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



```
                $title ('Help Utility Version 1.1')
  1             help:
                do;

                /*
                  Copyright (C) 1983
                  Digital Research
                  P.O. 579
                  Pacific Grove, CA 93950

                  Revised:
                    13 Feb 83  whf: fix paging bug
                    28 Jan 83  by Bill Fitler: add in Bruce's changes to date
                    07 Oct 82  by Bruce Skidmore
                */

                $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.
            =   ****/

  2   1         declare plmstart label public;

                /*********************************************
                          Interface Procedures
                *********************************************/
  3   1          mon1:
                   procedure (func,info) external;
  4   2              declare func byte;
  5   2              declare info address;
  6   2           end mon1;

  7   1          mon2:
                   procedure (func,info) byte external;
```

```
 8  2          declare func byte;
 9  2          declare info address;
10  2        end mon2;

11  1      mon3:
           procedure (func,info) address external;
12  2          declare func byte;
13  2          declare info address;
14  2        end mon3;



      /*********************************************
                Global Variables
      *********************************************/

15  1      declare (list$mode,nopage$mode,create$mode,extract$mode,page$mode) byte;
16  1      declare (offset,eod) byte;

17  1      declare fcb (13) byte external;
18  1      declare fcb2 (36) byte;

19  1      declare maxb address external;
20  1      declare fcb16 (1) byte external;
21  1      declare tbuff (128) byte external;

22  1      declare control$z literally '1AH';
23  1      declare cr literally '0DH';
24  1      declare lf literally '0AH';
25  1      declare tab literally '09H';
26  1      declare slash literally '''/''';
27  1      declare true literally '0FFH';
28  1      declare false literally '00H';

29  1      declare (cnt,index) byte;
30  1      declare sub(12) byte;
31  1      declare com(11) structure(
           name(15) byte);

32  1      declare sysbuff(8) structure(
           subject(12) byte,
           record address,
           rec$offset byte,
           level byte) at (.memory);

33  1     declare name(12) byte;
34  1     declare level byte;
35  1     declare gindex address;
36  1     declare tcnt byte;
37  1     declare version address;
38  1     declare page$len byte;
39  1     declare clear$screen (26) byte initial (cr,lf,lf,lf,lf,lf,lf,
                                                   lf,lf,lf,lf,lf,lf,
                                                   lf,lf,lf,lf,lf,lf,
                                                   lf,lf,lf,lf,lf,lf,'$');

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

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

 40   1         read$console:
                  procedure byte;
 41   2             return mon2 (1,0);
 42   2           end read$console;

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

 47   1         print$console$buf:
                  procedure (buff$adr);
 48   2             declare buff$adr address;
 49   2             call mon1 (9,buff$adr);
 50   2           end print$console$buf;

 51   1         read$console$buff:
                  procedure (buff$adr);
 52   2             declare buff$adr address;
 53   2             call mon1(10,buff$adr);
 54   2           end read$console$buff;

 55   1         direct$con$io:
                  procedure(func) byte;
 56   2             declare func byte;
 57   2               return mon2(6,func);
 58   2           end direct$con$io;

 59   1         get$version:
                  procedure  address;
 60   2             return mon3(12,0);
 61   2           end get$version;

 62   1         delete$file:
                  procedure (fcb$address);
 63   2             declare fcb$address address;
 64   2             call mon1(19,fcb$address);
 65   2           end delete$file;

 66   1         open$file:
                  procedure (fcb$address) byte;
 67   2             declare fcb$address address;
 68   2             declare fcb based fcb$address (1) byte;
 69   2             fcb(12) = 0; /* EX = 0 */
 70   2             fcb(32) = 0; /* CR = 0 */
 71   2             return mon2 (15,fcb$address);
 72   2           end open$file;

 73   1         close$file:
                  procedure (fcb$address) byte;
 74   2             declare fcb$address address;
```

```
 75  2            return mon2 (16,fcb$address);
 76  2          end close$file;

 77  1        read$record:
              procedure (fcb$address) byte;
 78  2            declare fcb$address address;
 79  2            return mon2 (20,fcb$address);
 80  2          end read$record;

 81  1        write$record:
              procedure (fcb$address) byte;
 82  2            declare fcb$address address;
 83  2            return mon2(21,fcb$address);
 84  2          end write$record;

 85  1        make$file:
              procedure (fcb$address) byte;
 86  2            declare fcb$address address;
 87  2            declare fcb based fcb$address (1) byte;
 88  2            fcb(12) = 0; /* EX = 0 */
 89  2            fcb(32) = 0; /* CR = 0 */
 90  2            return mon2(22,fcb$address);
 91  2          end make$file;

 92  1        read$rand:
              procedure (fcb$address) byte;
 93  2            declare fcb$address address;
 94  2            return mon2(33,fcb$address);
 95  2          end read$rand;

 96  1        set$dma:
              procedure (dma$address);
 97  2            declare dma$address address;
 98  2            call mon1(26,dma$address);
 99  2          end set$dma;

100  1        set$rand$rec:
              procedure (fcb$address);
101  2            declare fcb$address address;
102  2            call mon1(36,fcb$address);
103  2          end set$rand$rec;

104  1        set$user:
              procedure (usernum);
105  2            declare usernum byte;
106  2            call mon1(32,usernum);
107  2          end set$user;

108  1        declare cmdrv byte external;      /* declared in assembler interface */
109  1        get$sysdrive:
              procedure byte;
110  2            if (high(get$version) and 0F0H) <> 14H then   /* CCP/M-86 ? */
111  2              return 0;       /* no: sysdrive defaults to A */
112  2            return cmdrv;     /* yes: sysdrive is drive we loaded from */
113  2          end get$sysdrive;

114  1        terminate:
```

```
                    procedure;
115    2              call mon1 (0,0);
116    2            end terminate;

                /*********************************************
                            Error Procedure

                      Displays error messages and
                      terminates if required.
                *********************************************/
117    1        error:
                  procedure(term$code,err$msg$adr);
118    2            declare term$code byte;
119    2            declare err$msg$adr address;

120    2            call print$console$buf(.(cr,lf,'ERROR:  $'));
121    2            call print$console$buf(err$msg$adr);
122    2            call print$console$buf(.(cr,lf,'$'));
123    2            if term$code then
124    2              call terminate;
125    2          end error;

                /*********************************************
                            Move Procedure

                      Moves specified number of bytes
                      from the Source address to the
                      Destination address.
                *********************************************/
126    1        movef:
                  procedure (mvcnt,source$addr,dest$addr);
127    2            declare (source$addr,dest$addr) address;
128    2            declare mvcnt byte;
129    2            call move(mvcnt,source$addr,dest$addr);
130    2            return;
131    2          end movef;

                /*********************************************
                            Compare Function

                      Compares 12 byte strings

                      Results:  0 - string1 = string2
                                1 - string1 < string2
                                2 - string1 > string2
                *********************************************/
132    1        compare:
                  procedure(str1$addr,str2$addr) byte;
133    2            declare (str1$addr,str2$addr) address;
134    2            declare string1 based str1$addr (12) byte;
135    2            declare string2 based str2$addr (12) byte;
136    2            declare (result,i) byte;
137    2            result,
                    i = 0;
138    2            do while ((i < 12) and (string1(i) <> ' '));
139    3              if string1(i) <> string2(i) then
140    3                do;
```

```
141  4                if string1(i) < string2(i) then
142  4                do;
143  5                   result = 1;
144  5                end;
                      else
145  4                do;
146  5                   result = 2;
147  5                end;
148  4                i = 11;
149  4             end;
150  3             i = i + 1;
151  3          end;
152  2          return result;
153  2       end compare;

          /*********************************************
                    Increment Procedure

              Increments through a record.
