('Currently Opened$'));
131   2           if code = 7 then
132   2               call print$buf(.('Password Error$'));
133   2           if code = 8 then
134   2               call print$buf(.('already exists$'));
135   2           if code = 9 then do;
137   3               call print$buf(.bad$wildcard);
138   3               call terminate;
139   3               end;
140   2           end error;
```

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


                        /* print file name */
141   1     print$file: procedure(fcbp);
142   2             declare k byte;
143   2             declare typ lit '9';          /* file type */
144   2             declare fnam lit '11';         /* file type */
145   2             declare
                          fcbp   addr,
                          fcbv   based fcbp (32) byte;

146   2             do k = 1 to fnam;
147   3             if k = typ then
148   3               call printchar('.');
149   3               call printchar(fcbv(k) and 7fh);
150   3               end;
151   2             end print$file;
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


  /* try to rename fcb at old$fcb$adr to name at new$fcb$adr
             return error code if unsuccessful */
152   1       rename:
            procedure(old$fcb$adr) byte;
153   2         declare
                old$fcb$adr address,
                old$fcb based  old$fcb$adr (32) byte,
                error$code  address,
                code        byte;

154   2         call move (16,new$fcb$adr,old$fcb$adr+16);
155   2         call setdma(.passwd);                    /* password    */
156   2         call return$errors(0FFh);          /* return bdos errors */
157   2         error$code = rename$file (old$fcb$adr);
158   2         call return$errors(0);             /* normal error mode  */
159   2         if low(error$code) = 0FFh then do;
161   3             code = high(error$code);
162   3             if code < 3 then
163   3                 call error(code);
164   3             return code;
165   3             end;
166   2         return successful;
167   2         end rename;
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* upper case character from console */
168   1     ucase:   proc(c) byte;
169   2         dcl c byte;

170   2         if c >= 'a' then
171   2            if c < '{' then
172   2               return(c-20h);
173   2         return c;
174   2         end ucase;
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

```
                        /* get password and place at fcb + 16 */
175   1     getpasswd:  proc;
176   2         dcl (i,c) byte;

177   2         call crlf;
178   2         call print$buf(.('Password ? ','$'));
179   2     retry:
              call fill(.passwd,' ',8);
180   2             do i = 0 to 7;
181   3     nxtchr:
                  if (c:=ucase(conin)) >= ' ' then
182   3                 passwd(i)=c;
183   3             if c = cr then do;
185   4                 call crlf;
186   4                 goto exit;
187   4                 end;
188   3             if c = ctrlx then
189   3                 goto retry;
190   3             if c = bksp then do;
192   4                 if i<1 then
193   4                     goto retry;
194   4                 else do;
195   5                     passwd(i:=i-1)=' ';
196   5                     goto nxtchr;
197   5                     end;
198   4                 end;
199   3             if c = ctrlc then
200   3                 call terminate;
201   3             end;
202   2     exit:
              c = check$con$stat;
203   2         end getpasswd;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* check for wildcard in rename command */
204   1     wildcard:  proc byte;
205   2         dcl (i,wild) byte;

