CP/M 2 SYSTEM SOURCE LISTINGS

Disclaimer

Digital Research makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, Digital Research reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Digital Research to notify any person of such revision or changes.

Notice

The source listings included in this doucment are **PROPRIETARY**, and must be protected by the customer from copying and abstraction, as specified in the **DIGITAL RESEARCH SOFTWARE LICENSE AGREEMENT.** All Digital Research software source programs, machine code programs, and manuals are protected by copyright. That means it is illegal to make any copy of all or part of a Digital Research program, whether of source code or object code, to translate a Digital Research program (for example by using an assembler or compiler), to load or execute a Digital Research program, or to derive your own version of a Digital Research program, without the written permission of Digital Research.

Digital Research customers who wish to do these things can generally obtain written permission to do so: Digital Research licenses its proprietary software with several standard customer agreements. These agreements grant permission to exercise the copyrights that are required by the intended use of the product.

If your LICENSE AGREEMENT does not seem to grant you a permission you require, please contact Digital Research, Box 579, Pacific Grove, California, 93950.

DIGITAL RESEARCH
Box 579 Pacific Grove, California, 93950
SOFTWARE LICENSE AGREEMENT

IMPORTANT: All Digital Research programs are sold only on the condition that the purchaser agrees to the following license. READ THIS LICENSE CAREFULLY. If you do not agree to the terms contained in this license, return the packaged diskette UNOPENED to your distributor and your purchase price will be refunded. If you agree to the terms contained in this license, fill out the REGISTRATION information and RETURN by mail.

DIGITAL RESEARCH agrees to grant and the Customer agrees to accept on the following terms and conditions nontransferable and nonexclusive licenses to use the software program(s) (Licensed Programs) herein delivered with this agreement.

TERM:

This agreement is effective from the date of receipt of the above-referenced program(s) and shall remain in force until terminated by the Customer upon one month's prior written notice, or by Digital Research as provided below.

Any license under this Agreement may be discontinued by the Customer at any time upon one month's prior written notice. Digital Research may discontinue any license or terminate this Agreement if the Customer fails to comply with any of the terms and conditions of this Agreement.

LICENSE:

Each program license granted under this Agreement authorizes the Customer to use the Licensed Program in any machine readable form on any single computer system (referred to as System). A separate license is required for each System on which the Licensed Program will be used.

This Agreement and any of the licenses, programs or materials to which it applies may not be assigned, sublicensed or otherwise transferred by the Customer without prior written consent from Digital Research. No right to print or copy, in whole or in part, the Licensed Programs is granted except as hereinafter expressly provided.

PERMISSION TO COPY OR MODIFY LICENSED PROGRAMS:

The customer shall not copy, in whole or in part, any Licensed Programs which are provided by Digital Research in printed form under this Agreement. Additional copies of printed materials may be acquired from Digital Research.

Any Licensed Programs which are provided by Digital Research in machine readable form may be copied, in whole or in part, in printed or machine readable form in sufficient number for use by the Customer with the designated System, to understand the contents of such machine readable material, to modify the Licensed Program as provided below, for back-up purposes, or for archive purposes, provided, however, that no more than five (5) printed copies will be in existence under any license at any one time without prior written consent from Digital Research. The Customer agrees to maintain appropriate records of the number and location of all such copies of Licensed Programs. The original, and any copies of the Licensed Programs, in whole or in part, which are made by the Customer shall be the property of Digital Research. This does not imply, of course, that Digital Research owns the media on which the Licensed Programs are recorded. The Customer may modify any machine readable form of the Licensed Programs for his own use and merge it into other program material to form an updated work, provided that, upon discontinuance of the license for such Licensed Program, the Licensed Program supplied by Digital Research will be completely removed from the updated work. Any portion of the Licensed Program included in an updated work shall be used only if on the designated System and shall remain subject to all other terms of

this Agreement.

The Customer agrees to reproduce and include the copyright notice of Digital Research on all copies, in whole or in part, in any form, including partial copies of modifications, of Licensed Programs made hereunder.

PROTECTION AND SECURITY:

The customer agrees not to provide or otherwise make available any Licensed Program including but not limited to program listings, object code and source code, in any form, to any person other than Customer or Digital Research employees, without prior written consent from Digital Research, except with the Customer's permission for purposes specifically related to the Customer's use of the Licensed Program.