          *********************************************/
154  1       inc:
            procedure (inci) byte;
155  2          declare inci byte;
156  2          inci = inci + 1;
157  2          if inci > 127 then
158  2          do;
159  3             if read$record(.fcb) = 0 then
160  3             do;
161  4                inci = 0;
162  4             end;
                   else
163  3             do;
164  4                eod = true;
165  4                inci = 0;
166  4             end;
167  3          end;
168  2          return inci;
169  2       end inc;

          /*************************************
                  Page$check Procedure

              Halts display after page$len lines
          *************************************/
170  1       page$check:
            procedure(line$cnt$addr) byte;
171  2          declare line$cnt$addr address;
172  2          declare line$cnt based line$cnt$addr byte;
173  2          declare quit byte;
174  2          quit = 0;
175  2          if (not nopage$mode) and (page$mode) then
176  2          do;
177  3             if (line$cnt:=line$cnt+1) > page$len then
178  3             do;
179  4                call print$console$buf(.(cr,lf,'Press RETURN to continue $'));
180  4                line$cnt = 0;
181  4                do while (line$cnt = 0);
```

```
182  5                  line$cnt = direct$con$io(0FDH);
183  5               end;
184  4               call print$console$buf(.(cr,'
                                                  ',
                                               cr,'$'));
185  4               if line$cnt = 3 /* control c */ then
186  4               do;
187  5                  line$cnt = close$file(.fcb);
188  5                  call terminate;
189  5               end;
                     else
190  4               do;
191  5                  if line$cnt <> cr then
192  5                  do;
193  6                     quit = true;
194  6                  end;
195  5                  line$cnt = 0;
196  5               end;
197  4            end;
                  else
198  3            do;
199  4               call write$console(lf);
200  4            end;
201  3         end;
               else
202  2         do;
203  3            line$cnt = 0;
204  3            call write$console(lf);
205  3         end;
206  2         return quit;
207  2      end page$check;

          /*******************************************
                      Init Procedure

               Reads the index into memory
          *******************************************/
208  1      init:
           procedure;
209  2         declare (buf$size,max$buf,init$i) address;
210  2         declare end$index byte;
211  2         buf$size = maxb - .memory;
212  2         max$buf = buf$size;
213  2         end$index = 0;
214  2         init$i = 7;
215  2         do while (not end$index) and (max$buf > 127);
216  3            call set$dma(.sysbuff(init$i-7).subject);
217  3            if read$record(.fcb) <> 0 then
218  3            do;
219  4               init$i = close$file(.fcb);
220  4               call error(true,.('Reading HELP.HLP index.$'));
221  4            end;
222  3            if sysbuff(init$i).subject(0) = '$' then end$index = true;
224  3            if not end$index then
225  3            do;
226  4               max$buf = max$buf - 128;
227  4               init$i = init$i + 8;
228  4            end;
```

```
229  3          end;
230  2          call set$dma(.tbuff);
231  2          if (max$buf < 128) and (not end$index) then
232  2          do;
233  3             init$i = close$file(.fcb);
234  3             call error(true,.('Too many entries in Index Table.',
                                 '  Not enough memory.$'));
235  3          end;
236  2        end init;

        /*****************************************
                    Parse Procedure

             Parses the command tail
        *****************************************/
237  1      parse:
          procedure byte;
238  2        declare (index,begin,cnt,i,stop,bracket) byte;
239  2        index = 0;
240  2        if tbuff(0) <> 0 then
241  2        do;
242  3           do index = 1 to tbuff(0);
243  4              if tbuff(index) = tab then tbuff(index) = ' ';
245  4              else if tbuff(index) = ',' then tbuff(index) = ' ';
248  3           end;
249  3           index = 1;
250  3           do while(index < tbuff(0)) and (tbuff(index) = ' ');
251  4             index = index + 1;
252  4           end;
253  3           if tbuff(index) = '.' then
254  3             do;
255  4               begin = level;
256  4               tbuff(index) = ' ';
257  3             end;
258  3           else
259  4             begin = 0;
260  4           do index = begin to 10;
261  3              call movef(15,.('             ',cr,'$'),.com(index).name);
262  3           end;
263  3           index = begin;
264  3           cnt = 1;
265  3           stop,
266  4           bracket = 0;
267  5           do while (tbuff(cnt) <> 0) and (not stop);
268  5              if (tbuff(cnt) <> 20H) then
                   do;
269  6                 i = 0;
270  6                 do while (((tbuff(cnt) <> 20H) and (tbuff(cnt) <> '[')) and
                                (tbuff(cnt) <> 0)) and ((i < 12) and (index < 11));
271  7                    if (tbuff(cnt) > 60H) and (tbuff(cnt) < 78H) then
272  7                    do;
                            com(index).name(i) = tbuff(cnt) - 20H;
                          end;
                          else
273  6                    do;
274  7                       com(index).name(i) = tbuff(cnt);
```

```
275  7                    end;
276  6                    cnt = cnt + 1;
277  6                    i = i + 1;
278  6                 end;
279  5                 index = index + 1;
280  5                 if (bracket or (index > 10)) then
281  5                 do;
282  6                    stop = true;
283  6                 end;
                       else
284  5                 if tbuff(cnt) = '[' then
285  5                 do;
286  6                    if com(index-1).name(0) = ' ' then index = index - 1;
288  6                    com(index).name(0) = '[';
289  6                    cnt = cnt + 1;
290  6                    index = index + 1;
291  6                    bracket = true;
292  6                 end;
                    end;
                    else
294  4                 do;
295  5                    cnt = cnt + 1;
296  5                 end;
297  4              end;
298  3           end;
299  2           list$mode,
                nopage$mode,
                create$mode,
                extract$mode = false;
300  2           if index > 0 then
301  2           do;
302  3              i = 0;
303  3              do while (i < 10);
304  4                 if com(i).name(0) = '[' then
305  4                 do;
306  5                    if (com(i+1).name(0) = '[') then
307  5                    do;
308  6                       create$mode = true;
309  6                       index = index - 2;
310  6                    end;
311  5                    else if (com(i+1).name(0) = 'E') then
312  5                         do;
313  6                            extract$mode = true;
314  6                            index = index - 2;
315  6                         end;
316  5                         else if (com(i+1).name(0) = 'N') then
317  5                              do;
318  6                                 nopage$mode =true;
319  6                                 index = index - 2;
320  6                              end;
321  5                              else if (com(i+1).name(0) = 'L') then
322  5                                   do;
323  6                                      list$mode = true;
324  6                                      nopage$mode = true;
325  6                                      index = index - 2;
326  6                                   end;
327  5                                   else if (com(i+1).name(0) <> ' ') then
```

```
328   5                                    do;
329   6                                      index = index - 2;
330   6                                    end;
                                           else
331   5                                        do;
332   6                                          index = index - 1;
333   6                                        end;
334   5                  i = 10;
335   5                end;
336   4                i = i + 1;
337   4              end;
338   3            end;
339   2            return index;
340   2          end parse;

           /*****************************************
                   Create$index Procedure

                Creates HELP.HLP from HELP.DAT
           *****************************************/
341   1       create$index:
             procedure;
342   2         declare (cnt, i, rec$cnt) byte;
343   2         declare (index,count,count2,max$buf,save$size) address;
344   2         declare fcb3(36) byte;
345   2         call print$console$buf(.(cr,lf,'Creating HELP.HLP....$'));
346   2         do i = 0 to 7;
347   3            call movef(12,.('$           '),.sysbuff(i).subject);
348   3         end;
349   2         rec$cnt,
           index = 0;
350   2         save$size = maxb - .memory;
351   2         max$buf = save$size;
352   2         call movef(13,.(0,'HELP    DAT',0),.fcb);
353   2         if open$file(.fcb) = 0FFH then
354   2         do;
355   3            call error(true,.('HELP.DAT not on current drive.$'));
356   3         end;
357   2         eod = 0;
358   2         do while (not eod) and (read$record(.fcb) = 0);
359   3            i = 0;
360   3            do while(i < 128) and (not eod);
361   4               if tbuff(i) = control$z then
362   4               do;
363   5                  eod = true;
364   5               end;
                        else
365   4               do;
366   5                  if tbuff(i) = slash then
367   5                  do;
368   6                     cnt = 0;
369   6                     do while(not eod) and (tbuff(i) = slash);
370   7                        i = inc(i);
371   7                        cnt = cnt + 1;
372   7                     end;
373   6                     if (cnt = 3) and (not eod) then
374   6                     do;
```

```
375   7                        sysbuff(index).level = tbuff(i) - '0';
376   7                        i = inc(i);
377   7                        cnt = 0;
378   7                        do while ((cnt < 12) and (not eod)) and (tbuff(i) <> cr);
379   8                           if (tbuff(i) > 60H) and (tbuff(i) < 7BH) then
380   8                           do;
381   9                              sysbuff(index).subject(cnt) = tbuff(i) - 20H;
382   9                           end;
                                 else
383   8                           do;
384   9                              sysbuff(index).subject(cnt) = tbuff(i);
385   9                           end;
386   8                           i = inc(i);
387   8                           cnt = cnt + 1;
388   8                        end;
389   7                        if (not eod) then
390   7                        do;
391   8                           call set$rand$rec(.fcb);
392   8                           call movef(1,.fcb(33),.sysbuff(index).record);
393   8                           call movef(1,.fcb(34),.sysbuff(index).record+1);
394   8                           sysbuff(index).record = sysbuff(index).record - 0001H;
395   8                           sysbuff(index).rec$offset = 1;
396   8                           index = index + 1;
397   8                           if ((index mod 8) = 0) then
398   8                           do;
399   9                              rec$cnt = rec$cnt + 1;
400   9                              max$buf = max$buf - 128;
401   9                              if (max$buf < 128) and (not eod) then
402   9                              do;
403  10                                 cnt = close$file(.fcb);
404  10                                 call error(true,
                                                .('Too many entries in Index Table.',
                                                  '  Not enough memory.$'));
405  10                              end;
                                    else
406   9                              do count = index to index + 7;
407  10                                 call movef(12,.('$           '),
                                                      .sysbuff(count).subject);