206   2         wild = false;
207   2             do i=1 to 11;
208   3             if cur$fcb(i) = '?' then
209   3                 if new$fcb(i) <> '?' then do;
211   4                     call print$buf(.failed);
212   4                     call print$buf(.bad$wildcard);
213   4                     call terminate;
214   4                     end;
                      else
215   3                     wild = true;
216   3             end;
217   2         return wild;
218   2         end wildcard;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

```
                        /* set up new name for rename function */
219   1     set$new$fcb:  proc(old$fcb$adr);
220   2         dcl old$fcb$adr address,
                  old$fcb based old$fcb$adr (32) byte;
221   2         dcl i byte;

222   2         old$fcb(0) = cur$fcb(0);    /* set up drive */
223   2         do i=1 to 11;
224   3         if cur$fcb(i) = '?' then
225   3             new$fcb(i) = old$fcb(i);
226   3         end;
227   2         end set$new$fcb;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* try deleting files one at a time */
228   1       single$file:
            procedure;
229   2       declare (code,dcnt,savsearchl) byte;
230   2       declare (old$fcb$adr,savdcnt,savsearcha) addr;
231   2       declare old$fcb based old$fcb$adr (32) byte;

232   2       file$err: procedure(fcba);
233   3           dcl fcba address;
234   3           call print$buf(.failed);
235   3           call print$file(fcba);
236   3           call printchar(' ');
237   3           call error(code);
238   3           end file$err;

239   2       call setdma(.tbuff);
240   2       if (dcnt:=search$first(.cur$fcb)) = 0ffh then
241   2           call error(0);

242   2           do while dcnt <> 0ffh;
243   3           old$fcb$adr = shl(dcnt,5) + .tbuff;
244   3           savdcnt = uda.dcnt;
245   3           savsearcha = uda.searcha;
246   3           savsearchl = uda.searchl;
247   3           call set$new$fcb(old$fcb$adr);
248   3           if (code:=rename(old$fcb$adr)) = 8 then do;
250   4               call file$err(new$fcb$adr);
251   4               call print$buf(.(', delete (Y/N)?$'));
252   4               if ucase(read$console) = 'Y' then do;
254   5                   call delete$file(new$fcb$adr);
255   5                   code = rename(old$fcb$adr);
256   5                   end;
                    else
257   4                   go to next;
258   4               end;
259   3           if code = 7 then do;
261   4               call file$err(old$fcb$adr);
262   4               call getpasswd;
263   4               code = rename(old$fcb$adr);
264   4               end;
265   3           if code <> successful then
266   3               call file$err(old$fcb$adr);
```

```
267   3              else do;
268   4                  call crlf;
269   4                  call print$file(new$fcb$adr);
270   4                  call printchar('=');
271   4                  call print$file(old$fcb$adr);
272   4                  end;
273   3      next:
                     call setdma(.tbuff);
274   3              uda.dcnt = savdcnt;
275   3              uda.searcha = savsearcha;
276   3              uda.searchl = savsearchl;
277   3              dcnt = search$next;
278   3              end;
279   2          end single$file;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                                /* invalid rename command */
280   1      bad$entry:   proc;

281   2          call print$buf(.failed);
282   2          call print$buf(.('Invalid File','$'));
283   2          call terminate;
284   2          end bad$entry;


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

285   1      declare ver address;

286   1      declare last$dseg$byte byte
               initial (0);

287   1      plm$start:
               procedure public;
288   2          ver = version;
289   2          if low(ver) < Ver$BDOS  or  (high(ver) and Ver$Mask) = 0 then
290   2              call print$buf (.(Ver$Needs$OS,'$'));
291   2          else do;
292   3              call get$uda;
293   3              parse$fn.buff$adr = .tbuff(1);
294   3              new$fcb$adr, parse$fn.fcb$adr = .fcb;
295   3              if (parse$fn.fcb$adr:=parse) <> 0FFFFh then do; /* old file */
297   4                  parse$fn.buff$adr = parse$fn.fcb$adr + 1; /* skip delim */
298   4                  parse$fn.fcb$adr  = .cur$fcb;
299   4                  parse$fn.fcb$adr = parse;                /* new file */
300   4                  call move (8,.cur$fcb+16,.passwd);       /* password */
301   4                  end;
302   3              if parse$fn.fcb$adr = 0ffffh then
303   3                  call bad$entry;
304   3              if fcb(0) <> 0 then
305   3                  if cur$fcb(0) <> 0 then do;
307   4                      if fcb(0) <> cur$fcb(0) then
```

```
308   4                       call bdos$entry;
309   4                   end;
                      else
310   3                   cur$fcb(0) = new$fcb(0);        /* set drive */
311   3               if wildcard then
312   3                   call singlefile;
313   3               else if rename(.cur$fcb) <> successful then
314   3                   call singlefile;
                      end;
316   2           call mon1(0,0);
317   2           end plm$start;

318   1       end ren;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES											
104	0000H	1	A.	BYTE BASED(S)	106									
87	0021H	1	ABORT.	BYTE MEMBER(PD)										
5			ADDR	LITERALLY	104	145	230							
280	048DH	22	BADENTRY	PROCEDURE STACK=000EH					303	308				
95	0022H	17	BADWILDCARD. . . .	BYTE ARRAY(17) DATA				137	212					
5			BKSP	LITERALLY	190									
83	0000H	2	BUFFADR.	WORD MEMBER(PARSEFN)				293	297					
50	0004H	2	BUFFERADDRESS. . .	WORD PARAMETER AUTOMATIC					51	52				
168	0004H	1	C.	BYTE PARAMETER AUTOMATIC					169	170	171	172	173	
103	0004H	1	C.	BYTE PARAMETER AUTOMATIC					104	105				
176	0044H	1	C.	BYTE	181	182	183	188	190	199	202			
43	0004H	1	CHAR	BYTE PARAMETER AUTOMATIC					44	45				
47	0033H	15	CHECKCONSTAT . . .	PROCEDURE BYTE STACK=0008H						202				
29	0000H	1	CMDRV.	BYTE EXTERNAL(4)										
87	0020H	1	CNS.	BYTE MEMBER(PD)										
229	0048H	1	CODE	BYTE	237	248	255	259	263	265				
153	0042H	1	CODE	BYTE	161	162	163	164						
110	0004H	1	CODE	BYTE PARAMETER AUTOMATIC					111	112	117	122	127	129	131	133
			135											
40	0011H	15	CONIN.	PROCEDURE BYTE STACK=0008H						181				
87	0022H	2	CONMODE.	WORD MEMBER(PD)										
5			CR	LITERALLY	95	100	119		183					
99	0109H	17	CRLF	PROCEDURE STACK=000EH					177	185	268			
5			CTRLC.	LITERALLY	199									
5			CTRLX.	LITERALLY	188									
98	0020H	33	CURFCB	BYTE ARRAY(33)			208	222	224	240	298	300	305	307	310	313
88	000EH	2	DBLK	WORD MEMBER(UDA)										
5			DCL.	LITERALLY										
229	0049H	1	DCNT	BYTE	240	242	243	277						
88	000CH	2	DCNT	WORD MEMBER(UDA)				244	274					
64	0080H	16	DELETEFILE	PROCEDURE STACK=000AH					254					
88	0012H	8	DFPASSWORD	BYTE ARRAY(8) MEMBER(UDA)										
72	0004H	2	DMA.	WORD PARAMETER AUTOMATIC					73	74				
88	0002H	2	DMAOFST.	WORD MEMBER(UDA)										
88	0004H	2	DMASEG	WORD MEMBER(UDA)										
88	0000H	2	DPARAM	WORD MEMBER(UDA)										
87	0012H	1	DSK.	BYTE MEMBER(PD)										
87	0018H	2	DVRACT	WORD MEMBER(PD)										
110	013AH	123	ERROR.	PROCEDURE STACK=0010H					163	237	241			
153	000EH	2	ERRORCODE.	WORD	157	159	161							
88	0010H	1	ERRORMODE.	BYTE MEMBER(UDA)										
202	02DEH		EXIT	LABEL	186									
103	0006H	1	F.	BYTE PARAMETER AUTOMATIC					104	106				
95	0000H	16	FAILED	BYTE ARRAY(16) DATA				211	234	281				
5			FALSE.	LITERALLY	206									
30	0000H	1	FCB.	BYTE ARRAY(1) EXTERNAL(5)					294	304	307			
31	0000H	1	FCB16.	BYTE ARRAY(1) EXTERNAL(6)										
232	0004H	2	FCBA	WORD PARAMETER AUTOMATIC					233	235				
68	0004H	2	FCBADDRESS	WORD PARAMETER AUTOMATIC					69	70				