DISCONTINUANCE:

Within one month after the date of discontinuance of any license under this Agreement, the Customer will furnish Digital Research a certificate certifying the through his best effort, and to the best of his knowledge, the original and all copies, in whole or in part, in any form, including partial copies in modifications, of the Licensed Program received from Digital Research or made in connection with such license have been destroyed, except that, upon prior written authorization from Digital Research, the Customer may retain a copy for archive purposes.

DISCLAIMER OF WARRANTY:

Digital Research makes no warranties with respect to the Licensed Programs. The sole obligation of Digital Research shall be to make available all published modifications or updates made by Digital Research to Licensed Programs which are published within one (1) year from date of purchase, provided Customer has returned the Registration Card delivered with the Licensed Program.

LIMITATION OF LIABILITY:

THE FOREGOING WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL DIGITAL RESEARCH BE LIABLE FOR CONSEQUENTIAL DAMAGES EVEN IF DIGITAL RESEARCH HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

GENERAL

If any of the provisions, or portions thereof, of this Agreement are invalid under any applicable statute or rule of law, they are to that extent to be deemed omitted.

TABLE OF CONTENTS

MDS Cold Start Loader	1
Console Command Processor	3
Basic Disk Operating System	11
MDS Basic I/O System	32
Transient Interface Module	38
XSUB Relocator	38
XSUB Module	4Ø
Peripheral Interchange Program	42
SUBMIT Processor	58
System Status Program	61
Context Editor Program	7Ø
Hex File Loader Program	87
CP/M 2.Ø Relocator Module	91
System Generation Program	95
ASM Common Data Module	1ØØ
ASM I/O Module	1ØØ
ASM Scanner Module	1Ø7
ASM Symbol Table Module	111
ASM Table Search Module	115
ASM Operand Scan Module	119
ASM Main Module	125
DDT Relocator Module	134
DDT Assembler/Disassembler	135
DDT Main Module	146
ED Memory Move Patch	162
ASM String Handler Patch	163
DDT Version Number Patch	163

```
CP/M MACRO ASSEM 2.0     #001    mds cold start loader at 3000h

                         TITLE   'mds cold start loader at 3000h'
                ;
                ;        MDS E00 COLD START LOADER FOR CP M 2.0
                ;
                ;        VERSION 2.0 AUGUST  1979
                ;
0000 =          FALSE   EQU     0
FFFF =          TRUE    EQU     NOT FALSE
0000 =          TESTING EQU     FALSE   ;IF TRUE  THEN GO TO MON80 ON ERRORS
                ;
                        IF      TESTING
                BIAS    EQU     03400H
                        ENDIF
                        IF      NOT TESTING
0000 =          BIAS    EQU     0000H
                        ENDIF
0000            CPM     EQU     BIAS            ;BASE OF DOS LOAD
0E06 =          BDOS    EQU     806H+BIAS       ;ENTRY TO DOS FOR CALLS
1880 =          BDOSE   EQU     1880H+BIAS      ;END OF DOS LOAD
1600 =          BOOT    EQU     1600H+BIAS      ;COLD START ENTRY POINT
1603 =          RBOOT   EQU     BOOT+3          ;WARM START ENTRY POINT
                ;
3000                    ORG     03000H  ;LOADED DOWN FROM HARDWARE BOOT AT 3000H
                ;
1880            BDOSL   EQU     BDOSE-CPM
0002 =          NTRKS   EQU     2       ;NUMBER OF TRACKS TO READ
0031            BDOSS   EQU     BDOSL*128       ;NUMBER OF SECTORS IN DOS
0019 =          BDOS0   EQU     25      ;NUMBER OF BDOS SECTORS ON TRACK 0
