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



```
   1          GENCMD:
              DO;
              /*   CP/M 8086 CMD file generator

                  COPYRIGHT (C) 1983
                  DIGITAL RESEARCH
                  BOX 579 PACIFIC GROVE
                  CALIFORNIA 93950

                  */

              /**** The following commands were used on the VAX to compile GENCMD:

                  $ util := GENCMD
                  $ ccpmsetup ! set up environment
                  $ plm86 'util'.plm xref 'pl' optimize(3) debug
                  $ link86 f1: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'

              **** Followed on the micro by:
                  A>vax gencmd.h86 $fans
                  A>gencmd gencmd data[b1000 m86 xfff]

              ****/

   2   1      DECLARE
              digital$code literally '0081h',  /*DR code record */
              digital$data literally '0082h',  /* DR data record */
              digital02 literally '0085h',   /* DR 02 records */
              paragraph literally '16',
              ex literally '12',          /* extent */
              nr literally '32',          /* current record */
              maxb address external,
              fcba(33) byte external,     /* DEFAULT FILE CONTROL BLOCK */
              buffa(128) byte external; /* DEFAULT BUFFER ADDRESS */


   3   1      DECLARE COPYRIGHT(*) BYTE DATA
                  (' COPYRIGHT (C) 1983, DIGITAL RESEARCH ');

   4   1      MON1: PROCEDURE(F,A) EXTERNAL;
   5   2          DECLARE F BYTE, A ADDRESS;
   6   2          END MON1;

   7   1      MON2: PROCEDURE(F,A) BYTE EXTERNAL;
   8   2          DECLARE F BYTE, A ADDRESS;
   9   2          END MON2;

  10   1      DECLARE SP ADDRESS;
```

```
11   1      BOOT: PROCEDURE;
12   2          call monl (0,0);
13   2          END BOOT;

14   1      declare segmts(11) structure (name(5) byte,begin$add address)
               initial ('CODE ',00h,'DATA ',0ffffh,'EXTRA',uffffh,'STACK',0,
                 'X1   ',0,'X2   ',0,'X3   ',0,'X4   ',0,'8080 ',0,'NZERO',0,
                 'NHEAD',0);



15   1      declare header (15) structure
              (typseg byte,file$length address,absolute$add address,
                minimum$mem address,
                maximum$mem address) initial (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
               0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
               0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);


16   1      plmstart: PROCEDURE public;
17   2          DECLARE FCB (33) BYTE AT (.FCBA),
                   DFCBA LITERALLY 'FCBA';
18   2          DECLARE BUFFER (128) BYTE AT (.BUFFA),
                    DBUFF LITERALLY 'BUFFA';
19   2          DECLARE SFCB(33) BYTE,  /* SOURCE FILE CONTROL BLOCK */
                    BSIZE LITERALLY '1024',
                    EOFILE LITERALLY '1AH',
                    SBUFF(BSIZE) BYTE,  /* SOURCE FILE BUFFER */
                    RFLAG BYTE,         /* READER FLAG */
                    SBP ADDRESS;        /* SOURCE FILE BUFFER POINTER */
20   2          declare tbp address;     /* pointer to command tail  */
21   2          declare count$command$tail byte at (.buffa);
22   2          declare (t8080,nozero) byte;



23   2      DECLARE
               TRUE LITERALLY '1',
               FALSE LITERALLY '0',
               FOREVER LITERALLY 'WHILE TRUE',
               CR LITERALLY '13',
               LF LITERALLY '10',
               WHAT LITERALLY '63';

24   2      PRINTCHAR: PROCEDURE(CHAR);
25   3          DECLARE CHAR BYTE;
26   3          CALL MONI(2,CHAR);
27   3          END PRINTCHAR;

28   2      CRLF: PROCEDURE;
29   3          CALL PRINTCHAR(CR);
30   3          CALL PRINTCHAR(LF);
31   3          END CRLF;

32   2      PRINTNIB: PROCEDURE(N);
33   3          DECLARE N BYTE;
```

```
34   3              IF N > 9 THEN CALL PRINTCHAR(N+'A'-10); ELSE
36   3              CALL PRINTCHAR(N+'0');
37   3          END PRINTNIB;

38   2      PRINTHEX: PROCEDURE(B);
39   3          DECLARE B BYTE;
40   3              CALL PRINTNIB(SHR(B,4)); CALL PRINTNIB(B AND 0FH);
42   3          END PRINTHEX;

43   2      PRINTADDR: PROCEDURE(A);
44   3          DECLARE A ADDRESS;
45   3          CALL PRINTHEX(HIGH(A)); CALL PRINTHEX(LOW(A));
47   3          END PRINTADDR;

48   2      PRINTM: PROCEDURE(A);
49   3          DECLARE A ADDRESS;
50   3          CALL MON1(9,A);
51   3          END PRINTM;

52   2      PRINT: PROCEDURE(A);
53   3          DECLARE A ADDRESS;
                /* PRINT THE STRING STARTING AT ADDRESS A UNTIL THE
                NEXT DOLLAR SIGN IS ENCOUNTERED WITH PRECEDING CRLF */
54   3          CALL CRLF;
55   3          CALL PRINTM(A);
56   3          END PRINT;

57   2      declare    mbuffadr address,
                LA ADDRESS;  /* CURRENT LOAD ADDRESS */
58   2          declare head byte;

59   2      PERROR: PROCEDURE(A);
                /* PRINT ERROR MESSAGE */
60   3          DECLARE A ADDRESS;
61   3          CALL PRINT(.('ERROR: $'));
62   3          CALL PRINTM(A);
63   3          CALL PRINTM(.(', LOAD ADDRESS $'));
64   3          CALL PRINTADDR(LA);
65   3          CALL BOOT;
66   3          END PERROR;


67   2      diskerror: procedure;
68   3              call perror(.('DISK WRITE$'));
69   3          end diskerror;

70   2      DECLARE DCNT BYTE;


71   2      setdma: procedure(a);
72   3          declare a address;
73   3          call mon1 (26,a);
74   3          end setdma;

75   2      OPEN: PROCEDURE(FCB);
76   3          DECLARE FCB ADDRESS;
77   3          DCNT = MON2(15,FCB);
```

```
 78   3          END OPEN;

 79   2      CLOSE: PROCEDURE(FCB);
 80   3          DECLARE FCB ADDRESS;
 81   3          DCNT = MON2(16,FCB);
 82   3          END CLOSE;

 83   2      SEARCH: PROCEDURE(FCB);
 84   3          DECLARE FCB ADDRESS;
 85   3          DCNT = MON2(17,FCB);
 86   3          END SEARCH;

 87   2      SEARCHN: PROCEDURE;
 88   3          DCNT = MON2(18,0);
 89   3          END SEARCHN;

 90   2      DELETE: PROCEDURE(FCB);
 91   3          DECLARE FCB ADDRESS;
 92   3          CALL MON1(19,FCB);
 93   3          END DELETE;

 94   2      DISKREAD: PROCEDURE(FCB) BYTE;
 95   3          DECLARE FCB ADDRESS;
 96   3          RETURN MON2(20,FCB);
 97   3          END DISKREAD;

 98   2      DISKWRITE: PROCEDURE(FCB) BYTE;
 99   3          DECLARE FCB ADDRESS;
100   3          RETURN MON2(21,FCB);
101   3          END DISKWRITE;

102   2      MAKE: PROCEDURE(FCB);
103   3          DECLARE FCB ADDRESS;
104   3          DCNT = MON2(22,FCB);
105   3          END MAKE;

106   2      RENAME: PROCEDURE(FCB);
107   3          DECLARE FCB ADDRESS;
108   3          CALL MON1(23,FCB);
109   3          END RENAME;

110   2          MOVE: PROCEDURE(S,D,N);
111   3              DECLARE (S,D) ADDRESS, N BYTE,
                      A BASED S BYTE, B BASED D BYTE;
112   3                  DO WHILE (N:=N-1) <> 255;
113   4                  B = A; S=S+1; D=D+1;
116   4                  END;
117   3              END MOVE;