```
 64  0004H     2  FCBADDRESS . . . .  WORD PARAMETER AUTOMATIC          65   66
 57  0004H     2  FCBADDRESS . . . .  WORD PARAMETER AUTOMATIC          58   59
 83  0002H     2  FCBADR . . . . . .  WORD MEMBER(PARSEFN)       294   295  297  298  299  302
141  0004H     2  FCBP . . . . . .  WORD PARAMETER AUTOMATIC          145  149
145  0000H    32  FCBV . . . . . .  BYTE BASED(FCBP) ARRAY(32)            149
232  046CH    33  FILEERR. . . . . .  PROCEDURE STACK=0016H          250  261  266
103  011AH    32  FILL . . . . . .  PROCEDURE STACK=0008H          179
 87  0006H     2  FLAG . . . . . .  WORD MEMBER(PD)
144              FNAM . . . . . .  LITERALLY       146
  5              FOREVER. . . . . .  LITERALLY
 17  0000H     1  FUNC . . . . . .  BYTE PARAMETER          18
 13  0000H     1  FUNC . . . . . .  BYTE PARAMETER          14
 25  0000H     1  FUNC . . . . . .  BYTE PARAMETER          26
 88  0006H     1  FUNC . . . . . .  BYTE MEMBER(UDA)
 21  0000H     1  FUNC . . . . . .  BYTE PARAMETER          22
175  0259H   141  GETPASSWD. . . . .  PROCEDURE STACK=0012H          262
 89  00E1H    40  GETUDA . . . . . .  PROCEDURE STACK=0008H          292
              HIGH . . . . . .  BUILTIN          161  289
176  0043H     1  I. . . . . .  BYTE        180  182  192  195
205  0045H     1  I. . . . . .  BYTE        207  208  209
221  0047H     1  I. . . . . .  BYTE        223  224  225
 13  0000H     2  INFO . . . . . .  WORD PARAMETER          15
 25  0000H     2  INFO . . . . . .  WORD PARAMETER          27
 17  0000H     2  INFO . . . . . .  WORD PARAMETER          19
 21  0000H     2  INFO . . . . . .  WORD PARAMETER          23
142  0041H     1  K. . . . . .  BYTE        146  147  149
286  0048H     1  LASTDSEGBYTE . . .  BYTE INITIAL
 87  0014H     1  LDSK . . . . . .  BYTE MEMBER(PD)
 33  0000H     1  LEN0 . . . . . .  BYTE EXTERNAL(8)
 35  0000H     1  LEN1 . . . . . .  BYTE EXTERNAL(10)
  5              LF . . . . . .  LITERALLY         95  101  119
 87  0000H     2  LINK . . . . . .  WORD MEMBER(PD)
  5              LIT. . . . . .  LITERALLY          9   10   11   12   94  143  144
              LOW. . . . . .  BUILTIN          159  289
 87  0024H     1  LST. . . . . .  BYTE MEMBER(PD)
 87  0015H     1  LUSER. . . . . .  BYTE MEMBER(PD)
 87  0016H     2  MEM. . . . . .  WORD MEMBER(PD)
 76  0004H     1  MODE . . . . . .  BYTE PARAMETER AUTOMATIC          77   78
 13  0000H        MON1 . . . . . .  PROCEDURE EXTERNAL(0) STACK=0000H          45   52   66   74   78   81
                  316
 17  0000H        MON2 . . . . . .  PROCEDURE BYTE EXTERNAL(1) STACK=0000H          38   41   48   59   62
 21  0000H        MON3 . . . . . .  PROCEDURE WORD EXTERNAL(2) STACK=0000H          55   70   85
 25  0000H        MON4 . . . . . .  PROCEDURE POINTER EXTERNAL(3) STACK=0000H          90
              MOVE . . . . . .  BUILTIN          154  300
 88  0011H     1  MULTCNT. . . . . .  BYTE MEMBER(UDA)
 87  0008H     8  NAME . . . . . .  BYTE ARRAY(8) MEMBER(PD)
 87  001DH     1  NET. . . . . .  BYTE MEMBER(PD)
 97  0000H    32  NEWFCB . . . . . .  BYTE BASED(NEWFCBADR) ARRAY(32)          209  225  310
 97  000CH     2  NEWFCBADR. . . . . .  WORD          97  154  209  225  250  254  269  294  310
273  0441H        NEXT . . . . . .  LABEL       257
181  027FH        NXTCHR . . . . . .  LABEL       196
 88  0000H     2  OFFSET . . . . . .  WORD MEMBER(UDAPTR)         92
231  0000H    32  OLDFCB . . . . . .  BYTE BASED(OLDFCBADR) ARRAY(32)
153  0000H    32  OLDFCB . . . . . .  BYTE BASED(OLDFCBADR) ARRAY(32)
220  0000H    32  OLDFCB . . . . . .  BYTE BASED(OLDFCBADR) ARRAY(32)          222  225
230  0010H     2  OLDFCBADR. . . . .  WORD          231  243  247  248  255  261  263  266  271
219  0004H     2  OLDFCBADR. . . . .  WORD PARAMETER AUTOMATIC          220  222  225
```