0018 =          BDOS1   EQU     BDOSS-BDOS0     ;NUMBER OF SECTORS ON TRACK 1
                ;
F800 =          MON80   EQU     0F800H  ;INTEL MONITOR BASE
FF0F =          RMON80  EQU     0FF0FH  ;RESTART LOCATION FOR MON80
0078 =          BASE    EQU     078H    ;'BASE' USED BY CONTROLLER
0079 =          RTYPE   EQU     BASE+1  ;RESULT TYPE
007B =          RBYTE   EQU     BASE+3  ;RESULT BYTE
007F =          RESET   EQU     BASE+7  ;RESET CONTROLLER
                ;
0078 =          DSTAT   EQU     BASE    ;DISK STATUS PORT
0079            ILOW    EQU     BASE+1  ;LOW IOPB ADDRESS
007A =          IHIGH   EQU     BASE+2  ;HIGH IOPB ADDRESS
00FF =          BSW     EQU     0FFH    ;BOOT SWITCH
0003 =          RECAL   EQU     3H      ;RECALIBRATE SELECTED DRIVE
0004 =          READF   EQU     4H      ;DISK READ FUNCTION
0100 =          STACK   EQU     100H    ;USE END OF BOOT FOR STACK
                ;
                RSTART:
3000 310001             LXI     SP,STACK;IN CASE OF CALL TO MON80
                ;       CLEAR DISK STATUS
3003 DB79               IN      RTYPE
3005 DB7B               IN      RBYTE
                ;       CHECK IF BOOT SWITCH IS OFF
                COLDSTART:
3007 DBFF               IN      BSW
3009 E602               ANI     02H     ;SWITCH ON?
300B C20730             JNZ     COLDSTART
                ;       CLEAR THE CONTROLLER
```

```
CP/M MACRO ASSEM 2.0     #002    mds cold start loader at 3000h

300E D37F               OUT     RESET   ;LOGIC CLEARED
                ;
                ;
3010 0602               MVI     B,NTRKS ;NUMBER OF TRACKS TO READ
3012 214230             LXI     H,IOPB0
                ;
                START:
                ;
                ;       READ FIRST NEXT TRACK INTO CPMB
3015 7D                 MOV     A,L
3016 D379               OUT     ILOW
3018 7C                 MOV     A,H
3019 D37A               OUT     IHIGH
301B DB78       WAIT0:  IN      DSTAT
301D E604               ANI     4
301F CA1B30             JZ      WAIT0
                ;
                ;       CHECK DISK STATUS
3022 DB79               IN      RTYPE
3024 E603               ANI     11B
3026 FE02               CPI     2
                ;
                        IF      TESTING
                        CNC     RMON80  ;GO TO MONITOR IF 11 OR 10
                        ENDIF
                        IF      NOT TESTING
3028 D20030             JNC     RSTART  ;RETRY THE LOAD
                        ENDIF
                ;
302B DB7B               IN      RBYTE   ;I/O COMPLETE. CHECK STATUS
                ;       IF NOT READY, THEN GO TO MON80
302D 17                 RAL
302E DC0FFF             CC      RMON80  ;NOT READY BIT SET
3031 1F                 RAR             ;RESTORE
3032 E61F               ANI     11110B  ;OVERRUN/ADDR ERR/SEEK/CRC/XXXX
                ;
                        IF      TESTING
                        CNZ     RMON80  ;GO TO MONITOR
                        ENDIF
                        IF      NOT TESTING
3034 C20030             JNZ     RSTART  ;RETRY THE LOAD
                        ENDIF
                ;
                ;
3037 110700             LXI     D IOPBL ;LENGTH OF IOPB
303A 19                 DAD     D       ;ADDRESSING NEXT IOPB
303B 05                 DCR     B       ;COUNT DOWN TRACKS
303C C21530             JNZ     START
                ;
                ;
                ;       JMP TO BOOT TO PRINT INITIAL MESSAGE, AND SET UP JMPS
303F C30016             JMP     BOOT
                ;
                ;       PARAMETER BLOCKS
3042 80         IOPB0:  DB      80H     ;IOCW  NO UPDATE