118   2          declare char byte;



119   2      comline$error: procedure;
120   3          declare i byte;
```

```
121  3          call crlf;
122  3          do i = 1 to tbp;
123  4             call printchar (buffer(i));
124  4             end;
125  3          call printchar ('?');
126  3          call crlf;
127  3          call boot;
128  3          end comline$error;



129  2       retchar: procedure byte;
                /*  get another character from command tail */
130  3          if (tbp :=tbp+1) <= count$command$tail then
131  3             return buffer(tbp);
132  3          else return (0dh);
133  3          end retchar;

134  2       tran: procedure(b) byte;
135  3          declare b byte;
136  3          if b < ' ' then return 0dh;  /* non-graphic */
138  3          if b - 'a' < ('z' - 'a') then
139  3             b = b and 101$1111b;    /* upper case */
140  3          return b;
141  3          end tran;


142  2       next$non$blank: procedure;
143  3          char=tran(retchar);
144  3          do while char= ' ';
145  4          char= tran(retchar);
146  4          end;
147  3          end next$non$blank;


148  2          CHECK$ONE$HEX: PROCEDURE (h) BYTE;
                   /* READ ONE HEX CHARACTER FROM THE INPUT */
149  3             DECLARE H BYTE;
150  3             IF H  - '0' <= 9 THEN RETURN H - '0';
152  3             IF H - 'A' > 5 THEN
153  3             return (0ffh);
154  3             RETURN H - 'A' + 10;
155  3             END CHECK$ONE$HEX;



156  2          MAKE$DOUBLE: PROCEDURE(H,L) ADDRESS;
                   /* CREATE A BOUBLE BYTE VALUE FROM TWO SINGLE BYTES */
157  3             DECLARE (H,L) BYTE;
158  3             RETURN SHL(DOUBLE(H),8) OR L;
159  3             END MAKE$DOUBLE;



160  2       delimiter:  procedure byte;    /*   logical     */
161  3          declare i byte;
```

```
162   3          declare del (*) byte data (0dh,'[], ');
163   3          do i = 0 to last(del);
164   4              if char = del(i) then return true;
166   4          end;
167   3          return false;
168   3          end delimiter;


169   2      get$num:  procedure address;
170   3          declare paradd address;
171   3          paradd = 0;
172   3          char = retchar;
173   3          do while not delimiter ;
174   4             if (char:=check$one$hex(char)) = 0ffh then
175   4             call comline$error;   else
176   4             paradd = paradd * 16 + char;
177   4             char = retchar;
178   4             end;

179   3           return paradd;
180   3          end get$num;




181   2          GETCHAR: PROCEDURE BYTE;
                    /* GET NEXT CHARACTER FROM DISK BUFFER */
182   3              DECLARE I BYTE;
183   3              IF (SBP := SBP+1) <= LAST(SBUFF) THEN
184   3                  RETURN SBUFF(SBP);
                    /* OTHERWISE READ ANOTHER BUFFER FULL */
185   3                  DO SBP = 0 TO LAST(SBUFF) BY 128;
186   4                  IF (I:=DISKREAD(.SFCB)) = 0 THEN
187   4                      CALL MOVE(.buffer,.SBUFF(SBP),80H); ELSE
188   4                      DO;
189   5                      IF I<>1 THEN CALL PERROR(.('DISK READ$'));
191   5                      SBUFF(SBP) = EOFILE;
192   5                      SBP = LAST(SBUFF);
193   5                      END;
194   4                  END;
195   3              SBP = 0; RETURN SBUFF(0);
197   3              END GETCHAR;
198   2      DECLARE
              STACKPOINTER LITERALLY 'STACKPTR';

           /* INTEL HEX FORMAT LOADER */

199   2      RELOC: PROCEDURE;
200   3          DECLARE (RL, CS, RT,K) BYTE;
201   3          declare multi$segments byte;
202   3          DECLARE
                  tabs address,   /* temporary value */
                  TA ADDRESS,     /* TEMP ADDRESS */
                  SA ADDRESS,     /* PARAGRAPH LOAD ADDRESS */
                  FA ADDRESS,     /* FINAL ADDRESS */
                  NB ADDRESS,     /* NUMBER OF BYTES LOADED */
                  nxb byte,       /*  next byte in stream   */
```

```
                segadjst address,  /* segment adjust  */
                seg$length (8) address,    /* length of each segment  */
                write$add address,

                MBUFF based mbuffadr (256) BYTE,
                P BYTE;
203   3       declare high$add address;

204   3       SETMEM: PROCEDURE(B);
                /* set mbuff to b at location la */
205   4           DECLARE (B) BYTE;
206   4           if ((.memory+la) < 0) or ((.memory+la) > maxb) then
207   4           do;
208   5             call print (.('INSUFFICIENT MEMORY TO CREATE CMD FILL $'));
209   5             call boot;
210   5             end;
211   4           MBUFF(LA) = B;
212   4           END SETMEM;


213   3       zero$mem: procedure;
214   4           do while (.memory +la) <maxb and not nozero;
215   5              mbuff(la) = 0;
216   5              la = la +1;
217   5              end;
218   4       end zero$mem;



219   3       DIAGNOSE: PROCEDURE;

220   4       DECLARE M BASED TA BYTE;

221   4       NEWLINE: PROCEDURE;
222   5           CALL CRLF; CALL PRINTADDR(TA); CALL PRINTCHAR(':');
225   5           CALL PRINTCHAR(' ');
226   5           END NEWLINE;

                /* PRINT DIAGNOSTIC INFORMATION AT THE CONSOLE */
227   4           CALL PRINT(.('LOAD  ADDRESS $')); CALL PRINTADDR(TA);
229   4           CALL PRINT(.('ERROR ADDRESS $')); CALL PRINTADDR(LA);

231   4           CALL PRINT(.('BYTES READ:$')); CALL NEWLINE;
233   4           DO WHILE TA < LA;
234   5           IF (LOW(TA) AND 0FH) = 0  THEN CALL NEWLINE;
236   5           CALL PRINTHEX(MBUFF(TA)); TA=TA+1;
238   5           CALL PRINTCHAR(' ');
239   5           END;
240   4       CALL CRLF;
241   4       CALL BOOT;
242   4       END DIAGNOSE;
243   3     write$record: procedure;

244   4       call setdma(write$add);
245   4       if diskwrite(.fcba) <> 0 then call diskerror;
247   4       p = p+1;
248   4       end write$record;
```

```
249   3       empty$buffers:  procedure;
250   4            write$add = .memory;
251   4            do while write$add+127 <= (.memory+fa);
252   5               call write$record;
253   5               write$add = write$add+128;
254   5               end;
255   4            if not multi$segments then
256   4            do;
257   5               call write$record;
258   5               return;
259   5            end;
260   4            call move (write$add,.memory,(la:=.memory+fa+1-write$add));
261   4            end empty$buffers;



262   3           READHEX: PROCEDURE BYTE;
                      /* READ ONE HEX CHARACTER FROM THE INPUT */
263   4               declare khex byte;
264   4           if  (khex := check$one$hex(getchar)) <> 0ffh then return khex;
                     else
266   4                    DO; CALL PRINT(.('INVALID HEX DIGIT$'));
268   5                    CALL DIAGNOSE;
269   5                   end;
270   4            end readhex;

271   3           READBYTE: PROCEDURE BYTE;
                      /* READ TWO HEX DIGITS */
272   4               RETURN SHL(READHEX,4) OR READHEX;