```
152  0004H     2  OLDFCBADR. . . . .  WORD PARAMETER AUTOMATIC          153  154  157
 87  001CH     1  ORG. . . . . .      BYTE MEMBER(PD)
 87  001EH     2  PARENT . . . . .    WORD MEMBER(PD)
 84  00D2H    15  PARSE. . . . . .    PROCEDURE WORD STACK=0003H              295  299
 83  0000H     4  PARSEFN. . . . .    STRUCTURE          85  293  294  295  297  298  299  302
 32  0000H     2  PASS0. . . . . .    WORD EXTERNAL(7)
 34  0000H     2  PASS1. . . . . .    WORD EXTERNAL(9)
 96  0018H     8  PASSWD . . . . .    BYTE ARRAY(8)          155  179  182  195  300
 11             PCM11. . . . . .    LITERALLY
 11             PCMCTLC. . . . . .  LITERALLY
 11             PCMCTLO. . . . . .  LITERALLY
 11             PCMCTLS. . . . . .  LITERALLY
 11             PCMROUT. . . . . .  LITERALLY
 11             PCMRSX . . . . . .  LITERALLY
 87  0000H    48  PD . . . . . .      STRUCTURE BASED(PDPOINTER)              91
 88  001AH     1  PDCNT. . . . . .    BYTE MEMBER(PD)
  7             PDHDR. . . . . .    LITERALLY          87
 87  0004H     4  PDPOINTER. . . . .  POINTER            87   90   91
  8             PDSTRUCTURE. . . .  LITERALLY          87
 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
287  04A3H   179  PLMSTART . . . . .  PROCEDURE PUBLIC STACK=001EH
  6             PNAMSIZ. . . . . .  LITERALLY
 87  002CH     2  PRET . . . . . .    WORD MEMBER(PD)
 50  0042H    16  PRINTBUF . . . . .  PROCEDURE STACK=000AH          114  119  124  128  130  132  134  137
                                       178  211  212  234  251  281  282  290
 43  0020H    19  PRINTCHAR. . . . .  PROCEDURE STACK=000AH          100  101  148  149  236  270
141  0185H    56  PRINTFILE. . . . .  PROCEDURE STACK=0010H          235  269  271
 87  0005H     1  PRIOR. . . . . .    BYTE MEMBER(PD)
  5             PROC . . . . . .    LITERALLY          99  103  110  168  175  204  219  280
  9             PSCTOWAIT. . . . .  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
 37  0002H    15  READCONSOLE. . . .  PROCEDURE BYTE STACK=0008H          252
 95  0010H    18  READONLY . . . . .  BYTE ARRAY(18) DATA       124  128
  1  0002H        REN. . . . . . .    PROCEDURE STACK=0000H
152  01EDH    83  RENAME . . . . . .  PROCEDURE BYTE STACK=0016H          248  255  263  313
 68  0090H    16  RENAMEFILE . . . .  PROCEDURE WORD STACK=000AH          157
 87  0028H     4  RESERVD. . . . . .  BYTE ARRAY(4) MEMBER(PD)
```