3043 04                 DB      READF   ;READ FUNCTION
```

```
CP/M MACRO ASSEM 2.0     #003     mds cold start loader at 3000h

 3044 19                 DB       BDOS0    ;# SECTORS TO READ ON TRACK 0
 3045 00                 DB       0        ;TRACK 0
 3046 02                 DB       2        ;START WITH SECTOR 2 ON TRACK 0
 3047 0000               DW       CPMB     ;START AT BASE OF BDOS
 0007            IOPBL   EQU      $-IOPB0
                 ;
 3049 80         IOPB1:  DB       80H
 304A 04                 DB       READF
 304B 18                 DB       BDOS1    ;SECTORS TO READ ON TRACK 1
 304C 01                 DB       1        ;TRACK 1
 304D 01                 DB       1        ;SECTOR 1
 304E 800C               DW       CPMB+BDOS0*128  ;BASE OF SECOND READ
                 ;
 3050                    END

0
```



```
 078 BASE        0806 BDOS        0019 BDOS0       0018 BDOS1       1800 BDOSE
1800 BDOSL       0031 BDOSS       0000 BIAS        1600 BOOT        00FF BSW
3007 COLDSTART   0000 CPMB        0078 DSTAT       0000 FALSE       007A IHIGH
0079 ILOW        3042 IOPB0       3049 IOPB1       0007 IOPBL       F800 MON80
0002 NTRKS       1603 RBOOT       007E RBYTE       0004 READF       0003 RECAL
007F RESET       FF0F RMON80      3000 RSTART      0079 RTYPE       0100 STACK
3015 START       0000 TESTING     FFFF TRUE        301B WAIT0
```

```
                        TITLE   'console command processor (CCP), ver 2.0'
                ;       ASSEMBLY LANGUAGE VERSION OF THE CP/M CONSOLE COMMAND PR
                ;
                ;       VERSION 2.0 JULY, 1979
                ;
                ;       COPYRIGHT (C) 1976, 1977, 1978, 1979
                ;       DIGITAL RESEARCH
                ;       BOX 579, PACIFIC GROVE,
                ;       CALIFORNIA 93950
                ;
0000 =          FALSE   EQU     0000H
FFFF =          TRUE    EQU     NOT FALSE
0000            TESTING EQU     FALSE   ;TRUE IF DEBUGGING
                ;
                ;
                        IF      TESTING
                        ORG     3400H
                BDOSL   EQU     $+800H          ;BDOS LOCATION
                        ELSE
0000                    ORG     000H
0800 =          BDOSL   EQU     $+800H          ;BDOS LOCATION
                        ENDIF
0100 =          TRAN    EQU     100H
0000            TRANM   EQU     $
0000 =          CCPLOC  EQU     $
                ;
                ;       ***************************************************
                ;       *       BASE OF CCP CONTAINS THE FOLLOWING CODE/DATA    *
                ;       *       CCP:    JMP CCPSTART    ;START WITH COMMAND     *
                ;       *               JMP CCPCLEAR    ;START, CLEAR COMMAND   *
                ;       *       CCP+6   127             ;MAX COMMAND LENGTH     *
                ;       *       CCP+7   COMLEN          ;COMMAND LENGTH = 00)   *
                ;       *       CCP+8   ' ... '         ;16 BLANKS)             *
                ;       ***************************************************
                ;       * NORMAL ENTRY IS AT CCP, WHERE THE COMMAND LINE GIVEN *
                ;       * AT CCP+8 IS EXECUTED AUTOMATICALLY (NORMALLY A NULL  *
                ;       * COMMAND WITH COMLEN = 00).  AN INITIALIZING PROGRAM  *
                ;       * CAN BE AUTOMATICALLY LOADED BY STORING THE COMMAND   *
                ;       * AT CCP+8, WITH THE COMMAND LENGTH AT CCP+7.  IN THIS *
                ;       * CASE, THE CCP EXECUTES THE COMMAND BEFORE PROMPTING  *