273   4               END READBYTE;

274   3           READCS: PROCEDURE BYTE;
                      /* READ BYTE WHILE COMPUTING CHECKSUM */
275   4               DECLARE B BYTE;
276   4               CS = CS + (B := READBYTE);
277   4               RETURN B;
278   4               END READCS;


279   3       hex$input: procedure;
280   4            if rt = 2 or rt > 84h then
281   4            segadjst = make$double(readcs,readcs); else

282   4               do;
                                 /* PROCESS EACH BYTE */
283   5                   DO WHILE (RL := RL - 1) <> 255;
284   6                   CALL SETMEM(READCS); LA = LA+1;
286   6                   END;
287   5               IF LA > FA THEN FA = LA - 1;
289   5               end;


                        /* NOW READ CHECKSUM AND COMPARE */
```

```
290  4                      IF CS + READBYTE <> 0 THEN
291  4                      DO; CALL PRINT(.('CHECK SUM ERROR $'));
293  5                          CALL DIAGNOSE;
294  5                          END;
295  4              end hex$input;


296  3      get$buffer$len: procedure;
297  4          multi$segments = true;
298  4          if rt = 84h then rt = 83h;
300  4          else if rt = 83h then rt = 84h;
               if seg$length (rt-81h) <= (high$add:=la+rl-1) then
303  4              do;
304  5              if high$add=0 then high$add = 1;
306  5              seg$length (rt-81h) = high$add;
307  5              header(rt-81h).typseg = rt-80h;
308  5              end;
309  4          end get$buffer$len;

310  3      compute$la:  procedure (j) address;
311  4          declare j byte;
312  4          return (la and 000Fh)+shl((sa-segmts(j).begin$add),4);
313  4          end compute$la;



               /* INITIALIZE */
314  3          SA, FA, NB = 0;
315  3          P = 0; /* PARAGRAPH COUNT */
316  3          SBUFF(0) = EOFILE;
317  3          fcb(nr) = 0;
318  3          if head then fcb(nr) = 1;
320  3          multi$segments = false;
321  3          segadjst = 0;
322  3          do k= 0 to 7;
323  4             seglength(k) = 0;
324  4             end;

325  3          call zero$mem;

326  3          ta=0;
327  3          la=1;
               /* READ RECORDS UNTIL :00XXXX IS ENCOUNTERED */

328  3              DO FOREVER;
                   /* SCAN THE : */
329  4                  DO WHILE (nxb$:=getchar) <> ':';
330  5                    if nxb = eofile then go to second;
                       /* MAY BE THE END OF TAPE */
332  5                  END;

                   /* SET CHECK SUM TO ZERO, AND SAVE THE RECORD LENGTH */
333  4                  CS = 0;
334  4                        nb = nb +(rl$:=readcs);

335  4                        TA, LA = MAKE$DOUBLE(READCS,READCS) ;
```

```
336  4                    sa = segadjst + shr(la,4);


                          /* READ THE RECORD TYPE  */

                          /*   skip all records except type 0 2 81  */
337  4                    if (rt:=readcs) > digital$code and rt < digital02 then
338  4                       do;
339  5                       if not t8080 then
340  5                         call get$buffer$len;  else
341  5                         call hex$input;
342  5                       end;    else
343  4                           do;
344  5                           if (rt = digital$code) then
345  5                               do;
346  6                               call hex$input;
347  6                               header(0).typseg = 1;
348  6                               end;   else
349  5                                   do;
350  6                                   if (rt = 0 and sa < segmts(1).begin$add and sa >= segmts(0).begin$add)
                                         or rt = 2 then
351  6                                       do;
352  7                                       la = compute$la(0);
353  7                                       call hex$input;
354  7                                       header(0).typseg = 1;
355  7                                       end;
356  6                                   if (rt = 0 and sa >= segmts(1).begin$add) then
357  6                                        do;
358  7                                        multi$segments = true;
359  7                                        if seg$length(1) <
                                              (high$add:=compute$la(1) +rl-1) then
360  7                                            do;
361  8                                            seg$length(1) = high$add;
362  8                                            header(1).typseg=2;
363  8                                            end;
364  7                                        end;
365  6                                   end;
366  5                           end;
367  4                   end;



368  3       second:
                  call empty$buffers;
369  3                 ta = (la+paragraph-1) and 0fff0h;
370  3                 header(0).file$length=fa/16+1;
371  3                 if header(0).minimum$mem = 0 then header(0).minimum$mem = fa/16+1;
373  3                 fa=ta;
374  3                 if not multi$segments then go to fin;
376  3                 call zero$mem;
377  3                 multi$segments = false;
378  3                 sfcb(ex),sfcb(nr) = 0;
379  3                 call open(.sfcb);
380  3                 call setdma(.buffer);

381  3                 do k = 1 to 7;
382  4                    if seg$length(k) <> 0 then
```

```
383  4                do;
384  5                  seg$length(k) = seg$length(k)+paragraph and 0fff0h;
385  5                  header(k).file$length = seg$length(k)/16;
386  5                  if header(k).minimum$mem=0 then
387  5                  header(k).minimum$mem=seg$length(k)/16;
388  5                  end;
389  4                end;
390  3             segadjst = 0;
391  3             seg$length(0) = ta;
392  3             sbp=length(sbuff);

393  3             DO FOREVER;
                   /* SCAN THE : */
394  4                DO WHILE (nxb:=getchar) <> ':';
395  5                  if nxb = eofile then go to afin;
397  5                END;

398  4             cs = 0;
399  4             rl = readcs;

400  4             la = make$double(readcs,readcs);
401  4             sa = segadjst + shr(la,4);

402  4             if (rt := readcs) = eofile then go to afin;
404  4          if rt = 84h then rt = 83h;
406  4          else if rt = 83h then rt = 84h;
                   if rt > digital$code and rt < digital02 then
409  4             do;
410  5               do k = 0 to (rt-82h);
411  6               la = la + seg$length(k);
412  6               end;
413  5             call hex$input;
414  5             end;
415  4              if (rt = 0 and sa >= segmts(1).begin$add) or rt = 2 then
416  4              do;
417  5                 la = compute$la(1) + seg$length(0);
418  5                 call hex$input;
419  5                 end;

420  4             END;

421  3       afin:
                   call empty$buffers;

422  3          FIN:
                   /* PRINT FINAL STATISTICS */