```
 179  0266H        RETRY. . . . . .   LABEL       189  193
  76  0080H    19  RETURNERRORS . . .  PROCEDURE STACK=0007H          156  158
 103  0008H     2  S. . . . . .       WORD PARAMETER AUTOMATIC       104  106  107
 230  0012H     2  SAVDCNT. . . . .   WORD        244  274
 230  0014H     2  SAVSEARCHA . . . .  WORD        245  275
 229  004AH     1  SAVSEARCHL . . . .  BYTE        246  276
  87  002EH     2  SCRATCH. . . . . .  WORD MEMBER(PD)
  88  0008H     2  SEARCHA. . . . . .  WORD MEMBER(UDA)          245  275
  88  000AH     2  SEARCHABASE. . . .  WORD MEMBER(UDA)
  57  0061H    16  SEARCHFIRST. . . .  PROCEDURE BYTE STACK=000AH       240
  88  0007H     1  SEARCHL. . . . . .  BYTE MEMBER(UDA)          246  276
  61  0071H    15  SEARCHNEXT . . . .  PROCEDURE BYTE STACK=0008H       277
  88  0002H     2  SEGMENT. . . . . .  WORD MEMBER(UDAPTR)        91
  72  00A0H    16  SETDMA . . . . . .  PROCEDURE STACK=000AH          155  239  273
 219  0333H    57  SETNEWFCB. . . . .  PROCEDURE STACK=0004H          247
  87  0025H     1  SF3. . . . . .     BYTE MEMBER(PD)
  87  0026H     1  SF4. . . . . .     BYTE MEMBER(PD)
  87  0027H     1  SF5. . . . . .     BYTE MEMBER(PD)
              SHL. . . . . .     BUILTIN         243
 228  036CH   256  SINGLEFILE . . . .  PROCEDURE STACK=001AH          312  314
  87  0004H     1  STAT . . . . . .   BYTE MEMBER(PD)
  94             SUCCESSFUL . . . .  LITERALLY       166  265  313
  36  0000H     1  TBUFF. . . . . .   BYTE ARRAY(1) EXTERNAL(11)      239  243  273  293
  80  00C3H    15  TERMINATE. . . . .  PROCEDURE STACK=0008H          115  120  125  138  200  213  283
  87  0002H     2  THREAD . . . . . .  WORD MEMBER(PD)
   5             TRUE . . . . . .   LITERALLY       215
 143             TYP. . . . . .     LITERALLY       147
 168  0240H    25  UCASE. . . . . .   PROCEDURE BYTE STACK=0004H       181  252
  88  0000H    27  UDA. . . . . .     STRUCTURE BASED(UDAPOINTER)     244  245  246  274  275  276
  87  0010H     2  UDA. . . . . .     WORD MEMBER(PD)       91
  88  0008H     4  UDAPOINTER . . . .  POINTER          88  244  245  246  274  275  276
  88  0008H     4  UDAPTR . . . . . .  STRUCTURE AT         91   92
  12             UDASTRUCTURE . . .  LITERALLY        88
  87  0013H     1  USER . . . . . .   BYTE MEMBER(PD)
 285  0016H     2  VER. . . . . .     WORD        288  289
   4             VERBDOS. . . . . .  LITERALLY       289
   3             VERMASK. . . . . .  LITERALLY       289
   2             VERNEEDSDS . . . .  LITERALLY       290
   2             VERDS. . . . . .    LITERALLY
  54  0052H    15  VERSION. . . . . .  PROCEDURE WORD STACK=0008H       288
  87  001AH     2  WAIT . . . . . .   WORD MEMBER(PD)
 205  0046H     1  WILD . . . . . .   BYTE        206  215  217
 204  02E6H    77  WILDCARD . . . . .  PROCEDURE BYTE STACK=000EH       311
```


MODULE INFORMATION:

```
  CODE AREA SIZE     = 0556H   1366D
  CONSTANT AREA SIZE = 0000H      0D
  VARIABLE AREA SIZE = 004CH     76D
  MAXIMUM STACK SIZE = 001EH     30D
  635 LINES READ
  0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

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

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

BASE   OFFSET TYPE SYMBOL              BASE   OFFSET TYPE SYMBOL

0000H  05A3H  PUB  PLMSTART            0000H  0050H  PUB  XDOS
0000H  005DH  PUB  MON1                0000H  005DH  PUB  MON2
0000H  005DH  PUB  MON3                0000H  005DH  PUB  MON4
1000H  0004H  PUB  BDISK               1000H  0006H  PUB  MAXB
1000H  0050H  PUB  CMDRV               1000H  0051H  PUB  PASS0
1000H  0053H  PUB  LEN0                1000H  0054H  PUB  PASS1
1000H  0056H  PUB  LEN1                1000H  005CH  PUB  FCB
1000H  006CH  PUB  FCB16               1000H  007CH  PUB  CR
1000H  007DH  PUB  RR                  1000H  007FH  PUB  RO
1000H  0080H  PUB  BUFF                1000H  0080H  PUB  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

REN:  SYMBOLS AND LINES
1000H  0270H  SYM  MEMORY              0000H  0102H  SYM  READCONSOLE
0000H  0111H  SYM  CONIN               0000H  0120H  SYM  PRINTCHAR
STACK  0004H  SYM  CHAR                0000H  0133H  SYM  CHECKCONSTAT
0000H  0142H  SYM  PRINTBUF            STACK  0004H  SYM  BUFFERADDRESS
0000H  0152H  SYM  VERSION             0000H  0161H  SYM  SEARCHFIRST
STACK  0004H  SYM  FCBADDRESS          0000H  0171H  SYM  SEARCHNEXT
0000H  0180H  SYM  DELETEFILE          STACK  0004H  SYM  FCBADDRESS
0000H  0190H  SYM  RENAMEFILE          STACK  0004H  SYM  FCBADDRESS
0000H  01A0H  SYM  SETDMA              STACK  0004H  SYM  DMA
0000H  01B0H  SYM  RETURNERRORS        STACK  0004H  SYM  MODE
0000H  01C3H  SYM  TERMINATE           1000H  0108H  SYM  PARSEFN
0000H  01D2H  SYM  PARSE               1000H  010CH  SYM  PDPOINTER
1000H  010CH  BAS  PD                  1000H  0110H  SYM  UDAPOINTER
1000H  0110H  SYM  UDAPTR              1000H  0110H  BAS  UDA
0000H  01E1H  SYM  GETUDA              1000H  0192H  SYM  FAILED
1000H  01A2H  SYM  READONLY            1000H  01B4H  SYM  BADWILDCARD
1000H  0120H  SYM  PASSWD              1000H  0114H  SYM  NEWFCBADR
1000H  0114H  BAS  NEWFCB              1000H  0128H  SYM  CURFCB
0000H  0209H  SYM  CRLF                0000H  021AH  SYM  FILL
STACK  0008H  SYM  S                   STACK  0006H  SYM  F
STACK  0004H  SYM  C                   STACK  0008H  BAS  A
0000H  023AH  SYM  ERROR               STACK  0004H  SYM  CODE
0000H  0285H  SYM  PRINTFILE           STACK  0004H  SYM  FCBP
1000H  0149H  SYM  K                   STACK  0004H  BAS  FCBV
0000H  02EDH  SYM  RENAME              STACK  0004H  SYM  OLDFCBADR
STACK  0004H  BAS  OLDFCB              1000H  0116H  SYM  ERRORCODE
1000H  014AH  SYM  CODE                0000H  0340H  SYM  UCASE
STACK  0004H  SYM  C                   0000H  0359H  SYM  GETPASSWD
1000H  014BH  SYM  T                   1000H  014CH  SYM  C
0000H  0366H  SYM  RETRY               0000H  037FH  SYM  NXTCHR
0000H  03DEH  SYM  EXIT                0000H  03F6H  SYM  WILDCARD
1000H  014DH  SYM  I                   1000H  014EH  SYM  WILD
0000H  0433H  SYM  SETNEWFCB           STACK  0004H  SYM  OLDFCBADR
STACK  0004H  BAS  OLDFCB              1000H  014FH  SYM  T
0000H  046CH  SYM  SINGLEFILE          1000H  0150H  SYM  CODE
1000H  0151H  SYM  DCNT                1000H  0152H  SYM  SAVSEARCHL
```