                ;       * THE CONSOLE FOR INPUT.  NOTE THAT THE COMMAND IS EXE *
                ;       * CUTED ON BOTH WARM AND COLD STARTS.  WHEN THE COMMAND*
                ;       * LINE IS INITIALIZED, A JUMP TO "JMP CCPCLEAR" DIS-   *
                ;       * ABLES THE AUTOMATIC COMMAND EXECUTION.               *
                ;       ***************************************************
                ;
0000 C35C03             JMP     CCPSTART        ;START CCP WITH POSSIBLE INITIAL
0003 C35803             JMP     CCPCLEAR        ;CLEAR THE COMMAND BUFFER
0006 7F         MAXLEN: DB      127     ;MAX BUFFER LENGTH
0007 00         COMLEN: DB      0       ;COMMAND LENGTH (FILLED IN BY DOS)
                ;       (COMMAND EXECUTED INITIALLY IF COMLEN NON ZERO)
                COMBUF:
0008 2020202020         DB      '       '       ;8 CHARACTER FILL
0010 2020202020         DB      '       '       ;8 CHARACTER FILL
0018 434F505952         DB      'COPYRIGHT (C) 1979, DIGITAL RESEARCH  '; 38
003E                    DS      128-($-COMBUF)
```



```
                ;       TOTAL BUFFER LENGTH IS 128 CHARACTERS
0088 0800       COMADDR:DW      COMBUF  ;ADDRESS OF NEXT TO CHAR TO SCAN
008A            STADDR: DS      2       ;STARTING ADDRESS OF CURRENT FILLFCB REQ
                ;
0004 =          DISKA   EQU     0004H   ;DISK ADDRESS FOR CURRENT DISK
0005 =          BDOS    EQU     0005H   ;PRIMARY BDOS ENTRY POINT
0080 =          BUFF    EQU     0080H   ;DEFAULT BUFFER
005C =          FCB     EQU     005CH   ;DEFAULT FILE CONTROL BLOCK
                ;
0001 =          RCHARF  EQU     1       ;READ CHARACTER FUNCTION
0002 =          PCHARF  EQU     2       ;PRINT CHARACTER FUNCTION
0009 =          PBUFF   EQU     9       ;PRINT BUFFER FUNCTION
000A =          RBUFF   EQU     10      ;READ BUFFER FUNCTION
000B =          BREAKF  EQU     11      ;BREAK KEY FUNCTION
000C =          LIFTF   EQU     12      ;LIFT HEAD FUNCTION (NO OPERATION)
000D =          INITF   EQU     13      ;INITIALIZE BDOS FUNCTION
000E =          SELF    EQU     14      ;SELECT DISK FUNCTION
000F =          OPENF   EQU     15      ;OPEN FILE FUNCTION
0010 =          CLOSEF  EQU     16      ;CLOSE FILE FUNCTION
0011 =          SEARF   EQU     17      ;SEARCH FOR FILE FUNCTION
0012 =          SFARNF  EQU     18      ;SEARCH FOR NEXT FILE FUNCTION
0013 =          DELF    EQU     19      ;DELETE FILE FUNCTION
0014 =          DREADF  EQU     20      ;DISK READ FUNCTION
0015 =          DWRITF  EQU     21      ;DISK WRITE FUNCTION
0016 =          MAKEF   EQU     22      ;FILE MAKE FUNCTION
0017 =          RENF    EQU     23      ;RENAME FILE FUNCTION
0018 =          LOGF    EQU     24      ;RETURN LOGIN VECTOR
0019 =          CSELF   EQU     25      ;RETURN CURRENTLY SELECTED DRIVE NUMBER
001A =          DMAF    EQU     26      ;SET DMA ADDRESS
0020 =          USERF   EQU     32      ;SET USER NUMBER
                ;
                ;       SPECIAL FCB FLAGS
0009 =          ROFILE  EQU     9       ;READ ONLY FILE
000A =          SYSFILE EQU     10      ;SYSTEM FILE FLAG
                ;
                ;       SPECIAL CHARACTERS
000D =          CR      EQU     13      ;CARRIAGE RETURN
000A =          LF      EQU     10      ;LINE FEED
005F =          LA      EQU     5FH     ;LEFT ARROW
001A =          EOFILE  EQU     1AH     ;END OF FILE
                ;
                ;       UTILITY PROCEDURES
                PRINTCHAR:
008C 5F0F02C305         MOV E,A! MVI C,PCHARF! JMP BDOS