                   CALL PRINT(.('BYTES READ    $')); CALL PRINTADDR(NB);
424  3          CALL PRINT(.('RECORDS WRITTEN $')); CALL PRINTHEX(P+1);
426  3          CALL CRLF;

                   /*   write the header record   */
427  3          call close(.fcba);
428  3             if head then
```

```
429  3            do;
430  4               fcb(ex),fcb(nr) = 0;
431  4               call open(.fcba);
432  4               call move (.header,.buffer,128);
433  4               call setdma(.buffer);
434  4               if diskwrite(.fcba) <> 0 then call diskerror;

436  4            end;
437  3         END RELOC;


438  2      declare seg$number byte;

439  2      ignore$filename: procedure;
440  3           tbp = 0;
441  3           char = buffer(tbp);
442  3           call next$non$blank;
443  3           do while (char:=buffer(tbp)) <> ' ';
444  4           tbp = tbp +1;
445  4           end;

446  3      end ignore$filename;



447  2      parse$tail: procedure;
448  3         declare seg$index byte;

449  3         get$segmt: procedure byte;
           /*    get the segment name    */
450  4            declare ( kentry, match$flag,j, no$match) byte;
451  4            declare user$segmt(5)  byte;

452  4            do j = 0 to last (user$segmt);
453  5               if delimiter then
454  5               user$segmt(j) = ' ';  else
455  5                 do;
456  6                 user$segmt(j) = char;
457  6                 char = tran(retchar);
458  6                 end;
459  5               end;


460  4            seg$index = 0;
461  4            no$match, matchflag = true;

462  4            do while no$match and seg$index < 11;

463  5                match$flag=true;
464  5                kentry = 0;
465  5                do while match$flag and kentry <= last (segmts.name);
466  6                   if usersegmt(kentry) <> segmts(seg$index).name(kentry) then
467  6                      matchflag = false;  else
468  6                      kentry = kentry +1;
469  6                      end;
470  5                if matchflag then no$match = false; else
472  5                   seg$index = seg$index +1;
```

```
473  5              end;
474  4              if no$match then seg$index = 0ffh;
476  4              return seg$index;
477  4           end get$segmt;

478  3      get$switches: procedure;
479  4         do while char <> ']' and char <> cr;
480  5             call next$non$blank;
481  5           if char= 'A' then header(seg$index).absolute$add = (get$num);
483  5           else if
                 char= 'M' then
484  5               do;
485  6               header(seg$index).minimum$mem = (get$num);
486  6               header(seg$index).typseg = seg$index+1;
487  6               end;
488  5           else if
                 char= 'X' then header(seg$index).maximum$mem = (get$num);
490  5           else if
                 char= 'B' then segmts(seg$index).begin$add = (get$num);
492  5           else do;
493  6              call comline$error;
494  6              call boot;
495  6             end ;
496  5          end;


497  4       end get$switches;



498  3          do forever;
499  4             call next$non$blank;
500  4             if char = cr then return;
502  4             if  get$segmt = 0ffh then
503  4             do;
504  5               call comline$error;
505  5             end;
506  4             if seg$index < 8 then
507  4             do;
508  5                if char = ']' or char = cr then call comline$error;
510  5                call get$switches;
511  5             end;
                   else
512  4             do;
513  5                if seg$index = 8 then t8080 = true; else
515  5                do;
516  6                   if seg$index = 9 then nozero = true; else
518  6                    head = false;
519  6                end;
520  5             end;
521  4          end;

522  3          end parse$tail;


           /* ARRIVE HERE FROM THE SYSTEM MONITOR, READY TO READ THE HEX TAPE */
```

```
                /* SET UP STACKPOINTER IN THE LOCAL AREA */
523   2       DECLARE STACK(64) ADDRESS;
524   2       SP = STACKPOINTER; STACKPOINTER = .STACK(LENGTH(STACK));
526   2       LA = 0h;
527   2       mbuffadr = .memory;
528   2       t8080 = false;
529   2       nozero = false;
530   2       head = true;

531   2       SBP = LENGTH(SBUFF);
                   /* SET UP THE SOURCE FILE */
532   2          CALL MOVE(.FCBA,.SFCB,33);
533   2          CALL MOVE(.('H86',0),.SFCB(9),4);
534   2          CALL OPEN(.SFCB);
535   2          IF DCNT = 255 THEN CALL PERROR(.('CANNOT OPEN SOURCE$'));

537   2          CALL MOVE(.('CMD'),.FCBA+9,3);

                /* REMOVE ANY EXISTING FILE BY THIS NAME */
538   2          CALL DELETE(.FCBA);
                /* THEN OPEN A NEW FILE */
539   2          CALL MAKE(.FCBA); CALL OPEN(.FCBA);
541   2          IF DCNT = 255 THEN CALL PERROR(.('NO MORE DIRECTORY SPACE$')); ELSE
543   2                DO;
544   3                call ignore$filename;
545   3                call parse$tail;
546   3                CALL RELOC;
547   3                CALL CLOSE(.FCBA);
548   3                IF DCNT = 255 THEN CALL PERROR(.('CANNOT CLOSE FILE$'));
550   3                END;
551   2          CALL CRLF;

552   2          CALL BOOT;
553   2       END plmstart;
554   1       END;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
4	0000H	2	A. . . . . WORD PARAMETER 5
48	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 49 50
7	0000H	2	A. . . . . WORD PARAMETER 8
52	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 53 55
43	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 44 45 46
59	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 60 62
71	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 72 73
111	0000H	1	A. . . . . BYTE BASED(S) 113
15	0003H	2	ABSOLUTEADD. . . . WORD MEMBER(HEADER) 482
421	0704H		AFIN . . . . . LABEL 396 403
134	0004H	1	B. . . . . BYTE PARAMETER AUTOMATIC 135 136 138 139 140
275	05B2H	1	B. . . . . BYTE 276 277
38	0004H	1	B. . . . . BYTE PARAMETER AUTOMATIC 39 40 41
204	0004H	1	B. . . . . BYTE PARAMETER AUTOMATIC 205 211
111	0000H	1	B. . . . . BYTE BASED(D) 113
14	0005H	2	BEGINADD . . . . . WORD MEMBER(SEGMTS) 312 350 356 415 491
11	0000H	14	BOOT . . . . . PROCEDURE STACK=0008H 65 127 209 241 494 552
19			BSIZE. . . . . LITERALLY 19
2	0000H	128	BUFFA. . . . . BYTE ARRAY(128) EXTERNAL(2) 18 21
18	0000H	128	BUFFER . . . . . BYTE ARRAY(128) EXTERNAL(2) AT 443 123 131 187 380 432 433 441
24	0004H	1	CHAR . . . . . BYTE PARAMETER AUTOMATIC 25 26
118	05A6H	1	CHAR . . . . . BYTE 143 144 145 164 172 174 176 177 441 443 456 457 479 481 483 488 490 500 508
148	02E5H	36	CHECKONEHEX. . . . PROCEDURE BYTE STACK=0004H 174 264
79	01ABH	19	CLOSE. . . . . PROCEDURE STACK=000AH 427 547
119	025BH	51	COMLINEERROR . . . PROCEDURE STACK=0012H 175 493 504 509
310	09C0H	38	COMPUTELA. . . . . PROCEDURE WORD STACK=0006H 352 359 417
3	0000H	38	COPYRIGHT. . . . . BYTE ARRAY(38) DATA
21	0000H	1	COUNTCOMMANDTAIL . BYTE EXTERNAL(2) AT 130