```
1000H  0118H  SYM  OLDFCBADR
1000H  011CH  SYM  SAVSEARCHA
0000H  056CH  SYM  FILEERR
0000H  0541H  SYM  NEXT
1000H  011EH  SYM  VER
0000H  05A3H  SYM  PEMSTART
0000H  0105H  LIN     38
0000H  0111H  LIN     40
0000H  0120H  LIN     42
0000H  0123H  LIN     45
0000H  0133H  LIN     47
0000H  0142H  LIN     49
0000H  0145H  LIN     52
0000H  0152H  LIN     54
0000H  0161H  LIN     56
0000H  0164H  LIN     59
0000H  0171H  LIN     61
0000H  0180H  LIN     63
0000H  0183H  LIN     66
0000H  0190H  LIN     68
0000H  01A0H  LIN     71
0000H  01A3H  LIN     74
0000H  01B0H  LIN     76
0000H  01BFH  LIN     79
0000H  01C6H  LIN     81
0000H  01D2H  LIN     84
0000H  01E1H  LIN     86
0000H  01E4H  LIN     90
0000H  0201H  LIN     92
0000H  0209H  LIN     99
0000H  0212H  LIN    101
0000H  021AH  LIN    103
0000H  0229H  LIN    106
0000H  0234H  LIN    108
0000H  023AH  LIN    110
0000H  0243H  LIN    114
0000H  024DH  LIN    117
0000H  025AH  LIN    120
0000H  0263H  LIN    124
0000H  0260H  LIN    127
0000H  027AH  LIN    129
0000H  0287H  LIN    131
0000H  0294H  LIN    133
0000H  02A1H  LIN    135
0000H  02AEH  LIN    138
0000H  0285H  LIN    141
0000H  02C4H  LIN    147
0000H  02D1H  LIN    149
0000H  02E9H  LIN    151
0000H  02F0H  LIN    154
0000H  0309H  LIN    156
0000H  0318H  LIN    158
0000H  0325H  LIN    161
0000H  0331H  LIN    163
0000H  033AH  LIN    166
0000H  0340H  LIN    168
0000H  034AH  LIN    171
0000H  0352H  LIN    173
0000H  0359H  LIN    175
0000H  035FH  LIN    178

1000H  011AH  SYM  SAVDCNT
1000H  0118H  SYM  OLDFCB
STACK  0004H  SYM  FCBA
0000H  0580H  SYM  BADENTRY
1000H  0153H  SYM  LASTDSECBYTE
0000H  0102H  LIN     37
0000H  0111H  LIN     39
0000H  0114H  LIN     41
0000H  0120H  LIN     43
0000H  012FH  LIN     46
0000H  0136H  LIN     48
0000H  0142H  LIN     50
0000H  014EH  LIN     53
0000H  0155H  LIN     55
0000H  0161H  LIN     57
0000H  0171H  LIN     60
0000H  0174H  LIN     62
0000H  0180H  LIN     64
0000H  018CH  LIN     67
0000H  0193H  LIN     70
0000H  01A0H  LIN     72
0000H  01ACH  LIN     75
0000H  01B3H  LIN     78
0000H  01C3H  LIN     80
0000H  01D0H  LIN     82
0000H  01D5H  LIN     85
0000H  01E1H  LIN     89
0000H  01F6H  LIN     91
0000H  0207H  LIN     93
0000H  020CH  LIN    100
0000H  0218H  LIN    102
0000H  021DH  LIN    105
0000H  0231H  LIN    107
0000H  0236H  LIN    109
0000H  023DH  LIN    112
0000H  024AH  LIN    115
0000H  0253H  LIN    119
0000H  025DH  LIN    122
0000H  026AH  LIN    125
0000H  0273H  LIN    128
0000H  0280H  LIN    130
0000H  028DH  LIN    132
0000H  029AH  LIN    134
0000H  02A7H  LIN    137
0000H  02B1H  LIN    140
0000H  02B8H  LIN    146
0000H  02CBH  LIN    148
0000H  02E3H  LIN    150
0000H  02EDH  LIN    152
0000H  0302H  LIN    155
0000H  030FH  LIN    157
0000H  031EH  LIN    159
0000H  032DH  LIN    162
0000H  0335H  LIN    164
0000H  0340H  LIN    167
0000H  0343H  LIN    170
0000H  034EH  LIN    172
0000H  0359H  LIN    174
0000H  035CH  LIN    177
0000H  0366H  LIN    179
```