408  10                              end;
409   9                           end;
410   8                        end;
411   7                     end;
412   6                  end;
                           else
                           do;
413   5                          i = inc(i);
414   6                       end;
415   6                    end;
416   5                 end;
417   4              end;
418   3           end;
419   2           call set$dma(.sysbuff);
420   2           rec$cnt = rec$cnt + 1;
                  /*******************************
                       create HELP.HLP
                  *******************************/
421   2           call movef(13,.(0,'HELP    HLP',0),.fcb3);
422   2           call delete$file(.fcb3);
```

```
423  2          if make$file(.fcb3) = 0FFH then
424  2          do;
425  3             cnt = close$file(.fcb2);
426  3             call delete$file(.fcb2);
427  3             cnt = close$file(.fcb);
428  3             call error(true,.('Unable to Make HELP.HLP.$'));
429  3          end;
430  2          call movef(4,.(0,0,0,0),.fcb2+32);
431  2          cnt = read$rand(.fcb2);
432  2          do count = 0 to index - 1;
433  3             sysbuff(count).record = sysbuff(count).record + rec$cnt;
434  3          end;
435  2          do count = 0 to rec$cnt - 1;
436  3             call set$dma(.memory(shl(count,7)));
437  3             if write$record(.fcb3) <> 0 then
438  3             do;
439  4                cnt = close$file(.fcb3);
440  4                call delete$file(.fcb3);
441  4                cnt = close$file(.fcb2);
442  4                call delete$file(.fcb2);
443  4                cnt = close$file(.fcb);
444  4                call error(true,.('Writing file HELP.HLP.$'));
445  4             end;
446  3          end;
447  2          call movef(4,.(0,0,0,0),.fcb+32);
448  2          cnt = read$rand(.fcb);
449  2          eod = 0;
450  2          do while (not eod);
451  3               count = 0;
452  3               max$buf = save$size;
453  3               do while (not eod) and (max$buf > 127);
454  4                  call set$dma(.memory(shl(count,7)));
455  4                  if read$record(.fcb) <> 0 then
456  4                  do;
457  5                     eod = true;
458  5                  end;
                        else
459  4                  do;
460  5                     max$buf = max$buf - 128;
461  5                     count = count + 1;
462  5                  end;
463  4               end;
464  3               do count2 = 0 to count-1;
465  4                  call set$dma(.memory(shl(count2,7)));
466  4                  if write$record(.fcb3) <> 0 then
467  4                  do;
468  5                     i = close$file(.fcb3);
469  5                     call delete$file(.fcb3);
470  5                     i = close$file(.fcb);
471  5                     call error(true,.('Writing file HELP.HLP.$'));
472  5                  end;
473  4               end;
474  3          end;
475  2          if close$file(.fcb) = 0FFH then
476  2          do;
477  3             cnt = close$file(.fcb3);
478  3             call error(true,.('Closing file HELP.DAT.$'));
```

```
479  3          end;
480  2          if close$file(.fcb3) = 0FFH then
481  2          do;
482  3             call error(true,.(false,'Closing file HELP.HLP.$'));
483  3          end;
484  2          call print$console$buf(.('HELP.HLP created',cr,lf,'$'));
485  2        end create$index;

          /*******************************************
                  Extract$file Procedure

               Creates HELP.DAT from HELP.HLP
          *******************************************/
486  1       extract$file:
            procedure;
487  2          declare (end$index,i) byte;
488  2          declare (count,count2,max$buf,save$size) address;

489  2            call print$console$buf(.(cr,lf,'Extracting data....$'));
490  2            call movef(13,.(0,'HELP    HLP',0),.fcb);
491  2            if open$file(.fcb) = 0FFH then
492  2            do;
493  3               call error(true,.('Unable to find file HELP.HLP.$'));
494  3            end;
495  2            call movef(13,.(0,'HELP    DAT',0),.fcb2);
496  2            call delete$file(.fcb2);
497  2            if make$file(.fcb2) = 0FFH then
498  2            do;
499  3               i = close$file(.fcb);
500  3               call error(true,.('Unable to Make HELP.DAT.$'));
501  3            end;
502  2            call set$dma(.sysbuff);
503  2            end$index = 0;
504  2            do while ((i := read$record(.fcb)) = 0) and (not end$index);
505  3                if sysbuff(7).subject(0) = '$' then end$index = true;
507  3            end;
508  2            eod = 0;
509  2            if i <> 0 then eod = true;
511  2            i = write$record(.fcb2);
512  2            save$size = maxb - .memory;
513  2            do while (not eod);
514  3               count = 0;
515  3               max$buf = save$size;
516  3               do while (not eod) and (max$buf > 127);
517  4                  call set$dma(.memory(shl(count,7)));
518  4                  if read$record(.fcb) <> 0 then
519  4                  do;
520  5                     eod = true;
521  5                  end;
                        else
522  4                  do;
523  5                     max$buf = max$buf - 128;
524  5                     count = count + 1;
525  5                  end;
526  4               end;
527  3               do count2 = 0 to count-1;
528  4                  call set$dma(.memory(shl(count2,7)));
```

```
529   4                 if write$record(.fcb2) <> 0 then
530   4                 do;
531   5                    i = close$file(.fcb2);
532   5                    call delete$file(.fcb2);
533   5                    i = close$file(.fcb);
534   5                    call error(true,.('Writing file HELP.DAT.$'));
535   5                 end;
536   4              end;
537   3           end;
538   2           if close$file(.fcb) = 0FFH then
539   2           do;
540   3              call error(false,.('Unable to Close HELP.HLP.$'));
541   3           end;
542   2           if close$file(.fcb2) = 0FFH then
543   2           do;
544   3              call delete$file(.fcb2);
545   3              call error(true,.('Unable to Close HELP.DAT.$'));
546   3           end;
547   2           call print$console$buf(.('Extraction complete',cr,lf,lf,
                                          'HELP.DAT created',cr,lf,'$'));

548   2        end extract$file;

                /***********************************************
                          Display$ind Procedure

                     Displays the avialable topics
                ***********************************************/
549   1        display$ind:
                  procedure;
550   2           declare (disp$level,i,eod,written) byte;
551   2           declare (offset,index,count) address;
552   2           declare name (14) byte;
553   2            offset,
                     written,
                     eod = 0;
554   2            disp$level = level + 1;
555   2            if disp$level < 10 then
556   2            do;
557   3               if level = 0 then
558   3               do;
559   4                  offset = 0;
560   4               end;
                        else
561   3               do;
562   4                  offset = gindex;
563   4               end;
564   3               count = 0;
565   3            end;
                     else
566   2            do;
567   3               eod = true;
568   3            end;
569   2            index = offset;
570   2            offset = 0;
571   2            do while (not eod);
572   3                 if sysbuff(index).subject(0) = '$' then
```

```
573   3                do;
574   4                   eod = true;
575   4                end;
                       else
576   3                do;
577   4                   if sysbuff(index).level = disp$level then
578   4                   do;
579   5                      if not written then
580   5                      do;
581   6                         written = true;
582   6                         i = page$check(.tcnt);
583   6                         if disp$level = 1 then
584   6                         do;
585   7                            call print$console$buf(.(cr,'Topics available:$'));
586   7                         end;
                                else
587   6                         do;
588   7                            call print$console$buf(.(cr,'Additional topics ',
                                                           'available:$'));
589   7                         end;
590   6                         i = page$check(.tcnt);
591   6                         call print$console$buf(.(cr,'$'));
592   6                      end;
593   5                      if (count mod 6) = 0 then
594   5                      do;
595   6                         i = page$check(.tcnt);
596   6                         call write$console(cr);
597   6                      end;
598   5                      do i = 0 to 13;
599   6                         name(i) = ' ';
600   6                      end;
601   5                      name(13) = '$';
602   5                      call movef(12,.sysbuff(index).subject,.name);
603   5                      call print$console$buf(.name);
604   5                      count = count + 1;
605   5                   end;
                          else
606   4                   do;
607   5                      if sysbuff(index).level < disp$level then eod = true;
609   5                   end;
610   4                   index = index + 1;
611   4                end;
612   3             end;
613   2             if written then
614   2               do;
615   3                  i = page$check(.tcnt);
616   3                  call print$console$buf(.(cr,lf,'$'));
617   3               end;
618   2             call set$dma(.tbuff);
619   2          end display$ind;

              /***********************************************
                        Search$file Procedure

                   Searches the index table for the key
              ***********************************************/
620   1          search$file:
```

```
                procedure byte;
621   2           declare (eod, error, cnt, found, saved, save$level) byte;
622   2           declare index address;
623   2           eod,
                  error,
                  found,
                  saved,
                  index = 0;
624   2           do while(not eod) and (not error);
625   3                 if sysbuff(index).subject(0) <> '$' then
626   3                 do;
627   4                    if sysbuff(index).level = level + 1 then
628   4                    do;
629   5                       cnt = compare(.com(level).name,.sysbuff(index).subject);
630   5                       if cnt = 0 then
631   5                       do;
632   6                          call movef(12,.sysbuff(index).subject,.com(level).name);
633   6                          level = level + 1;
634   6                          if (not saved) then
635   6                          do;
636   7                             save$level = level;
637   7                             saved = true;
638   7                          end;
639   6                          if ((level > 8) or (com(level).name(0) = ' '))
                                                  or (com(level).name(0) = 'E') then
640   6                          do;
641   7                             found = true;
642   7                             eod = true;
643   7                          end;
                                 else
644   6                          do;
645   7                             index = index + 1;
646   7                             found = 0;
647   7                          end;
648   6                       end;
                              else
649   5                       do;
650   6                          index = index + 1;
651   6                       end;
652   5                    end;
                           else
653   4                    do;
654   5                       if saved then
655   5                       do;
656   6                          if save$level < sysbuff(index).level then
657   6                          do;
658   7                             index = index + 1;
659   7                          end;
                                 else
660   6                          do;
661   7                              error = true;