23			CR . . . . . LITERALLY 29 479 500 508
28	0005H	17	CRLF . . . . . PROCEDURE STACK=000EH 54 121 126 222 240 426 551
200	05ABH	1	CS . . . . . BYTE 276 290 333 398
110	0006H	2	D. . . . . WORD PARAMETER AUTOMATIC 111 113 115
18			DBUFF. . . . . LITERALLY
70	05A5H	1	DCNT . . . . . BYTE 77 81 85 88 104 535 541 548
162	0026H	5	DEL. . . . . BYTE ARRAY(5) DATA 163 164
90	01E3H	16	DELETE . . . . . PROCEDURE STACK=000AH 538
160	0320H	44	DELIMITER. . . . . PROCEDURE BYTE STACK=0002H 173 453
17			DFCBA. . . . . LITERALLY
219	07C7H	90	DIAGNOSE . . . . . PROCEDURE STACK=0024H 268 293
2			DIGITAL02. . . . . LITERALLY 337 408
2			DIGITALCODE. . . . LITERALLY 337 344 408
2			DIGITALDATA. . . . LITERALLY
67	017CH	12	DISKERROR. . . . . PROCEDURE STACK=0026H 246 435
94	01F3H	16	DISKREAD . . . . . PROCEDURE BYTE STACK=000AH 186
98	0203H	16	DISKWRITE. . . . . PROCEDURE BYTE STACK=000AH 245 434
			DOUBLE . . . . . BUILTIN 158
249	085AH	81	EMPTYBUFFERS . . . PROCEDURE STACK=002EH 368 421

```
 19                 EOFILE . . . . . LITERALLY        131  316  330  395  402
  2                 EX . . . . . . . LITERALLY        378  430
  7  0000H     1    F. . . . . . . . BYTE PARAMETER              8
  4  0000H     1    F. . . . . . . . BYTE PARAMETER              5
202  0012H     2    FA . . . . . . . WORD          251  260  287  288  314  370  372  373
 23                 FALSE. . . . . . LITERALLY        167  320  377  467  471  518  528  529
106  0004H     2    FCB. . . . . . . WORD PARAMETER AUTOMATIC                  107  108
 98  0004H     2    FCB. . . . . . . WORD PARAMETER AUTOMATIC                   99  100
 17  0000H    33    FCB. . . . . . . BYTE ARRAY(33) EXTERNAL(1) AT             317  319  430
102  0004H     2    FCB. . . . . . . WORD PARAMETER AUTOMATIC                  103  104
 90  0004H     2    FCB. . . . . . . WORD PARAMETER AUTOMATIC                   91   92
 83  0004H     2    FCB. . . . . . . WORD PARAMETER AUTOMATIC                   84   85
 79  0004H     2    FCB. . . . . . . WORD PARAMETER AUTOMATIC                   80   81
 75  0004H     2    FCB. . . . . . . WORD PARAMETER AUTOMATIC                   76   77
 94  0004H     2    FCB. . . . . . . WORD PARAMETER AUTOMATIC                   95   96
  2  0000H    33    FCBA . . . . . . BYTE ARRAY(33) EXTERNAL(1)                 17  245  427  431  434  532  537
                                      538  539  540  547
 15  0001H     2    FILELENGTH . . . . WORD MEMBER(HEADER)          370  385
422  0707H         FIN. . . . . . . LABEL       375
 23                 FOREVER. . . . . LITERALLY        328  393  498
  1  0000H         GENCMD . . . . . PROCEDURE STACK=0000H
296  094EH   114    GETBUFFERLEN . . . PROCEDURE STACK=0004H                   340
181  0391H   124    GETCHAR. . . . . PROCEDURE BYTE STACK=0026H                264  329  394
169  034CH    69    GETNUM . . . . . PROCEDURE WORD STACK=0016H                482  485  489  491
449  0A6EH   201    GETSEGMT . . . . PROCEDURE BYTE STACK=0008H                502
478  0B37H   175    GETSWITCHES. . . . PROCEDURE STACK=001AH                   510
148  0004H     1    H. . . . . . . . BYTE PARAMETER AUTOMATIC                  149  150  151  152  154
156  0006H     1    H. . . . . . . . BYTE PARAMETER AUTOMATIC                  157  158
 58  05A4H     1    HEAD . . . . . . BYTE        318  428  518  530
 15  00F9H   135    HEADER . . . . . STRUCTURE ARRAY(15) INITIAL               307  347  354  362  370  371  372
                                      385  386  387  432  482  485  486  489
279  08F2H    92    HEXINPUT . . . . . PROCEDURE STACK=003AH                   341  346  353  413  418
                    HIGH . . . . . . BUILTIN           45
203  002AH     2    HIGHADD. . . . . WORD          302  304  305  306  359  361
182  05A9H     1    I. . . . . . . . BYTE        186  189
161  05A8H     1    I. . . . . . . . BYTE        163  164
120  05A7H     1    I. . . . . . . . BYTE        122  123
439  09E6H    45    IGNOREFILENAME . . PROCEDURE STACK=000CH                   544
310  0004H     1    J. . . . . . . . BYTE PARAMETER AUTOMATIC                  311  312
450  05B7H     1    J. . . . . . . . BYTE        452  454  456
200  05ADH     1    K. . . . . . . . BYTE        322  323  381  382  384  385  386  387  410  411
450  05B5H     1    KENTRY . . . . . BYTE        464  465  466  468
263  05B1H     1    KHEX . . . . . . BYTE        264  265
156  0004H     1    L. . . . . . . . BYTE PARAMETER AUTOMATIC                  157  158
 57  0008H     2    LA . . . . . . . WORD           64  206  211  214  215  216  230  233  260  285  287  288
                                      302  312  327  335  336  352  369  400  401  411  417  526
                    LAST . . . . . . BUILTIN         163  183  185  192  452  465
                    LENGTH . . . . . BUILTIN         392  525  531
 23                 LF . . . . . . . LITERALLY        30
                    LOW. . . . . . . BUILTIN          46  234
220  0000H     1    M. . . . . . . . BYTE BASED(TA)
102  0213H    19    MAKE . . . . . . PROCEDURE STACK=000AH                     539
156  0309H    23    MAKEDOUBLE . . . . PROCEDURE WORD STACK=0006H              281  335  400
450  05B6H     1    MATCHFLAG. . . . . BYTE        461  463  465  467  470
  2  0000H     2    MAXB . . . . . . WORD EXTERNAL(0)          206  214
 15  0007H     2    MAXIMUMMEM . . . . WORD MEMBER(HEADER)     489
202  0000H   256    MBUFF. . . . . . BYTE BASED(MBUFFADR) ARRAY(256)           211  215  236
```

```
  57  0006H     2  MBUFFADR . . . . .  WORD        202  211  215  236  527
      0000H        MEMORY . . . . . .  BYTE ARRAY(0)         206  214  250  251  260  527
  15  0005H     2  MINIMUMMEM . . . .  WORD MEMBER(HEADER)         371  372  386  387  485
   4  0000H        MON1 . . . . . . .  PROCEDURE EXTERNAL(3) STACK=0000H    12   26   50   73   92  108
   7  0000H        MON2 . . . . . . .  PROCEDURE BYTE EXTERNAL(4) STACK=0000H         77   81   85   88   96
                                        100  104
 110  0236H    37  MOVE . . . . . . .  PROCEDURE STACK=0008H          187  260  432  532  533  537
 201  05AEH     1  MULTISEGMENTS. . .  BYTE        255  297  320  358  374  377
  32  0004H     1  N. . . . . . . . .  BYTE PARAMETER AUTOMATIC        33   34   35   36
 110  0004H     1  N. . . . . . . . .  BYTE PARAMETER AUTOMATIC       111  112
  14  0000H     5  NAME . . . . . . .  BYTE ARRAY(5) MEMBER(SEGMTS)        465  466
 202  0014H     2  NB . . . . . . . .  WORD        314  334  423
 221  0821H    27  NEWLINE. . . . . .  PROCEDURE STACK=0020H          232  235
 142  02CFH    22  NEXTNONBLANK . . .  PROCEDURE STACK=0008H          442  480  499
 450  0588H     1  NOMATCH. . . . . .  BYTE        461  462  471  474
  22  05A3H     1  NOZERO . . . . . .  BYTE        214  517  529
   2              NR . . . . . . . .  LITERALLY        317  319  378  430
 202  05AFH     1  NXB. . . . . . . .  BYTE        329  330  394  395
  75  0198H    19  OPEN . . . . . . .  PROCEDURE STACK=00DAH          379  431  534  540