```
0000H  0373H  LIN   190
0000H  038DH  LIN   192
0000H  03A1H  LIN   195
0000H  03A6H  LIN   198
0000H  03B4H  LIN   192
0000H  03CCH  LIN   196
0000H  03D5H  LIN   200
0000H  03DEH  LIN   202
0000H  03E6H  LIN   204
0000H  03EEH  LIN   207
0000H  0405H  LIN   209
0000H  0417H  LIN   212
0000H  0421H  LIN   214
0000H  0428H  LIN   216
0000H  0433H  LIN   218
0000H  0436H  LIN   222
0000H  044AH  LIN   224
0000H  0462H  LIN   226
0000H  046CH  LIN   228
0000H  056FH  LIN   234
0000H  057CH  LIN   236
0000H  0589H  LIN   238
0000H  0476H  LIN   240
0000H  048AH  LIN   242
0000H  04A3H  LIN   244
0000H  04B7H  LIN   246
0000H  04C3H  LIN   248
0000H  04D8H  LIN   251
0000H  04EAH  LIN   254
0000H  04FBH  LIN   256
0000H  04FFH  LIN   259
0000H  050DH  LIN   262
0000H  051AH  LIN   265
0000H  052AH  LIN   268
0000H  0534H  LIN   270
0000H  0541H  LIN   273
0000H  0553H  LIN   275
0000H  0561H  LIN   277
0000H  056AH  LIN   279
0000H  0590H  LIN   281
0000H  059EH  LIN   283
0000H  05A3H  LIN   287
0000H  05ACH  LIN   289
0000H  05C6H  LIN   292
0000H  05CFH  LIN   294
0000H  05E6H  LIN   297
0000H  05F3H  LIN   299
0000H  0607H  LIN   302
0000H  0611H  LIN   304
0000H  061FH  LIN   307
0000H  0628H  LIN   309
0000H  0636H  LIN   311
0000H  0648H  LIN   314
0000H  0654H  LIN   317

0000H  037FH  LIN   181
0000H  0394H  LIN   183
0000H  03A4H  LIN   186
0000H  03ADH  LIN   190
0000H  03B6H  LIN   195
0000H  03CEH  LIN   199
0000H  03D8H  LIN   201
0000H  03E4H  LIN   203
0000H  03F9H  LIN   206
0000H  03FAH  LIN   208
0000H  0410H  LIN   211
0000H  041CH  LIN   213
0000H  0423H  LIN   215
0000H  042EH  LIN   217
0000H  0433H  LIN   219
0000H  043EH  LIN   223
0000H  0455H  LIN   225
0000H  0468H  LIN   227
0000H  056CH  LIN   232
0000H  0576H  LIN   235
0000H  0582H  LIN   237
0000H  046FH  LIN   239
0000H  0484H  LIN   241
0000H  0494H  LIN   243
0000H  04AFH  LIN   245
0000H  04BFH  LIN   247
0000H  04D1H  LIN   250
0000H  04DFH  LIN   252
0000H  04F1H  LIN   255
0000H  04FDH  LIN   257
0000H  0506H  LIN   261
0000H  0510H  LIN   263
0000H  0521H  LIN   266
0000H  052DH  LIN   269
0000H  053AH  LIN   271
0000H  0548H  LIN   274
0000H  055AH  LIN   276
0000H  0567H  LIN   278
0000H  058DH  LIN   280
0000H  0597H  LIN   282
0000H  05A1H  LIN   284
0000H  05A6H  LIN   288
0000H  05BCH  LIN   290
0000H  05C9H  LIN   293
0000H  05DBH  LIN   295
0000H  05EDH  LIN   298
0000H  05F9H  LIN   300
0000H  060EH  LIN   303
0000H  0618H  LIN   305
0000H  0628H  LIN   308
0000H  0620H  LIN   310
0000H  063DH  LIN   313
0000H  0648H  LIN   316
0000H  0102H  LIN   318

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

SEGMENT MAP
```

START	STOP	LENGTH	ALIGN	NAME	CLASS
00000H	00655H	0656H	G	CODE	CODE
10000H	10107H	0108H	G	DATS	DATA
10108H	10153H	004CH	W	DATA	DATA
10154H	10191H	003EH	W	STACK	STACK
10192H	10261H	0000H	W	CONST	CONST
10270H	10270H	0000H	G	??SEG	
10270H	10270H	0000H	W	MEMORY	MEMORY

GROUP MAP

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