662   7                          end;
663   6                       end;
                              else
664   5                       do;
665   6                          index = index + 1;
666   6                       end;
```

```
667   5                 end;
668   4              end;
                     else
669   3              do;
670   4                 error = true;
671   4              end;
672   3           end;
673   2           if found then
674   2           do;
675   3              gindex = index + 1;
676   3              call movef(1,.sysbuff(index).record,.fcb(33));
677   3              call movef(1,.sysbuff(index).record+1,.fcb(34));
678   3              fcb(35) = 0;
679   3              offset = sysbuff(index).rec$offset;
680   3              level = sysbuff(index).level;
681   3           end;
682   2           return error;
683   2        end search$file;

               /**************************************
                        Token Display Procedure

                     Displays the Parsed Tokens
               **************************************/
684   1        display$tokens:
               procedure (no$tokens);
685   2           declare (token$cnt1, token$cnt2, no$tokens) byte;
686   2           token$cnt1 = 0;
687   2           do while (token$cnt1 < no$tokens) and (not eod);
688   3              eod = page$check(.tcnt);
689   3              if (not eod) then
690   3              do;
691   4                 do token$cnt2 = 0 to token$cnt1;
692   5                    call print$console$buf(.(' $'));
693   5                 end;
694   4                 call print$console$buf(.com(token$cnt1).name);
695   4                 token$cnt1 = token$cnt1 + 1;
696   4              end;
697   3           end;
698   2        end display$tokens;

               /**************************************
                         Print Procedure

                     Displays the Help text
               **************************************/
699   1        print:
               procedure;
700   2           declare (i,ii,char,eod2) byte;
701   2           declare temp(3) byte;
702   2           call write$console(cr);
703   2           call display$tokens(level);
704   2           if (not eod) then eod = page$check(.tcnt);
706   2           if (not eod) then
707   2           do;
708   3              if read$rand(.fcb) <> 0 then
709   3              do;
```

```
                  offset =close$file(.fcb);
                  call error(true,.('Reading file HELP.HLP.$'));
               end;
               else
               do;
                  eod2 = 0;
                  do while ((not eod2) and (not eod)) and (read$record (.fcb) = 0);
                     i = offset - 1;
                     do while (((i:=i+1) <= 127) and (not eod2));
                        if (char := tbuff(i)) = control$z then eod = true;
                        ii = 0;
                        do while((not eod2) and (not eod)) and
                                                ((ii < 3) and (tbuff(i) = slash));
                           ii = ii + 1;
                           i = inc(i);
                           temp(ii-1) = tbuff(i);
                        end;
                        if ii = 3 then eod2 = true; else temp(ii) = '$';
                        if ((not eod) and (not eod2)) then
                        do;
                           if (char = 1f) and (not nopage$mode) then
                           do;
                              eod = page$check(.tcnt);
                           end;
                           else
                           do;
                              call write$console (char);
                           end;
                           if ii > 0 then call print$console$buf(.temp);
                           ii = 0;
                        end;
                     end;
                     offset = 0;
                  end;
               end;
            end;
            eod = 0;
         end print;

         /**************************************
                  Prompt Procedure

           Prompts for input from the user
         **************************************/
         prompt:
           procedure byte;
             declare temp byte;
             call movef(1,.(128),.tbuff-1);
             temp = page$check(.tcnt);
             call print$console$buf(.(cr,'HELP> $'));
             call read$console$buff(.tbuff-1);
             tbuff(tbuff(0)+1) = 0;
             tcnt = -1;
             temp = parse;
             if (temp <> 0) and (not list$mode)
             then call print$console$buf(.clear$screen);
             return temp;
```

```
761  2       end prompt;

            /*************************************
                    Main Program
            *************************************/

762  1       declare oret byte;

763  1       declare last$dseg$byte byte
              initial (0);

764  1       plmstart:
              do;
765  2           eod,
                tcnt = 0;
766  2           version = get$version;

            /******** CP/M 3
                if (high(version) = 1) or (low(version) < 30h) then
                  do;
                    call error(true,.('Requires CP/M Version 3$'));
                  end;
                page$len = mon2(49,.(1ch,0,));
                if mon2(49,.(2ch,0)) = 0 then
                  page$mode = true;
                else
                  page$mode = false;
            *********/

767  2           page$len = 22;
768  2           page$mode = true;

769  2           cnt = parse;
770  2           if create$mode then
771  2           do;
772  3              call create$index;
773  3           end;
                else
774  2           if extract$mode then
775  2           do;
776  3              call extract$file;
777  3           end;
                else
778  2           do;
779  3              call movef(13,.(0,'HELP ',0A0H,'  HLP',0),.fcb); /* open read/only */
780  3              oret = open$file (.fcb);
781  3              if oret = 0FFH then
782  3                do;
783  4                  fcb(0) = get$sysdrive;
784  4                call set$user(0);        /* try user 0 */
785  4                   oret = open$file (.fcb); /* on system drive */
786  4                end;
787  3              if oret <> 0FFH then
788  3              do;
789  4                call init;
790  4                if (not list$mode) then
```

```
 791   4                call print$console$buf(.clear$screen);
 792   4              if cnt = 0 then
 793   4              do;
 794   5                 level = 0;
 795   5                 call print$console$buf(.(cr,lf,'HELP UTILITY V1.1',cr,lf,lf,
                                                  'At "HELP>" enter ',
                                                  'topic (,subtopic)...',cr,lf,lf,
                                                  'EXAMPLE:  HELP> DIR EXAMPLES',
                                                   cr,lf,'$'));
 796   5                 tcnt = 2;
 797   5                 call display$ind;
 798   5                 cnt = prompt;      /* Prompt for user input */
 799   5              end;
 800   4              do while cnt <> 0;    /* If user didn't hit a return do */
 801   5               level = 0;
 802   5               if compare(.com(0).name,.('?       ')) = 0 then
 803   5               do;
 804   6                 ; /* NULL COMMAND */
 805   6               end;
                       else
 806   5               if search$file <> 0FFH then
 807   5               do;
 808   6                  call print;
 809   6                  if compare(.com(0).name,.('HELP       ')) = 0 then
 810   6                  do;
 811   7                     level = 0;
 812   7                  end;
 813   6               end;
                       else
 814   5               do;
 815   6                  eod = page$check(.tcnt);
 816   6                  call write$console(cr);
 817   6                  if (not eod) then
 818   6                  do;
 819   7                     eod = page$check(.tcnt);
 820   7                     if (not eod) then
 821   7                     do;
 822   8                        call print$console$buf(.('Topic:$'));
 823   8                        eod = page$check(.tcnt);
 824   8                        call write$console(cr);
 825   8                        call display$tokens(cnt);
 826   8                        eod = page$check(.tcnt);
 827   8                        call write$console(cr);
 828   8                        eod = page$check(.tcnt);
 829   8                        call write$console(cr);
 830   8                        call print$console$buf(.('Not found$'));
 831   8                        eod = page$check(.tcnt);
 832   8                        call write$console(cr);
 833   8                     end;
 834   7                  end;
 835   6                  level = 0;
 836   6               end;
 837   5               if (not eod) then call display$ind;
 839   5               cnt = prompt;   /* Prompt for user input */
 840   5            end;
 841   4           offset = close$file(.fcb);
 842   4        end;
```

```
                else
 843    3       do;
 844    4          call error(false,.('No HELP.HLP file on the default drive.$'));
 845    4       end;
 846    3     end;
 847    2     end;
 848    1     call terminate;
 849    1   end help;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
238	0130H	1	BEGIN. . . . . BYTE 254 257 258 261
238	0134H	1	BRACKET. . . . . BYTE 263 280 291
51	0004H	2	BUFFADR. . . . . WORD PARAMETER AUTOMATIC 52 53
47	0004H	2	BUFFADR. . . . . WORD PARAMETER AUTOMATIC 48 49
209	0004H	2	BUFSIZE. . . . . WORD 211 212
700	017AH	1	CHAR . . . . . BYTE 718 731 736
43	0004H	1	CHAR . . . . . BYTE PARAMETER AUTOMATIC 44 45
39	0111H	26	CLEARSCREEN. . . . BYTE ARRAY(26) INITIAL 759 791
73	023EH	16	CLOSEFILE. . . . . PROCEDURE BYTE STACK=000AH 187 219 233 403 425 427 439 441 443 468 470 475 477 480 499 531 533 538 542 710 841
108	0000H	1	CMDRV. . . . . BYTE EXTERNAL(7) 112
621	0172H	1	CNT. . . . . BYTE 629 630
342	0135H	1	CNT. . . . . BYTE 368 371 373 377 378 381 384 387 403 425 427 431 439 441 443 448 477
29	004FH	1	CNT. . . . . BYTE 769 792 798 800 825 839
238	0131H	1	CNT. . . . . BYTE 262 264 265 268 269 271 274 276 284 289 295
31	005DH	165	COM. . . . . STRUCTURE ARRAY(11) 259 271 274 286 288 304 306 311 316 321 327 629 632 639 694 802 809
132	032BH	93	COMPARE. . . . . PROCEDURE BYTE STACK=0008H 629 802 809
22			CONTROLZ . . . . . LITERALLY 361 718
551	0020H	2	COUNT. . . . . WORD 564 593 604
343	000CH	2	COUNT. . . . . WORD 406 407 432 433 435 436 451 454 461 464
488	0014H	2	COUNT. . . . . WORD 514 517 524 527
488	0016H	2	COUNT2 . . . . . WORD 527 528
343	000EH	2	COUNT2 . . . . . WORD 464 465
23			CR . . . . . LITERALLY 39 120 122 179 184 191 259 345 378 484 489 547 585 588 591 596 616 702 753 795 816 824 827 829 832
341	079DH	1183	CREATEINDEX. . . . PROCEDURE STACK=0016H 772
15	0026H	1	CREATEMODE . . . . BYTE 299 308 770
62	0215H	16	DELETEFILE . . . . PROCEDURE STACK=000AH 422 426 440 442 469 496 532 544
126	0004H	2	DESTADDR . . . . . WORD PARAMETER AUTOMATIC 127 129
55	01F3H	19	DIRECTCONIO. . . . PROCEDURE BYTE STACK=000AH 182
549	0DE8H	359	DISPLAYIND . . . . PROCEDURE STACK=0014H 797 838
684	10BCH	104	DISPLAYTOKENS. . . PROCEDURE STACK=0016H 703 825
550	015EH	1	DISPLEVEL. . . . . BYTE 554 555 577 583 607
96	0004H	2	DMAADDRESS . . . . WORD PARAMETER AUTOMATIC 97 98
487	015CH	1	ENDINDEX . . . . . BYTE 503 504 506
210	012EH	1	ENDINDEX . . . . . BYTE 213 215 223 224 231
16	002AH	1	EOD. . . . . BYTE 164 357 358 360 363 369 373 378 389 401 449 450 453 457 508 510 513 516 520 687 688 689 704 705 706 715 719 721 729 733 747 765 815 817 819 820 823 826 828 831 837
550	0160H	1	EOD. . . . . BYTE 553 567 571 574 608
621	0170H	1	EOD. . . . . BYTE 623 624 642
700	0178H	1	EOD2 . . . . . BYTE 714 715 717 721 727 729
117	0004H	2	ERRMSGADR. . . . . WORD PARAMETER AUTOMATIC 119 121
621	0171H	1	ERROR. . . . . BYTE 623 624 661 670 682
117	02EFH	37	ERROR. . . . . PROCEDURE STACK=0012H 220 234 355 404 428 444 471 478 482 493 500 534 540 545 711 844

```
486  0C3CH   428  EXTRACTFILE. . . .  PROCEDURE STACK=0016H                776
 15  0027H     1  EXTRACTMODE. . . .  BYTE          299  313  774
 28             FALSE. . . . . .    LITERALLY       299  482  540  844
 17  0000H    13  FCB. . . . . .     BYTE ARRAY(13) EXTERNAL(3)                159  187  217  219  233  352  353
                                      358  391  392  393  403  427  443  447  448  455  470  475  490  491