 202  0580H     1  P. . . . . . . . .  BYTE        247  315  425
 170  000AH     2  PARADD . . . . . .  WORD        171  176  179
   2              PARAGRAPH. . . . .  LITERALLY        369  384
 447  0A13H    91  PARSETAIL. . . . .  PROCEDURE STACK=001EH          545
  59  0157H    37  PERROR . . . . . .  PROCEDURE STACK=0022H           68  190  536  542  549
  16  000EH   180  PLMSTART . . . . .  PROCEDURE PUBLIC STACK=0042H
  52  0147H    16  PRINT. . . . . . .  PROCEDURE STACK=0014H           61  208  227  229  231  267  292  422
                                        424
  43  0120H    23  PRINTADDR. . . . .  PROCEDURE STACK=001CH           64  223  228  230  423
  24  00C2H    19  PRINTCHAR. . . . .  PROCEDURE STACK=000AH           29   30   35   36  123  125  224  225
                                        238
  38  0105H    27  PRINTHEX . . . . .  PROCEDURE STACK=0016H           45   46  236  425
  48  0137H    16  PRINTM . . . . . .  PROCEDURE STACK=000AH           55   62   63
  32  00E6H    31  PRINTNIB . . . . .  PROCEDURE STACK=0010H           40   41
 271  08CDH    19  READBYTE . . . . .  PROCEDURE BYTE STACK=0030H          276  290
 274  08E0H    18  READCS . . . . . .  PROCEDURE BYTE STACK=0034H          281  284  334  335  337  399  400
                                        402
 262  08ABH    34  READHEX. . . . . .  PROCEDURE BYTE STACK=002AH          272
 199  040DH   863  RELOC. . . . . . .  PROCEDURE STACK=003EH          546
 106  0226H    16  RENAME . . . . . .  PROCEDURE STACK=000AH
 129  028EH    32  RETCHAR. . . . . .  PROCEDURE BYTE STACK=0002H          143  145  172  177  457
  19  05A1H     1  RFLAG. . . . . . .  BYTE
 200  05AAH     1  RL . . . . . . . .  BYTE        283  302  334  359  399
 200  05ACH     1  RT . . . . . . . .  BYTE        280  298  299  300  301  302  306  307  337  344  350  356
                                        402  404  405  406  407  408  410  415
 110  0008H     2  S. . . . . . . . .  WORD PARAMETER AUTOMATIC       111  113  114
 202  0010H     2  SA . . . . . . . .  WORD        312  314  336  350  356  401  415
  19  0002H     2  SBP. . . . . . . .  WORD        183  184  185  187  191  192  195  392  531
  19  01A1H  1024  SBUFF. . . . . . .  BYTE ARRAY(1024)          183  184  185  187  191  192  196  316  392
                                        531
  83  01BEH    19  SEARCH . . . . . .  PROCEDURE STACK=000AH
  87  01D1H    18  SEARCHN. . . . . .  PROCEDURE STACK=0008H
 368  0572H        SECOND . . . . . .  LABEL       331
 202  0016H     2  SEGADJST . . . . .  WORD        281  321  336  390  401
 448  0584H     1  SEGINDEX . . . . .  BYTE        460  462  466  472  475  476  482  485  486  489  491  506
                                        513  516
 202  0018H    16  SEGLENGTH. . . . .  WORD ARRAY(8)         302  306  323  359  361  382  384  385  387  391
                                        411  417
```

```
  14  00ACH    77  SEGMTS . . . . . .  STRUCTURE ARRAY(11) INITIAL          312  350  356  415  465  466  491
 438  05B3H     1  SEGNUMBER. . . . .  BYTE
  71  0188H    16  SETDMA . . . . . .  PROCEDURE STACK=000AH                244  380  433
 204  076CH    42  SETMEM . . . . . .  PROCEDURE STACK=001AH                284
  19  0180H    33  SFCB . . . . . .  BYTE ARRAY(33)          186  378  379  532  533  534
                   SHL. . . . . .  BUILTIN           158  272  312
                   SHR. . . . . .  BUILTIN            40  336  401
  10  0000H     2  SP . . . . . .  WORD        524
 523  002CH   128  STACK. . . . . .  WORD ARRAY(64)          525
 198               STACKPOINTER . . .  LITERALLY         524
                   STACKPTR . . . . .  BUILTIN           524  525
  22  05A2H     1  T8080. . . . . .  BYTE        339  514  528
 202  000EH     2  TA . . . . . .  WORD        220  223  228  233  234  236  237  326  335  369  373  391
 202  000CH     2  TABS . . . . . .  WORD
  20  0004H     2  TBP. . . . . .  WORD        122  130  131  440  441  443  444
 134  02AEH    33  TRAN . . . . . .  PROCEDURE BYTE STACK=0004H            143  145  457
  23               TRUE . . . . . .  LITERALLY         165  297  328  358  393  461  463  498  514  517  530
  15  0000H     1  TYPSEG . . . . . .  BYTE MEMBER(HEADER)          307  347  354  362  486
 451  05B9H     5  USERSEGMT. . . . .  BYTE ARRAY(5)          452  454  456  466
  23               WHAT . . . . . .  LITERALLY
 202  0028H     2  WRITEADD . . . . .  WORD        244  250  251  253  260
 243  083CH    30  WRITERECORD. . . .  PROCEDURE STACK=002AH                252  257
 213  0796H    49  ZEROMEM. . . . . .  PROCEDURE STACK=0002H                325  376
```


MODULE INFORMATION:

```
  CODE AREA SIZE     = 0BE6H   3046D
  CONSTANT AREA SIZE = 0131H    305D
  VARIABLE AREA SIZE = 058EH   1470D
  MAXIMUM STACK SIZE = 0042H     66D
  771 LINES READ
  0 PROGRAM ERROR(S)
```

END OF PL/M-86 COMPILATION

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

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

BASE   OFFSET TYPE SYMBOL              BASE   OFFSET TYPE SYMBOL

0000H  010EH  PUB  PLMSTART            0000H  005DH  PUB  XDOS
0000H  005DH  PUB  MON1                0000H  005DH  PUB  MON2
0000H  005DH  PUB  MON3                0000H  005DH  PUB  MON4
1000H  0004H  PUB  BDISK               1000H  0006H  PUB  MAXB
1000H  0050H  PUB  CMDRV               1000H  0051H  PUB  PASS0
1000H  0053H  PUB  LEN0                1000H  0054H  PUB  PASS1
1000H  0056H  PUB  LEN1                1000H  005CH  PUB  FCB
1000H  006CH  PUB  FCB16               1000H  007CH  PUB  CR
1000H  007DH  PUB  RR                  1000H  007FH  PUB  R0
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

GENCMD:  SYMBOLS AND LINES
1000H  0860H  SYM  MEMORY              1000H  0728H  SYM  COPYRIGHT
1000H  0108H  SYM  SP                  0000H  0100H  SYM  BOOT
1000H  01B4H  SYM  SEGMTS              1000H  0201H  SYM  HEADER
0000H  010EH  SYM  PLMSTART            1000H  005CH  SYM  FCB
1000H  0080H  SYM  BUFFER              1000H  0288H  SYM  SFCB
1000H  02A9H  SYM  SBUFF               1000H  06A9H  SYM  RFLAG
1000H  010AH  SYM  SBP                 1000H  010CH  SYM  TBP
1000H  0080H  SYM  COUNTCOMMANDTAIL    1000H  06AAH  SYM  T8080
1000H  06ABH  SYM  NOZERO              0000H  01C2H  SYM  PRINTCHAR
STACK  0004H  SYM  CHAR                0000H  01D5H  SYM  CRLF
0000H  01E6H  SYM  PRINTNIB            STACK  0004H  SYM  N
0000H  0205H  SYM  PRINTHEX            STACK  0004H  SYM  B
0000H  0220H  SYM  PRINTADDR           STACK  0004H  SYM  A
0000H  0237H  SYM  PRINTM              STACK  0004H  SYM  A
0000H  0247H  SYM  PRINT               STACK  0004H  SYM  A
1000H  010EH  SYM  MBUFFADR            1000H  0110H  SYM  LA
1000H  06ACH  SYM  HEAD                0000H  0257H  SYM  PERROR
STACK  0004H  SYM  A                   0000H  027CH  SYM  DISKERROR
1000H  06ADH  SYM  DCNT                0000H  0288H  SYM  SETDMA
STACK  0004H  SYM  A                   0000H  0298H  SYM  OPEN
STACK  0004H  SYM  FCB                 0000H  02ABH  SYM  CLOSE