                                      499  504  518  533  538  676  677  678  708  710  715  779  780  783
                                      785  841
 68  0000H     1  FCB. . . . . .     BYTE BASED(FCBADDRESS) ARRAY(1)           69   70
 87  0000H     1  FCB. . . . . .     BYTE BASED(FCBADDRESS) ARRAY(1)           88   89
 20  0000H     1  FCB16. . . . . .    BYTE ARRAY(1) EXTERNAL(5)
 18  0028H    36  FCB2 . . . . . .    BYTE ARRAY(36)          425  426  430  431  441  442  495  496  497  511
                                      529  531  532  542  544
344  0138H    36  FCB3 . . . . . .    BYTE ARRAY(36)          421  422  423  437  439  440  466  468  469  477
                                      480
 77  0004H     2  FCBADDRESS . . . .  WORD PARAMETER AUTOMATIC          78   79
 73  0004H     2  FCBADDRESS . . . .  WORD PARAMETER AUTOMATIC          74   75
 66  0004H     2  FCBADDRESS . . . .  WORD PARAMETER AUTOMATIC          67   68   69   70   71
 62  0004H     2  FCBADDRESS . . . .  WORD PARAMETER AUTOMATIC          63   64
 85  0004H     2  FCBADDRESS . . . .  WORD PARAMETER AUTOMATIC          86   87   88   89   90
100  0004H     2  FCBADDRESS . . . .  WORD PARAMETER AUTOMATIC         101  102
 92  0004H     2  FCBADDRESS . . . .  WORD PARAMETER AUTOMATIC          93   94
 81  0004H     2  FCBADDRESS . . . .  WORD PARAMETER AUTOMATIC          82   83
621  0173H     1  FOUND. . . . . .    BYTE          623  641  646  673
 55  0004H     1  FUNC . . . . . .    BYTE PARAMETER AUTOMATIC          56   57
 11  0000H     1  FUNC . . . . . .    BYTE PARAMETER           12
  7  0000H     1  FUNC . . . . . .    BYTE PARAMETER            8
  3  0000H     1  FUNC . . . . . .    BYTE PARAMETER            4
109  02CAH    23  GETSYSDRIVE. . . .  PROCEDURE BYTE STACK=000CH               783
 59  0206H    15  GETVERSION . . . .  PROCEDURE WORD STACK=0008H               110  766
 35  0000H     2  GINDEX . . . . . .  WORD          562  675
  1  0002H   431  HELP . . . . . .    PROCEDURE STACK=001CH
                  HIGH . . . . . .    BUILTIN           110
700  0178H     1  I. . . . . .        BYTE          716  717  718  721  723  724
136  012CH     1  I. . . . . .        BYTE          137  138  139  141  148  150
550  015FH     1  I. . . . . .        BYTE          582  590  595  598  599  615
342  0136H     1  I. . . . . .        BYTE          346  347  359  360  361  366  369  370  375  376  378  379
                                      381  384  386  395  414  468  470
238  0132H     1  I. . . . . .        BYTE          267  268  271  274  277  302  303  304  306  311  316  321
                                      327  334  336
487  015DH     1  I. . . . . .        BYTE          499  504  509  511  531  533
700  0179H     1  II . . . . . .      BYTE          720  721  722  724  726  728  738  740
154  0388H    42  INC. . . . . .      PROCEDURE BYTE STACK=0010H               370  376  386  414  723
154  0004H     1  INCI . . . . . .    BYTE PARAMETER AUTOMATIC         155  156  157  161  165  168
622  0022H     2  INDEX. . . . . .    WORD          623  625  627  629  632  645  650  656  658  665  675  676
                                      677  679  680
551  001EH     2  INDEX. . . . . .    WORD          569  572  577  602  607  610
 29  0050H     1  INDEX. . . . . .    BYTE
343  000AH     2  INDEX. . . . . .    WORD          349  375  381  384  392  393  394  395  396  397  406  432
238  012FH     1  INDEX. . . . . .    BYTE          239  242  243  244  245  246  248  249  250  252  255  258
                                      259  261  268  271  274  279  280  286  287  288  290  300  309  314
                                      319  325  329  332  339
 11  0000H     2  INFO . . . . . .    WORD PARAMETER           13
  7  0000H     2  INFO . . . . . .    WORD PARAMETER            9
  3  0000H     2  INFO . . . . . .    WORD PARAMETER            5
208  0440H   194  INIT . . . . . .    PROCEDURE STACK=0016H                789
209  0008H     2  INITI. . . . . .    WORD          214  216  219  222  227  233
763  0181H     1  LASTDSEGBYTE . . .  BYTE INITIAL
```

```
  34  010EH     1  LEVEL. . . . . .  BYTE        254  554  557  627  629  632  633  636  639  680  703  794
                                       801  811  835
  32  000FH     1  LEVEL. . . . . .  BYTE MEMBER(SYSBUFF)       375  577  607  627  656  680
  24             LF . . . . . .  LITERALLY          39  120  122  179  199  204  345  484  489  547  615
                                       731  795
 172  0000H     1  LINECNT. . . . . .  BYTE BASED(LINECNTADDR)          177  180  181  182  185  187  191  195
                                       203
 170  0004H     2  LINECNTADDR. . . .  WORD PARAMETER AUTOMATIC         171  172  177  180  181  182  185  187
                                       191  195  203
  15  0024H     1  LISTMODE . . . . .  BYTE         299  323  758  790
  85  026EH    25  MAKEFILE . . . . .  PROCEDURE BYTE STACK=000AH            423  497
  19  0000H     2  MAXB . . . . . .  WORD EXTERNAL(4)             211  350  512
 488  0018H     2  MAXBUF . . . . . .  WORD         515  516  523
 343  0010H     2  MAXBUF . . . . . .  WORD         351  400  401  452  453  460
 209  0006H     2  MAXBUF . . . . . .  WORD         212  215  226  231
      0000H        MEMORY . . . . . .  BYTE ARRAY(0)           32  211  350  436  454  465  512  517  528
   3  0000H        MON1 . . . . . .  PROCEDURE EXTERNAL(0) STACK=0000H          45   49   53   64   98  102
                                       106  115
   7  0000H        MON2 . . . . . .  PROCEDURE BYTE EXTERNAL(1) STACK=0000H          41   57   71   75   79
                                        83   90   96
  11  0000H        MON3 . . . . . .  PROCEDURE WORD EXTERNAL(2) STACK=0000H          60
                  MOVE . . . . . .  BUILTIN          129
 126  0314H    23  MOVEF. . . . . .  PROCEDURE STACK=000AH            259  347  352  392  393  407  421  430
                                       447  490  495  602  632  676  677  751  779
 126  0008H     1  MVCNT. . . . . .  BYTE PARAMETER AUTOMATIC         128  129
  33  0102H    12  NAME . . . . . .  BYTE ARRAY(12)
 552  0162H    14  NAME . . . . . .  BYTE ARRAY(14)          599  601  602  603
  31  0000H    15  NAME . . . . . .  BYTE ARRAY(15) MEMBER(COM)          259  271  274  286  288  304  306
                                       311  316  321  327  629  632  639  694  802  809
  15  0025H     1  NOPAGEMODE . . . .  BYTE         175  299  318  324  731
 684  0004H     1  NOTOKENS . . . . .  BYTE PARAMETER AUTOMATIC         685  687
 551  001CH     2  OFFSET . . . . . .  WORD         553  559  562  569  570
  16  0029H     1  OFFSET . . . . . .  BYTE         679  710  716  743  841
  66  0225H    25  OPENFILE . . . . .  PROCEDURE BYTE STACK=000AH            353  491  780  785
 762  0180H     1  ORET . . . . . .  BYTE         780  781  785  787
 170  03B2H   142  PAGECHECK. . . . .  PROCEDURE BYTE STACK=0010H            582  590  595  615  688  705  733
                                       752  815  819  823  826  828  831
  38  0110H     1  PAGELEN. . . . . .  BYTE         177  767
  15  0028H     1  PAGEMODE . . . . .  BYTE         175  768
 237  0502H   667  PARSE. . . . . .  PROCEDURE BYTE STACK=000EH            757  769
   2  0010H        PLMSTART . . . . .  LABEL PUBLIC          764
 699  1124H   382  PRINT. . . . . .  PROCEDURE STACK=001AH          808
  47  01D3H    16  PRINTCONSOLEBUF. .  PROCEDURE STACK=000AH          120  121  122  179  184  345  484  489
                                       547  585  588  591  603  616  692  694  739  753  759  791  795  822
                                       830
 749  12A2H    93  PROMPT . . . . . .  PROCEDURE BYTE STACK=0014H            798  839
 173  012DH     1  QUIT . . . . . .  BYTE         174  193  206
  40  01B1H    15  READCONSOLE. . . .  PROCEDURE BYTE STACK=0008H
  51  01E3H    16  READCONSOLEBUFF. .  PROCEDURE STACK=000AH          754
  92  0287H    16  READRAND . . . . .  PROCEDURE BYTE STACK=000AH            431  448  708
  77  024EH    16  READRECORD . . . .  PROCEDURE BYTE STACK=000AH            159  217  358  455  504  518  715
 342  0137H     1  RECCNT . . . . . .  BYTE         349  399  420  433  435
  32  000EH     1  RECOFFSET. . . . .  BYTE MEMBER(SYSBUFF)       395  679
  32  000CH     2  RECORD . . . . . .  WORD MEMBER(SYSBUFF)       392  393  394  433  676  677
 136  0128H     1  RESULT . . . . . .  BYTE         137  143  146  152
 621  0174H     1  SAVED. . . . . .  BYTE         623  634  637  654
 621  0175H     1  SAVELEVEL. . . . .  BYTE         636  656
```

```
 488  001AH     2  SAVESIZE . . . . .  WORD        512  515
 343  0012H     2  SAVESIZE . . . . .  WORD        350  351  452
 620  0F4FH   365  SEARCHFILE . . . .  PROCEDURE BYTE STACK=000EH          806
  96  0297H    16  SETDMA . . . . . .  PROCEDURE STACK=000AH          216  230  419  436  454  465  502  517
                                        528  618
 100  02A7H    16  SETRANDREC . . . .  PROCEDURE STACK=000AH          391
 104  02B7H    19  SETUSER. . . . . .  PROCEDURE STACK=000AH          784
                   SHL. . . . . .  .  BUILTIN          436  454  465  517  528
  26               SLASH. . . . . .  LITERALLY       366  369  721
 126  0006H     2  SOURCEADDR . . . .  WORD PARAMETER AUTOMATIC       127  129
 238  0133H     1  STOP . . . . . .  BYTE        263  264  282
 132  0006H     2  STR1ADDR . . . . .  WORD PARAMETER AUTOMATIC       133  134  138  139  141
 132  0004H     2  STR2ADDR . . . . .  WORD PARAMETER AUTOMATIC       133  135  139  141
 134  0000H    12  STRING1. . . . . .  BYTE BASED(STR1ADDR) ARRAY(12)      138  139  141
 135  0000H    12  STRING2. . . . . .  BYTE BASED(STR2ADDR) ARRAY(12)      139  141
  30  0051H    12  SUB. . . . . .  .  BYTE ARRAY(12)
  32  0000H    12  SUBJECT. . . . . .  BYTE ARRAY(12) MEMBER(SYSBUFF)      216  222  347  381  384  407  505
                                        572  602  625  629  632
  32  0000H   128  SYSBUFF. . . . . .  STRUCTURE ARRAY(8) AT        216  222  347  375  381  384  392  393
                                        394  395  407  419  433  502  505  572  577  602  607  625  627  629
                                        632  656  676  677  679  680
  25               TAB. . . . . .  .  LITERALLY       243
  21  0000H   128  TBUFF. . . . . .  BYTE ARRAY(128) EXTERNAL(6)      230  240  242  243  244  245  246
                                        249  252  255  264  265  268  269  271  274  284  361  366  369  375
                                        378  379  381  384  618  718  721  724  751  754  755
  36  010FH     1  TCNT . . . . . .  BYTE        582  590  595  615  688  705  733  752  756  765  796  815
                                        819  823  826  828  831
 750  017FH     1  TEMP . . . . . .  BYTE        752  757  758  760
 701  017CH     3  TEMP . . . . . .  BYTE ARRAY(3)        724  728  739
 117  0006H     1  TERMCODE . . . . .  BYTE PARAMETER AUTOMATIC       118  123
 114  02E1H    14  TERMINATE. . . . .  PROCEDURE STACK=0008H          124  188  848
 685  0176H     1  TOKENCNT1. . . . .  BYTE        686  687  691  694  695
 685  0177H     1  TOKENCNT2. . . . .  BYTE        691
  27               TRUE . . . . . .  LITERALLY       164  193  220  223  234  282  291  308  313  318  323
                                        324  355  363  404  428  444  457  471  478  482  493  500  506  510
                                        520  534  545  567  574  581  608  637  641  642  661  670  711  719
                                        727  768
 104  0004H     1  USERNUM. . . . . .  BYTE PARAMETER AUTOMATIC       105  106
  37  0002H     2  VERSION. . . . . .  WORD        766
  43  01C0H    19  WRITECONSOLE . . .  PROCEDURE STACK=000AH          199  204  596  702  736  816  824  827
                                        829  832
  81  025EH    16  WRITERECORD. . . .  PROCEDURE BYTE STACK=000AH          437  466  511  529
 550  0161H     1  WRITTEN. . . . . .  BYTE        553  579  581  613
```

MODULE INFORMATION:

```
     CODE AREA SIZE     = 12FFH   4863D
     CONSTANT AREA SIZE = 03C3H    963D
     VARIABLE AREA SIZE = 0182H    386D
     MAXIMUM STACK SIZE = 001CH     28D
     1142 LINES READ
     0 PROGRAM ERROR(S)
```

END OF PL/M-86 COMPILATION

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

SYMBOL TABLE OF MODULE SCO
READ FROM FILE HELP.LNK
WRITTEN TO FILE HELP.