STACK  0004H  SYM  FCB                 0000H  02BEH  SYM  SEARCH
STACK  0004H  SYM  FCB                 0000H  02D1H  SYM  SEARCHN
0000H  02E3H  SYM  DELETE              STACK  0004H  SYM  FCB
0000H  02F3H  SYM  DISKREAD            STACK  0004H  SYM  FCB
0000H  0303H  SYM  DISKWRITE           STACK  0004H  SYM  FCB
0000H  0313H  SYM  MAKE                STACK  0004H  SYM  FCB
0000H  0326H  SYM  RENAME              STACK  0004H  SYM  FCB
0000H  0336H  SYM  MOVE                STACK  0008H  SYM  S
STACK  0006H  SYM  D                   STACK  0004H  SYM  N
STACK  0008H  BAS  A                   STACK  0006H  BAS  B
1000H  06AEH  SYM  CHAR                0000H  0358H  SYM  COMLINEERROR
1000H  06AFH  SYM  I                   0000H  038EH  SYM  RETCHAR
0000H  03AEH  SYM  TRAN                STACK  0004H  SYM  B
0000H  03CFH  SYM  NEXTNONBLANK        0000H  03E5H  SYM  CHECKONEHEX
STACK  0004H  SYM  H                   0000H  0409H  SYM  MAKEDOUBLE
```

```
STACK  0006H  SYM  H
0000H  0420H  SYM  DELIMITER
1000H  074EH  SYM  DEL
1000H  0112H  SYM  PARADD
1000H  0681H  SYM  I
1000H  0682H  SYM  RL
1000H  0684H  SYM  RT
1000H  0686H  SYM  MULTISEGMENTS
1000H  0116H  SYM  TA
1000H  011AH  SYM  FA
1000H  0687H  SYM  NXB
1000H  0120H  SYM  SEGLENGTH
1000H  010EH  BAS  MBUFF
1000H  0132H  SYM  HIGHADD
STACK  0004H  SYM  B
0000H  08C7H  SYM  DIAGNOSE
0000H  0921H  SYM  NEWLINE
0000H  095AH  SYM  EMPTYBUFFERS
1000H  0689H  SYM  KHEX
0000H  09E0H  SYM  READCS
0000H  09F2H  SYM  HEXINPUT
0000H  0AC0H  SYM  COMPUTELA
0000H  0672H  SYM  SECOND
0000H  0807H  SYM  FIN
0000H  0AE6H  SYM  IGNOREFILENAME
1000H  068CH  SYM  SEGINDEX
1000H  068DH  SYM  KENTRY
1000H  068FH  SYM  J
1000H  06C1H  SYM  USERSEGMT
1000H  0134H  SYM  STACK
0000H  0103H  LIN     12
0000H  010EH  LIN     16
0000H  01C5H  LIN     26
0000H  01D5H  LIN     28
0000H  01DEH  LIN     30
0000H  01E6H  LIN     32
0000H  01EFH  LIN     35
0000H  0201H  LIN     37
0000H  0208H  LIN     40
0000H  021CH  LIN     42
0000H  0223H  LIN     45
0000H  0233H  LIN     47
0000H  023AH  LIN     50
0000H  0247H  LIN     52
0000H  024DH  LIN     55
0000H  0257H  LIN     59
0000H  0261H  LIN     62
0000H  026EH  LIN     64
0000H  0278H  LIN     66
0000H  027FH  LIN     68
0000H  0288H  LIN     71
0000H  0294H  LIN     74
0000H  029BH  LIN     77
0000H  02ABH  LIN     79
0000H  02BAH  LIN     82
0000H  02C1H  LIN     85
0000H  02D1H  LIN     87
0000H  02E1H  LIN     89
0000H  02E6H  LIN     92
0000H  02F3H  LIN     94

STACK  0004H  SYM  L
1000H  0630H  SYM  I
0000H  044CH  SYM  GETNUM
0000H  0491H  SYM  GETCHAR
0000H  050DH  SYM  RELOC
1000H  0683H  SYM  CS
1000H  0685H  SYM  K
1000H  0114H  SYM  TABS
1000H  0118H  SYM  SA
1000H  011CH  SYM  NB
1000H  011EH  SYM  SEGADJST
1000H  0130H  SYM  WRITEADD
1000H  0688H  SYM  P
0000H  086CH  SYM  SETMEM
0000H  0896H  SYM  ZEROMEM
1000H  0116H  BAS  M
0000H  093CH  SYM  WRITERECORD
0000H  09ABH  SYM  READHEX
0000H  09CDH  SYM  READBYTE
1000H  068AH  SYM  B
0000H  0A4EH  SYM  GETBUFFERLEN
STACK  0004H  SYM  J
0000H  0804H  SYM  AFIN
1000H  068BH  SYM  SEGNUMBER
0000H  0813H  SYM  PARSETAIL
0000H  0B6EH  SYM  GETSEGMT
1000H  068EH  SYM  MATCHFLAG
1000H  06C0H  SYM  NOMATCH
0000H  0C37H  SYM  GETSWITCHES
0000H  0100H  LIN     11
0000H  010CH  LIN     13
0000H  01C2H  LIN     24
0000H  01D1H  LIN     27
0000H  01D8H  LIN     29
0000H  01E4H  LIN     31
0000H  01E9H  LIN     34
0000H  01F8H  LIN     36
0000H  0205H  LIN     38
0000H  0213H  LIN     41
0000H  0220H  LIN     43
0000H  022CH  LIN     46
0000H  0237H  LIN     48
0000H  0243H  LIN     51
0000H  024AH  LIN     54
0000H  0253H  LIN     56
0000H  025AH  LIN     61
0000H  0267H  LIN     63
0000H  0275H  LIN     65
0000H  027CH  LIN     67
0000H  0286H  LIN     69
0000H  028BH  LIN     73
0000H  0298H  LIN     75
0000H  02A7H  LIN     78
0000H  02AEH  LIN     81
0000H  02BEH  LIN     83
0000H  02C0H  LIN     86
0000H  02D4H  LIN     88
0000H  02E3H  LIN     90
0000H  02EFH  LIN     93
0000H  02F6H  LIN     96
```

```
0000H  0303H  LIN    97      0000H  0303H  LIN    98
0000H  0306H  LIN   100      0000H  0313H  LIN   101
0000H  0313H  LIN   102      0000H  0316H  LIN   104
0000H  0322H  LIN   105      0000H  0326H  LIN   106
0000H  0329H  LIN   108      0000H  0332H  LIN   109
0000H  0336H  LIN   110      0000H  0339H  LIN   112
0000H  0345H  LIN   113      0000H  034FH  LIN   114
0000H  0352H  LIN   115      0000H  0355H  LIN   116
0000H  0357H  LIN   117      0000H  0358H  LIN   119
0000H  035EH  LIN   121      0000H  0361H  LIN   122
0000H  0371H  LIN   123      0000H  037AH  LIN   124
0000H  0380H  LIN   125      0000H  0386H  LIN   126
0000H  0389H  LIN   127      0000H  038CH  LIN   128
0000H  038EH  LIN   129      0000H  0391H  LIN   130
0000H  03A2H  LIN   131      0000H  03AAH  LIN   132
0000H  03AEH  LIN   134      0000H  03B1H  LIN   136
0000H  03B7H  LIN   137      0000H  03BBH  LIN   138
0000H  03C4H  LIN   139      0000H  03C8H  LIN   140
0000H  03CFH  LIN   141      0000H  03CFH  LIN   142
0000H  03D2H  LIN   143      0000H  03DCH  LIN   144
0000H  03E3H  LIN   147      0000H  03E5H  LIN   148
0000H  03E8H  LIN   150      0000H  03F1H  LIN   152
0000H  03FAH  LIN   153      0000H  03FEH  LIN   154
0000H  0409H  LIN   155      0000H  0409H  LIN   156
0000H  040CH  LIN   158      0000H  0420H  LIN   159
0000H  0420H  LIN   160      0000H  0423H  LIN   163
0000H  042FH  LIN   164      0000H  043EH  LIN   165
0000H  0442H  LIN   166      0000H  0448H  LIN   167
0000H  044CH  LIN   168      0000H  044CH  LIN   169
0000H  044FH  LIN   171      0000H  0455H  LIN   172
0000H  045BH  LIN   173      0000H  0464H  LIN   174
0000H  0472H  LIN   175      0000H  0477H  LIN   176
0000H  048AH  LIN   177      0000H  048CH  LIN   178
0000H  048CH  LIN   179      0000H  0491H  LIN   180
0000H  0491H  LIN   181      0000H  0496H  LIN   183
0000H  04A0H  LIN   184      0000H  04A8H  LIN   185
0000H  04C0H  LIN   186      0000H  04CEH  LIN   187
0000H  04E3H  LIN   189      0000H  04EAH  LIN   190
0000H  04F1H  LIN   191      0000H  04FAH  LIN   192
0000H  0500H  LIN   194      0000H  050DH  LIN   195
0000H  0508H  LIN   196      0000H  050DH  LIN   197
0000H  050DH  LIN   199      0000H  086CH  LIN   204
0000H  086FH  LIN   206      0000H  0878H  LIN   208
0000H  0882H  LIN   209      0000H  0885H  LIN   211
0000H  0892H  LIN   212      0000H  0896H  LIN   213
0000H  0899H  LIN   214      0000H  08B4H  LIN   215
0000H  08BFH  LIN   216      0000H  08C3H  LIN   217
0000H  08C5H  LIN   218      0000H  08C7H  LIN   219
0000H  0921H  LIN   221      0000H  0924H  LIN   222
0000H  0927H  LIN   223      0000H  092EH  LIN   224
0000H  0934H  LIN   225      0000H  093AH  LIN   226
0000H  08CAH  LIN   227      0000H  08D1H  LIN   228
0000H  08D8H  LIN   229      0000H  08DFH  LIN   230
0000H  08E6H  LIN   231      0000H  08EDH  LIN   232
0000H  08F0H  LIN   233      0000H  08F9H  LIN   234
0000H  08FDH  LIN   235      0000H  0900H  LIN   236
0000H  090DH  LIN   237      0000H  0911H  LIN   238
0000H  0917H  LIN   239      0000H  0919H  LIN   240
0000H  091CH  LIN   241      0000H  091FH  LIN   242
0000H  093CH  LIN   243      0000H  093FH  LIN   244
```

```
0000H  0946H  LIN   245
0000H  0951H  LIN   246
0000H  0954H  LIN   247
0000H  0958H  LIN   248
0000H  095AH  LIN   249
0000H  095DH  LIN   250
0000H  0963H  LIN   251
0000H  0975H  LIN   252
0000H  0978H  LIN   253
0000H  097EH  LIN   254
0000H  0980H  LIN   255
0000H  0989H  LIN   257
0000H  098CH  LIN   258
0000H  098EH  LIN   260
0000H  09A9H  LIN   261
0000H  09ABH  LIN   262
0000H  09AEH  LIN   264
0000H  09BCH  LIN   265
0000H  09C1H  LIN   267
0000H  09C8H  LIN   268
0000H  09C8H  LIN   270
0000H  09CDH  LIN   271
0000H  09D0H  LIN   272
0000H  09E0H  LIN   273
0000H  09E0H  LIN   274
0000H  09E3H  LIN   276
0000H  09EDH  LIN   277
0000H  09F2H  LIN   278
0000H  09F2H  LIN   279
0000H  09F5H  LIN   280
0000H  0A03H  LIN   281
0000H  0A13H  LIN   283
0000H  0A1FH  LIN   284
0000H  0A26H  LIN   285
0000H  0A2AH  LIN   286
0000H  0A2CH  LIN   287
0000H  0A35H  LIN   288
0000H  0A39H  LIN   290
0000H  0A42H  LIN   292
0000H  0A49H  LIN   293
0000H  0A4CH  LIN   295
0000H  0A4EH  LIN   296
0000H  0A51H  LIN   297
0000H  0A56H  LIN   298
0000H  0A5DH  LIN   299
0000H  0A64H  LIN   300
0000H  0A6BH  LIN   301
0000H  0A70H  LIN   302
0000H  0A8EH  LIN   304
0000H  0A92H  LIN   305
0000H  0A98H  LIN   306
0000H  0AA8H  LIN   307
0000H  0ABEH  LIN   309
0000H  0AC0H  LIN   310
0000H  0AC3H  LIN   312
0000H  0AE6H  LIN   313
0000H  0510H  LIN   314
0000H  051CH  LIN   315
0000H  051FH  LIN   316
0000H  0524H  LIN   317
0000H  0527H  LIN   318
0000H  052EH  LIN   319
0000H  0533H  LIN   320
0000H  0538H  LIN   321
0000H  053DH  LIN   322
0000H  0547H  LIN   323
0000H  0555H  LIN   324
0000H  0558H  LIN   325
0000H  055EH  LIN   326
0000H  0564H  LIN   327
0000H  056AH  LIN   328
0000H  056AH  LIN   329
0000H  0574H  LIN   330
0000H  057EH  LIN   331
0000H  057EH  LIN   332
0000H  0580H  LIN   333
0000H  0585H  LIN   334
0000H  0591H  LIN   335
0000H  05A2H  LIN   336
0000H  05B0H  LIN   337
0000H  05CFH  LIN   339
0000H  05D8H  LIN   340
0000H  05DDH  LIN   341
0000H  05E0H  LIN   342
0000H  05E2H  LIN   344
0000H  05E9H  LIN   346
0000H  05ECH  LIN   347
0000H  05F1H  LIN   348
0000H  05F3H  LIN   350
0000H  0626H  LIN   352
0000H  062FH  LIN   353
0000H  0632H  LIN   354
0000H  0637H  LIN   356
0000H  0647H  LIN   358
0000H  064CH  LIN   359
0000H  0664H  LIN   361
0000H  066AH  LIN   362
0000H  066FH  LIN   367
0000H  0672H  LIN   368
0000H  0675H  LIN   369
0000H  0684H  LIN   370
0000H  068FH  LIN   371
0000H  0696H  LIN   372
0000H  06A1H  LIN   373
0000H  06A7H  LIN   374
0000H  06B3H  LIN   375
0000H  06B3H  LIN   376
0000H  06B6H  LIN   377
0000H  06BBH  LIN   378
0000H  06C1H  LIN   379
0000H  06C8H  LIN   380
0000H  06CFH  LIN   381
0000H  06DBH  LIN   382
0000H  06E8H  LIN   384
0000H  06F9H  LIN   385
0000H  0710H  LIN   386
0000H  0717H  LIN   387
0000H  071BH  LIN   389
0000H  0721H  LIN   390
0000H  0727H  LIN   391
```

```
0000H  072DH  LIN    392
0000H  0733H  LIN    394
0000H  0744H  LIN    396
0000H  0746H  LIN    398
0000H  0751H  LIN    400
0000H  076DH  LIN    402
0000H  077AH  LIN    404
0000H  0788H  LIN    406
0000H  0794H  LIN    408
0000H  07B2H  LIN    411
0000H  07C8H  LIN    413
0000H  07F1H  LIN    417
0000H  0801H  LIN    420
0000H  0807H  LIN    422
0000H  0815H  LIN    424
0000H  0825H  LIN    426
0000H  082FH  LIN    428
0000H  0840H  LIN    431
0000H  0855H  LIN    433
0000H  0867H  LIN    435
0000H  0AE6H  LIN    439
0000H  0AEFH  LIN    441
0000H  0AFCH  LIN    443
0000H  0B0FH  LIN    445
0000H  0B13H  LIN    447
0000H  0B71H  LIN    452
0000H  0B84H  LIN    454
0000H  0B9EH  LIN    457
0000H  0BAEH  LIN    460
0000H  0BBDH  LIN    462
0000H  0BD4H  LIN    464
0000H  0BEBH  LIN    466
0000H  0C0CH  LIN    468
0000H  0C12H  LIN    470
0000H  0C20H  LIN    472
0000H  0C26H  LIN    474
0000H  0C32H  LIN    476
0000H  0C37H  LIN    478
0000H  0C59H  LIN    480
0000H  0C63H  LIN    482
0000H  0C7EH  LIN    485
0000H  0CA4H  LIN    487
0000H  0CADH  LIN    489
0000H  0CC8H  LIN    491
0000H  0CDFH  LIN    494
0000H  0CE4H  LIN    497
0000H  0B16H  LIN    499
0000H  0B20H  LIN    501
0000H  0B29H  LIN    504
0000H  0B33H  LIN    508
0000H  0B44H  LIN    510
0000H  0B49H  LIN    513
0000H  0B57H  LIN    516
0000H  0B65H  LIN    518
0000H  0B6CH  LIN    522
0000H  0116H  LIN    525
0000H  0125H  LIN    527
0000H  012EH  LIN    529
0000H  0136H  LIN    531
0000H  014AH  LIN    533

0000H  0733H  LIN    393
0000H  073DH  LIN    395
0000H  0744H  LIN    397
0000H  0748H  LIN    399
0000H  075FH  LIN    401
0000H  077AH  LIN    403
0000H  0781H  LIN    405
0000H  078FH  LIN    407
0000H  07A2H  LIN    410
0000H  07C2H  LIN    412
0000H  07CBH  LIN    415
0000H  07FEH  LIN    418
0000H  0804H  LIN    421
0000H  080EH  LIN    423
0000H  081CH  LIN    425
0000H  0828H  LIN    427
0000H  0836H  LIN    430
0000H  0847H  LIN    432
0000H  085CH  LIN    434
0000H  086AH  LIN    437
0000H  0AE9H  LIN    440
0000H  0AF9H  LIN    442
0000H  0B08H  LIN    444
0000H  0B11H  LIN    446
0000H  086EH  LIN    449
0000H  087DH  LIN    453
0000H  0B91H  LIN    456
0000H  08A8H  LIN    459
0000H  0BB3H  LIN    461
0000H  0BCFH  LIN    463
0000H  0BD9H  LIN    465
0000H  0C05H  LIN    467
0000H  0C10H  LIN    469
0000H  0C19H  LIN    471
0000H  0C24H  LIN    473
0000H  0C2DH  LIN    475
0000H  0C37H  LIN    477
0000H  0C3AH  LIN    479
0000H  0C5CH  LIN    481
0000H  0C77H  LIN    483
0000H  0C90H  LIN    486
0000H  0CA6H  LIN    488
0000H  0CC1H  LIN    490
0000H  0CDCH  LIN    493
0000H  0CE2H  LIN    496
0000H  0B16H  LIN    498
0000H  0B19H  LIN    500
0000H  0B22H  LIN    502
0000H  0B2CH  LIN    506
0000H  0B41H  LIN    509
0000H  0B47H  LIN    511
0000H  0B50H  LIN    514
0000H  0B5EH  LIN    517
0000H  086AH  LIN    521
0000H  0111H  LIN    524
0000H  011FH  LIN    526
0000H  012BH  LIN    528
0000H  0131H  LIN    530
0000H  013CH  LIN    532
0000H  0158H  LIN    534
```

```
0000H  015FH  LIN     535          0000H  0166H  LIN     536
0000H  016DH  LIN     537          0000H  017BH  LIN     538
0000H  0182H  LIN     539          0000H  0189H  LIN     540
0000H  0190H  LIN     541          0000H  0197H  LIN     542
0000H  019CH  LIN     544          0000H  019FH  LIN     545
0000H  01A2H  LIN     546          0000H  01A5H  LIN     547
0000H  01ACH  LIN     548          0000H  01B3H  LIN     549
0000H  01BAH  LIN     551          0000H  01BDH  LIN     552
0000H  01C0H  LIN     553          0000H  0100H  LIN     554

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

SEGMENT MAP

START     STOP      LENGTH ALIGN NAME            CLASS

00000H    00CE5H    0CE6H   G    CODE            CODE
10000H    10107H    0108H   G    DATS            DATA
10108H    106C5H    05BEH   W    DATA            DATA
106C6H    10727H    0062H   W    STACK           STACK
1072BH    10858H    0131H   W    CONST           CONST
10860H    10860H    0000H   G    ??SEG
10860H    10860H    0000H   W    MEMORY          MEMORY


GROUP MAP

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