BASE   OFFSET TYPE SYMBOL              BASE   OFFSET TYPE SYMBOL

0000H  0110H  PUB  PLMSTART           0000H  0050H  PUB  XDOS
0000H  0050H  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

HELP:  SYMBOLS AND LINES
1000H  0690H  SYM  MEMORY            0000H  0110H  SYM  PLMSTART
1000H  012CH  SYM  LISTMODE          1000H  0120H  SYM  NOPAGEMODE
1000H  012EH  SYM  CREATEMODE        1000H  012FH  SYM  EXTRACTMODE
1000H  0130H  SYM  PAGEMODE          1000H  0131H  SYM  OFFSET
1000H  0132H  SYM  EOD               1000H  0133H  SYM  FCB2
1000H  0157H  SYM  CNT               1000H  0158H  SYM  INDEX
1000H  0159H  SYM  SUB               1000H  0165H  SYM  COM
1000H  0690H  SYM  SYSBUFF           1000H  020AH  SYM  NAME
1000H  0216H  SYM  LEVEL             1000H  0108H  SYM  GINDEX
1000H  0217H  SYM  TCNT              1000H  010AH  SYM  VERSION
1000H  0218H  SYM  PAGELEN           1000H  0219H  SYM  CLEARSCREEN
0000H  0281H  SYM  READCONSOLE       0000H  02C0H  SYM  WRITECONSOLE
STACK  0004H  SYM  CHAR              0000H  02D3H  SYM  PRINTCONSOLEBUF
STACK  0004H  SYM  BUFFADR           0000H  02E3H  SYM  READCONSOLEBUFF
STACK  0004H  SYM  BUFFADR           0000H  02F3H  SYM  DIRECTCONIO
STACK  0004H  SYM  FUNC              0000H  0306H  SYM  GETVERSION
0000H  0315H  SYM  DELETEFILE        STACK  0004H  SYM  FCBADDRESS
0000H  0325H  SYM  OPENFILE          STACK  0004H  SYM  FCBADDRESS
STACK  0004H  BAS  FCB               0000H  033EH  SYM  CLOSEFILE
STACK  0004H  SYM  FCBADDRESS        0000H  034EH  SYM  READRECORD
STACK  0004H  SYM  FCBADDRESS        0000H  035EH  SYM  WRITERECORD
STACK  0004H  SYM  FCBADDRESS        0000H  036EH  SYM  MAKEFILE
STACK  0004H  SYM  FCBADDRESS        STACK  0004H  BAS  FCB
0000H  0387H  SYM  READRAND          STACK  0004H  SYM  FCBADDRESS
0000H  0397H  SYM  SETDMA            STACK  0004H  SYM  DMAADDRESS
0000H  03A7H  SYM  SETRANDREC        STACK  0004H  SYM  FCBADDRESS
0000H  03B7H  SYM  SETUSER           STACK  0004H  SYM  USERNUM
0000H  03CAH  SYM  GETSYSDRIVE       0000H  03E1H  SYM  TERMINATE
0000H  03EFH  SYM  ERROR             STACK  0006H  SYM  TERMCODE
STACK  0004H  SYM  ERRMSGADR         0000H  0414H  SYM  MOVEF
STACK  0008H  SYM  MVCNT             STACK  0006H  SYM  SOURCEADDR
STACK  0004H  SYM  DESTADDR          0000H  042BH  SYM  COMPARE
STACK  0006H  SYM  STR1ADDR          STACK  0004H  SYM  STR2ADDR
STACK  0006H  BAS  STRING1           STACK  0004H  BAS  STRING2
1000H  0233H  SYM  RESULT            1000H  0234H  SYM  I
0000H  0488H  SYM  INC               STACK  0004H  SYM  INC1
```

```
0000H  04B2H  SYM  PAGECHECK
STACK  0004H  BAS  LINECNT
0000H  0540H  SYM  INIT
1000H  010EH  SYM  MAXBUF
1000H  0236H  SYM  ENDINDEX
1000H  0237H  SYM  INDEX
1000H  0239H  SYM  CNT
1000H  023BH  SYM  STOP
0000H  089DH  SYM  CREATEINDEX
1000H  023EH  SYM  I
1000H  0112H  SYM  INDEX
1000H  0116H  SYM  COUNT2
1000H  011AH  SYM  SAVESIZE
0000H  0D3CH  SYM  EXTRACTFILE
1000H  0265H  SYM  I
1000H  011EH  SYM  COUNT2
1000H  0122H  SYM  SAVESIZE
1000H  0266H  SYM  DISPLEVEL
1000H  0268H  SYM  EOD
1000H  0124H  SYM  OFFSET
1000H  0128H  SYM  COUNT
0000H  104FH  SYM  SEARCHFILE
1000H  0279H  SYM  ERROR
1000H  027BH  SYM  FOUND
1000H  027DH  SYM  SAVELEVEL
0000H  11BCH  SYM  DISPLAYTOKENS
1000H  027EH  SYM  TOKENCNT1
0000H  1224H  SYM  PRINT
1000H  0281H  SYM  II
1000H  0283H  SYM  EOD2
0000H  13A2H  SYM  PROMPT
1000H  0288H  SYM  ORET
0000H  02B1H  LIN     40
0000H  02C0H  LIN     42
0000H  02C3H  LIN     45
0000H  02D3H  LIN     47
0000H  02DFH  LIN     50
0000H  02E6H  LIN     53
0000H  02F3H  LIN     55
0000H  0306H  LIN     58
0000H  0309H  LIN     60
0000H  0315H  LIN     62
0000H  0321H  LIN     65
0000H  0328H  LIN     69
0000H  0333H  LIN     71
0000H  033EH  LIN     73
0000H  034EH  LIN     76
0000H  0351H  LIN     79
0000H  035EH  LIN     81
0000H  036EH  LIN     84
0000H  0371H  LIN     88
0000H  037CH  LIN     90
0000H  0387H  LIN     92
0000H  0397H  LIN     95
0000H  039AH  LIN     98
0000H  03A7H  LIN    100
0000H  03B3H  LIN    103
0000H  03BAH  LIN    106
0000H  03CAH  LIN    109
0000H  03D8H  LIN    111

STACK  0004H  SYM  LINECNTADDR
1000H  0235H  SYM  QUIT
1000H  010CH  SYM  BUFSIZE
1000H  0110H  SYM  INIT1
0000H  0602H  SYM  PARSE
1000H  0238H  SYM  BEGIN
1000H  023AH  SYM  I
1000H  023CH  SYM  BRACKET
1000H  023DH  SYM  CNT
1000H  023FH  SYM  RECCNT
1000H  0114H  SYM  COUNT
1000H  0118H  SYM  MAXBUF
1000H  0240H  SYM  FCB3
1000H  0264H  SYM  ENDINDEX
1000H  011CH  SYM  COUNT
1000H  0120H  SYM  MAXBUF
0000H  0EE8H  SYM  DISPLAYIND
1000H  0267H  SYM  I
1000H  0269H  SYM  WRITTEN
1000H  0126H  SYM  INDEX
1000H  026AH  SYM  NAME
1000H  0278H  SYM  EOD
1000H  027AH  SYM  CNT
1000H  027CH  SYM  SAVED
1000H  012AH  SYM  INDEX
STACK  0004H  SYM  NOTOKENS
1000H  027FH  SYM  TOKENCNT2
1000H  0280H  SYM  I
1000H  0282H  SYM  CHAR
1000H  0284H  SYM  TEMP
1000H  0287H  SYM  TEMP
1000H  0289H  SYM  LASTDSEGBYTE
0000H  02B4H  LIN     41
0000H  02C0H  LIN     43
0000H  02CFH  LIN     46
0000H  02D6H  LIN     49
0000H  02E3H  LIN     51
0000H  02EFH  LIN     54
0000H  02F6H  LIN     57
0000H  0306H  LIN     59
0000H  0315H  LIN     61
0000H  0318H  LIN     64
0000H  0325H  LIN     66
0000H  032FH  LIN     70
0000H  033EH  LIN     72
0000H  0341H  LIN     75
0000H  034EH  LIN     77
0000H  035EH  LIN     80
0000H  0361H  LIN     83
0000H  036EH  LIN     85
0000H  0378H  LIN     89
0000H  0387H  LIN     91
0000H  038AH  LIN     94
0000H  0397H  LIN     96
0000H  03A3H  LIN     99
0000H  03AAH  LIN    102
0000H  03B7H  LIN    104
0000H  03C6H  LIN    107
0000H  03CDH  LIN    110
0000H  03DCH  LIN    112
```

```
0000H  03E1H  LIN  113      0000H  03E1H  LIN  114
0000H  03E4H  LIN  115      0000H  03E0H  LIN  116
0000H  03EFH  LIN  117      0000H  03F2H  LIN  120
0000H  03F9H  LIN  121      0000H  03FFH  LIN  122
0000H  0406H  LIN  123      0000H  040DH  LIN  124
0000H  0410H  LIN  125      0000H  0414H  LIN  126
0000H  0417H  LIN  129      0000H  0427H  LIN  130
0000H  042BH  LIN  131      0000H  042BH  LIN  132
0000H  042EH  LIN  137      0000H  0438H  LIN  138
0000H  045FH  LIN  139      0000H  0456H  LIN  141
0000H  046AH  LIN  143      0000H  046FH  LIN  144
0000H  0471H  LIN  146      0000H  0476H  LIN  148
0000H  047BH  LIN  150      0000H  047FH  LIN  151
0000H  0481H  LIN  152      0000H  0488H  LIN  153
0000H  0488H  LIN  154      0000H  048BH  LIN  156
0000H  0493H  LIN  157      0000H  0497H  LIN  159
0000H  04A2H  LIN  164      0000H  04A7H  LIN  165
0000H  04ABH  LIN  168      0000H  04B2H  LIN  169
0000H  04B2H  LIN  170      0000H  04B5H  LIN  174
0000H  04BAH  LIN  175      0000H  04C7H  LIN  177
0000H  04D6H  LIN  179      0000H  04DDH  LIN  180
0000H  04E3H  LIN  181      0000H  04EBH  LIN  182
0000H  04F6H  LIN  183      0000H  04F8H  LIN  184
0000H  04FFH  LIN  185      0000H  0507H  LIN  187
0000H  0513H  LIN  188      0000H  0516H  LIN  189
0000H  0518H  LIN  191      0000H  0520H  LIN  193
0000H  0525H  LIN  195      0000H  0528H  LIN  197
0000H  052DH  LIN  201      0000H  052DH  LIN  203
0000H  0533H  LIN  204      0000H  0539H  LIN  206
0000H  0540H  LIN  207      0000H  0540H  LIN  208
0000H  0543H  LIN  211      0000H  054CH  LIN  212
0000H  054FH  LIN  213      0000H  0554H  LIN  214
0000H  055AH  LIN  215      0000H  0570H  LIN  216
0000H  0585H  LIN  217      0000H  0590H  LIN  219
0000H  059CH  LIN  220      0000H  05A6H  LIN  222
0000H  05B7H  LIN  223      0000H  05BCH  LIN  224
0000H  05C5H  LIN  226      0000H  05CBH  LIN  227
0000H  05D0H  LIN  229      0000H  05D2H  LIN  230
0000H  05D9H  LIN  231      0000H  05EAH  LIN  233
0000H  05F6H  LIN  234      0000H  0600H  LIN  236
0000H  0602H  LIN  237      0000H  0605H  LIN  239
0000H  060AH  LIN  240      0000H  0614H  LIN  242
0000H  0622H  LIN  243      0000H  062DH  LIN  245
0000H  063AH  LIN  246      0000H  063FH  LIN  247
0000H  0645H  LIN  248      0000H  064AH  LIN  249
0000H  066EH  LIN  250      0000H  0672H  LIN  251
0000H  0674H  LIN  252      0000H  0681H  LIN  254
0000H  0687H  LIN  255      0000H  068CH  LIN  256
0000H  068EH  LIN  257      0000H  0693H  LIN  258
0000H  06A0H  LIN  259      0000H  06B8H  LIN  260
0000H  06BEH  LIN  261      0000H  06C4H  LIN  262
0000H  06C9H  LIN  263      0000H  06D3H  LIN  264
0000H  06F2H  LIN  265      0000H  06FCH  LIN  267
0000H  0701H  LIN  268      0000H  074BH  LIN  269
0000H  0755H  LIN  271      0000H  0761H  LIN  272
0000H  0761H  LIN  274      0000H  0780H  LIN  276
0000H  0784H  LIN  277      0000H  0788H  LIN  278
0000H  078BH  LIN  279      0000H  0793H  LIN  280
0000H  079EH  LIN  282      0000H  07A3H  LIN  283
0000H  07A5H  LIN  284      0000H  07B2H  LIN  286
```

```
0000H  07C4H  LIN   287
0000H  07DAH  LIN   289
0000H  07E2H  LIN   291
0000H  07E9H  LIN   295
0000H  07F0H  LIN   299
0000H  0808H  LIN   302
0000H  0815H  LIN   304
0000H  0829H  LIN   308
0000H  0830H  LIN   310
0000H  0840H  LIN   313
0000H  084CH  LIN   315
0000H  085CH  LIN   321
0000H  0871H  LIN   324
0000H  0878H  LIN   326
0000H  0888H  LIN   332
0000H  0891H  LIN   336
0000H  0898H  LIN   339
0000H  089DH  LIN   341
0000H  08A7H  LIN   346
0000H  08C8H  LIN   348
0000H  08DCH  LIN   350
0000H  08E8H  LIN   352
0000H  0901H  LIN   355
0000H  0910H  LIN   358
0000H  0934H  LIN   360
0000H  0957H  LIN   363
0000H  095EH  LIN   366
0000H  0973H  LIN   369
0000H  0996H  LIN   371
0000H  099CH  LIN   373
0000H  09C8H  LIN   376
0000H  09D4H  LIN   378
0000H  0A0AH  LIN   381
0000H  0A1FH  LIN   384
0000H  0A49H  LIN   387
0000H  0A4FH  LIN   389
0000H  0A62H  LIN   392
0000H  0A95H  LIN   394
0000H  0AAAH  LIN   396
0000H  0ABCH  LIN   399
0000H  0AC9H  LIN   401
0000H  0AE1H  LIN   404
0000H  0AEDH  LIN   406
0000H  0B18H  LIN   408
0000H  0B21H  LIN   414
0000H  0B2EH  LIN   418
0000H  0B35H  LIN   420
0000H  0B47H  LIN   422
0000H  0B59H  LIN   425
0000H  0B6AH  LIN   427
0000H  0B7EH  LIN   430
0000H  0B96H  LIN   432
0000H  0BBAH  LIN   434
0000H  0BD4H  LIN   436
0000H  0BEFH  LIN   439
0000H  0C00H  LIN   441
0000H  0C11H  LIN   443
0000H  0C25H  LIN   446
0000H  0C3AH  LIN   448
0000H  0C49H  LIN   450

0000H  07CCH  LIN   288
0000H  07DEH  LIN   290
0000H  07E7H  LIN   293
0000H  07EDH  LIN   297
0000H  07FEH  LIN   300
0000H  080BH  LIN   303
0000H  0822H  LIN   306
0000H  082EH  LIN   309
0000H  0830H  LIN   311
0000H  0845H  LIN   314
0000H  084CH  LIN   316
0000H  086CH  LIN   323
0000H  0876H  LIN   325
0000H  0878H  LIN   327
0000H  088CH  LIN   334
0000H  0895H  LIN   337
0000H  089DH  LIN   340
0000H  08A0H  LIN   345
0000H  08B3H  LIN   347
0000H  08D1H  LIN   349
0000H  08E5H  LIN   351
0000H  08F6H  LIN   353
0000H  090BH  LIN   357
0000H  092FH  LIN   359
0000H  094AH  LIN   361
0000H  095CH  LIN   364
0000H  096EH  LIN   368
0000H  098FH  LIN   370
0000H  099AH  LIN   372
0000H  09ACH  LIN   375
0000H  09CFH  LIN   377
0000H  0A00H  LIN   379
0000H  0A1FH  LIN   382
0000H  0A3FH  LIN   386
0000H  0A4DH  LIN   388
0000H  0A5BH  LIN   391
0000H  0A78H  LIN   393
0000H  0AA3H  LIN   395
0000H  0AB1H  LIN   397
0000H  0AC0H  LIN   400
0000H  0AD7H  LIN   403
0000H  0AEBH  LIN   405
0000H  0AFFH  LIN   407
0000H  0B1FH  LIN   412
0000H  0B28H  LIN   417
0000H  0B2EH  LIN   419
0000H  0B39H  LIN   421
0000H  0B4EH  LIN   423
0000H  0B63H  LIN   426
0000H  0B74H  LIN   428
0000H  0B8CH  LIN   431
0000H  0BA6H  LIN   433
0000H  0BC1H  LIN   435
0000H  0BE4H  LIN   437
0000H  0BF9H  LIN   440
0000H  0C0AH  LIN   442
0000H  0C1BH  LIN   444
0000H  0C2CH  LIN   447
0000H  0C44H  LIN   449
0000H  0C55H  LIN   451
```

```
0000H  0C5BH  LIN   452      0000H  0C61H  LIN   453
0000H  0C77H  LIN   454      0000H  0C87H  LIN   455
0000H  0C92H  LIN   457      0000H  0C97H  LIN   458
0000H  0C99H  LIN   460      0000H  0C9FH  LIN   461
0000H  0CA3H  LIN   463      0000H  0CA5H  LIN   464
0000H  0CB5H  LIN   465      0000H  0CC5H  LIN   466
0000H  0CD0H  LIN   468      0000H  0CDAH  LIN   469
0000H  0CE1H  LIN   470      0000H  0CEBH  LIN   471
0000H  0CF5H  LIN   473      0000H  0CFCH  LIN   474
0000H  0CFFH  LIN   475      0000H  0D0AH  LIN   477
0000H  0D14H  LIN   478      0000H  0D1EH  LIN   480
0000H  0D29H  LIN   482      0000H  0D33H  LIN   484
0000H  0D3AH  LIN   485      0000H  0D3CH  LIN   486
0000H  0D3FH  LIN   489      0000H  0D46H  LIN   490
0000H  0D54H  LIN   491      0000H  0D5FH  LIN   493
0000H  0D69H  LIN   495      0000H  0D77H  LIN   496
0000H  0D7EH  LIN   497      0000H  0D89H  LIN   499
0000H  0D93H  LIN   500      0000H  0D9DH  LIN   502
0000H  0DA4H  LIN   503      0000H  0DA9H  LIN   504
0000H  0DC6H  LIN   505      0000H  0DCDH  LIN   506
0000H  0DD2H  LIN   507      0000H  0DD4H  LIN   508
0000H  0DD9H  LIN   509      0000H  0DE0H  LIN   510
0000H  0DE5H  LIN   511      0000H  0DEFH  LIN   512
0000H  0DF8H  LIN   513      0000H  0E04H  LIN   514
0000H  0E0AH  LIN   515      0000H  0E10H  LIN   516
0000H  0E26H  LIN   517      0000H  0E36H  LIN   518
0000H  0E41H  LIN   520      0000H  0E46H  LIN   521
0000H  0E48H  LIN   523      0000H  0E4EH  LIN   524
0000H  0E52H  LIN   526      0000H  0E54H  LIN   527
0000H  0E64H  LIN   528      0000H  0E74H  LIN   529
0000H  0E7FH  LIN   531      0000H  0E89H  LIN   532
0000H  0E90H  LIN   533      0000H  0E9AH  LIN   534
0000H  0EA4H  LIN   536      0000H  0EA8H  LIN   537
0000H  0EAEH  LIN   538      0000H  0EB9H  LIN   540
0000H  0EC3H  LIN   542      0000H  0ECEH  LIN   544
0000H  0ED5H  LIN   545      0000H  0EDFH  LIN   547
0000H  0EE6H  LIN   548      0000H  0EE8H  LIN   549
0000H  0EE8H  LIN   553      0000H  0EF7H  LIN   554
0000H  0EFFH  LIN   555      0000H  0F03H  LIN   557
0000H  0F0AH  LIN   559      0000H  0F10H  LIN   560
0000H  0F12H  LIN   562      0000H  0F18H  LIN   564
0000H  0F1EH  LIN   565      0000H  0F20H  LIN   567
0000H  0F25H  LIN   569      0000H  0F2BH  LIN   570
0000H  0F31H  LIN   571      0000H  0F3DH  LIN   572
0000H  0F4EH  LIN   574      0000H  0F53H  LIN   575
0000H  0F55H  LIN   577      0000H  0F6CH  LIN   579
0000H  0F75H  LIN   581      0000H  0F7AH  LIN   582
0000H  0F84H  LIN   583      0000H  0F8BH  LIN   585
0000H  0F90H  LIN   586      0000H  0F90H  LIN   588
0000H  0F97H  LIN   590      0000H  0FA1H  LIN   591
0000H  0FA8H  LIN   593      0000H  0FB6H  LIN   595
0000H  0FC0H  LIN   596      0000H  0FC6H  LIN   598
0000H  0FD2H  LIN   599      0000H  0FDDH  LIN   600
0000H  0FE3H  LIN   601      0000H  0FE8H  LIN   602
0000H  1001H  LIN   603      0000H  1008H  LIN   604
0000H  100CH  LIN   605      0000H  100EH  LIN   607
0000H  1022H  LIN   608      0000H  1027H  LIN   610
0000H  1028H  LIN   612      0000H  102EH  LIN   613
0000H  1035H  LIN   615      0000H  103FH  LIN   616
0000H  1046H  LIN   618      0000H  1040H  LIN   619
```

```
0000H  104FH  LIN   620
0000H  1065H  LIN   624
0000H  108DH  LIN   627
0000H  10B4H  LIN   630
0000H  10E1H  LIN   633
0000H  10F2H  LIN   636
0000H  10FDH  LIN   639
0000H  1110H  LIN   642
0000H  1124H  LIN   645
0000H  112DH  LIN   648
0000H  112FH  LIN   654
0000H  114AH  LIN   661
0000H  1151H  LIN   665
0000H  1158H  LIN   672
0000H  115FH  LIN   675
0000H  117FH  LIN   677
0000H  119EH  LIN   679
0000H  11B7H  LIN   682
0000H  11BCH  LIN   684
0000H  11C4H  LIN   687
0000H  11E5H  LIN   689
0000H  11FCH  LIN   692
0000H  1209H  LIN   694
0000H  121EH  LIN   697
0000H  1224H  LIN   699
0000H  122DH  LIN   703
0000H  123DH  LIN   705
0000H  1250H  LIN   708
0000H  1265H  LIN   711
0000H  1276H  LIN   714
0000H  129EH  LIN   716
0000H  12C4H  LIN   718
0000H  12DAH  LIN   720
0000H  1312H  LIN   722
0000H  131DH  LIN   724
0000H  1337H  LIN   726
0000H  1345H  LIN   728
0000H  1361H  LIN   731
0000H  1378H  LIN   734
0000H  1384H  LIN   738
0000H  1392H  LIN   740
0000H  139AH  LIN   743
0000H  13A2H  LIN   749
0000H  13B4H  LIN   752
0000H  13C5H  LIN   754
0000H  13D8H  LIN   756
0000H  13E3H  LIN   758
0000H  13FAH  LIN   760
0000H  0102H  LIN   764
0000H  011FH  LIN   766
0000H  012AH  LIN   768
0000H  0135H  LIN   770
0000H  013FH  LIN   773
0000H  0148H  LIN   776
0000H  014EH  LIN   779
0000H  0166H  LIN   781
0000H  0173H  LIN   784
0000H  0183H  LIN   787
0000H  0190H  LIN   790
0000H  01A0H  LIN   792

0000H  1052H  LIN   623
0000H  1079H  LIN   625
0000H  1098H  LIN   629
0000H  10BEH  LIN   632
0000H  10E9H  LIN   634
0000H  10F8H  LIN   637
0000H  1118H  LIN   641
0000H  1122H  LIN   643
0000H  1128H  LIN   646
0000H  112FH  LIN   652
0000H  1136H  LIN   656
0000H  114FH  LIN   663
0000H  1155H  LIN   668
0000H  1158H  LIN   673
0000H  1166H  LIN   676
0000H  1199H  LIN   678
0000H  11B0H  LIN   680
0000H  11BCH  LIN   683
0000H  11BFH  LIN   686
0000H  11DBH  LIN   688
0000H  11EEH  LIN   691
0000H  1203H  LIN   693
0000H  121AH  LIN   695
0000H  1220H  LIN   698
0000H  1227H  LIN   702
0000H  1234H  LIN   704
0000H  1247H  LIN   706
0000H  125BH  LIN   710
0000H  126FH  LIN   712
0000H  127BH  LIN   715
0000H  12A6H  LIN   717
0000H  12D5H  LIN   719
0000H  12DFH  LIN   721
0000H  1316H  LIN   723
0000H  1335H  LIN   725
0000H  133EH  LIN   727
0000H  1350H  LIN   729
0000H  1371H  LIN   733
0000H  137DH  LIN   736
0000H  1388H  LIN   739
0000H  1397H  LIN   742
0000H  139FH  LIN   744
0000H  13A5H  LIN   751
0000H  13BEH  LIN   753
0000H  13CDH  LIN   755
0000H  13DDH  LIN   757
0000H  13F3H  LIN   759
0000H  13FFH  LIN   761
0000H  0115H  LIN   765
0000H  0125H  LIN   767
0000H  012FH  LIN   769
0000H  013CH  LIN   772
0000H  0141H  LIN   774
0000H  0148H  LIN   777
0000H  015CH  LIN   780
0000H  016DH  LIN   783
0000H  0179H  LIN   785
0000H  018DH  LIN   789
0000H  0199H  LIN   791
0000H  01A7H  LIN   794
```

```
0000H  01ACH  LIN    795              0000H  0193H  LIN    796
0000H  01B8H  LIN    797              0000H  01BBH  LIN    798
0000H  01C1H  LIN    800              0000H  01C8H  LIN    801
0000H  01D0H  LIN    802              0000H  01E2H  LIN    806
0000H  01E9H  LIN    808              0000H  01ECH  LIN    809
0000H  01FEH  LIN    811              0000H  0201H  LIN    813
0000H  0201H  LIN    815              0000H  020BH  LIN    816
0000H  0211H  LIN    817              0000H  021AH  LIN    819
0000H  0224H  LIN    820              0000H  022DH  LIN    822
0000H  0234H  LIN    823              0000H  023EH  LIN    824
0000H  0244H  LIN    825              0000H  024BH  LIN    826
0000H  0255H  LIN    827              0000H  025BH  LIN    828
0000H  0265H  LIN    829              0000H  026BH  LIN    830
0000H  0272H  LIN    831              0000H  027CH  LIN    832
0000H  0282H  LIN    835              0000H  0287H  LIN    837
0000H  0293H  LIN    839              0000H  0296H  LIN    840
0000H  0296H  LIN    841              0000H  02A0H  LIN    842
0000H  02A2H  LIN    844              0000H  02ACH  LIN    848
0000H  02AFH  LIN    849

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

MODULE START ADDRESS  PARAGRAPH = 0000H  OFFSET = 0102H
SEGMENT MAP

START     STOP      LENGTH ALIGN NAME             CLASS

00000H    013FEH     13FFH   G   CODE             CODE
10000H    10107H     0108H   G   DATS             DATA
10108H    10289H     0182H   W   DATA             DATA
1028AH    102C5H     003CH   W   STACK            STACK
102C6H    10688H     03C3H   W   CONST            CONST
10690H    10690H     0000H   G   ??SEG
10690H    10690H     0000H   W   MEMORY           MEMORY


GROUP MAP

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