                ;
                PRINTBC:
                        ;PRINT CHARACTER, BUT SAVE B,C REGISTERS
0092 C5CD8C00C1         PUSH B! CALL PRINTCHAR! POP B! RET
                ;
                CRLF:
0098 3E0DCD9200         MVI A,CR! CALL PRINTBC
009D 3E0AC39200         MVI A,LF! JMP  PRINTBC
                ;
                BLANK:
00A2 3E20C39200         MVI A,' '! JMP PRINTBC
                ;
```

```
                PRINT:  ;PRINT STRING STARTING AT B,C UNTIL NEXT 00 ENTRY
00A7 C5CD9800E1         PUSH B! CALL CRLF! POP H ;NOW PRINT THE STRING
00AC 7EB7C8     PRIN0:  MOV A,M! ORA A! RZ ;STOP ON 00
00AF 23E5               INX H! PUSH H ;READY FOR NEXT
00B1 CD8C00E1           CALL PRINTCHAR! POP H ;CHARACTER PRINTED
00B5 C3AC00             JMP PRIN0 ;FOR ANOTHER CHARACTER
                ;
                INITIALIZE:
00B8 0E0DC30500         MVI C,INITF! JMP BDOS
                ;
                SELECT:
00BD 5F0E0EC305         MOV E,A! MVI C SELF! JMP BDOS
                ;
                BDOS$INR:
00C3 CD050032EE         CALL BDOS! STA DCNT! INR A! RET
                ;
                OPEN:   ;OPEN THE FILE GIVEN BY D,E
00CB 0E0FC3C300         MVI C,OPENF! JMP BDOS$INR
                ;
                OPENC:  ;OPEN COMFCB
00D0 AF32ED07           XRA A! STA COMREC ;CLEAR NEXT RECORD TO READ
00D4 11CD07C3CB         LXI D,COMFCB! JMP OPEN
                ;
                CLOSE:  ;CLOSE THE FILE GIVEN BY D E
00DA 0E10C3C300         MVI C,CLOSEF! JMP BDOS$INR
                ;
                SEARCH: ;SEARCH FOR THE FILE GIVEN BY D,E
00DF 0E11C3C300         MVI C SEARF! JMP BDOS$INR
                ;
                SEARCHN:
                        ;SEARCH FOR THE NEXT OCCURRENCE OF THE FILE GIVEN BY D E
00E4 0E12C3C300         MVI C,SEARNF! JMP BDOS$INR
                ;
                SEARCHCOM:
                        ;SEARCH FOR COMFCB FILE
00E9 11CD07C3DF         LXI D COMFCB! JMP SEARCH
                ;
                DELETE: ;DELETE THE FILE GIVEN BY D,E
00EF 0E13C30500         MVI C,DELF! JMP BDOS
                ;
                BDOS$COND:
00F4 CD0500B7C9         CALL BDOS! ORA A! RET
                ;
                DISKREAD:
                        ;READ THE NEXT RECORD FROM THE FILE GIVEN BY D,E
00F9 0E14C3F400         MVI C,DREADF! JMP BDOS$COND
                ;
                DISKREADC:
                        ;READ THE COMFCB FILE
00FE 11CD07C3F9         LXI D,COMFCB! JMP DISKREAD
                ;
                DISKWRITE:
                        ;WRITE THE NEXT RECORD TO THE FILE GIVEN BY D,E
0104 0E15C3F400         MVI C,DWRITEF! JMP BDOS$COND
                ;
                MAKE:   ;CREATE THE FILE GIVEN BY D E
```



```
0109 0E16C3C300         MVI C,MAKEF! JMP BDOS$INR
                ;
                RENAM:  ;RENAME THE FILE GIVEN BY D,E
010E 0E17C30500         MVI C,RENF! JMP BDOS
                ;
                GETUSER:
                        ;RETURN CURRENT USER CODE IN A
0113 1EFF               MVI E 0FFH ;DROP THROUGH TO SETUSER
                ;
                SETUSER:
0115 0E20C30500         MVI C,USERF! JMP BDOS ;SETS USER NUMBER
                ;
                SAVEUSER:
                        ;SAVE USER# DISK# BEFORE POSSIBLE ^C OR TRANSIENT
011A CD1301             CALL GETUSER ;CODE TO A
011D 87878787           ADD A! ADD A! ADD A! ADD A ;ROT LEFT
0121 21EF07B6           LXI H,CDISK! ORA M ;4B=USER, 4B=DISK
0125 320400             STA DISKA ;STORED AWAY IN MEMORY FOR LATER
0128 C9                 RET
                ;
                SETDISKA:
0129 3AEF073204         LDA CDISK! STA DISKA ;USER/DISK
012F C9                 RET
                ;
                TRANSLATE:
                        ;TRANSLATE CHARACTER IN REGISTER A TO UPPER CASE
0130 FE61D8             CPI 61H! RC ;RETURN IF BELOW LOWER CASE A
0133 FE7BD0             CPI 7BH! RNC ;RETURN IF ABOVE LOWER CASE Z
0136 E65FC9             ANI 5FH! RET ;TRANSLATED TO UPPER CASE
                ;
                READCOM:
                        ;READ THE NEXT COMMAND INTO THE COMMAND BUFFER
                        ;CHECK FOR SUBMIT FILE
0139 3AAB07B7CA         LDA SUBMIT! ORA A! JZ NOSUB
                                ;SCANNING A SUBMIT FILE
                                ;CHANGE DRIVES TO OPEN AND READ THE FILE
0140 3AEF07B73E                 LDA CDISK! ORA A! MVI A,0! CNZ SELECT
                                ;HAVE TO OPEN AGAIN IN CASE XSUB PRESENT
0149 11AC07CDCB                 LXI D,SUBFCB! CALL OPEN! JZ NOSUB ;SKIP IF NO SU
0152 3ABB073D                   LDA SUBRC! DCR A ;READ LAST RECORD(S) FIRST
0156 32CC07                     STA SUBCR ;CURRENT RECORD TO READ
0159 11AC07CDF9                 LXI D,SUBFCB! CALL DISKREAD ;END OF FILE IF LAST
015F C29601                     JNZ NOSUB
                                        ;DISK READ IS OK, TRANSFER TO COMBUF
0162 1107002180                         LXI D COMLEN! LXI H,BUFF! MVI B 128! CAL
                                        ;LINE IS TRANSFERRED, CLOSE THE FILE WIT
                                        ;DELETED RECORD
016D 21BA073600                         LXI H,SUBMOD! MVI M,0 ;CLEAR FWFLAG
0172 2335                               INX H! DCR M ;ONE LESS RECORD
0174 11AC07CDDA                         LXI D,SUBFCB! CALL CLOSE! JZ NOSUB
                                        ;CLOSE WENT OK  RETURN TO ORIGINAL DRIVE
017D 3AEF07B7C4                         LDA CDISK! ORA A! CNZ SELECT
                                        ;PRINT TO THE 00
0184 210800CDAC                         LXI H,COMBUF! CALL PRIN0
018A CDC201CAA7                         CALL BREAK$KEY! JZ NOREAD
0190 CDDD01C382                         CALL DEL$SUB! JMP CCP ;BREAK KEY DEPRESS
```

```
CP/M MACRO ASSEM 2.0     #005     console command processor (CCP), ver 2.0

                                  ;
0196 CDDD01             NOSUB:  ;NO SUBMIT FILE! CALL DEL$SUB
                        ;TRANSLATE TO UPPER CASE  STORE ZERO AT END
0199 CD1A01             CALL SAVEUSER ;USER # SAVE IN CASE CONTROL C
019C 0E0A110600         MVI C RBUFF! LXI D MAXLEN! CALL BDOS
01A4 CD2901             CALL SETDISKA ;NO CONTROL C, SO RESTORE DISKA
                        NOREAD: ;ENTER HERE FROM SUBMIT FILE
                        ;SET THE LAST CHARACTER TO ZERO FOR LATER SCANS
01A7 21070046           LXI H COMLEN! MOV B M ;LENGTH IS IN B
01AB 2370B7             READCOM0: INX H! MOV A,B! ORA A ;END OF SCAN?
01AE CABA017E                   JZ READCOM1! MOV A M ;GET CHARACTER AND TRANSLAT
01B2 CD30017705                 CALL TRANSLATE! MOV M,A! DCR B! JMP READCOM0
                                ;
                        READCOM1: ;END OF SCAN, H,L ADDRESS END OF COMMAND
01BA 77                         MOV M A ;STORE A ZERO
01BB 210E002288                 LXI H,COMBUF! SHLD COMADDR ;READY TO SCAN TO ZER
01C1 C9                 RET
                ;
                BREAK$KEY:
                        ;CHECK FOR A CHARACTER READY AT THE CONSOLE
01C2 0E0BCD0500         MVI C,BREAKF! CALL BDOS
01C7 B7C8               ORA A! RZ
01C9 0E01CD0500         MVI C RCHARF! CALL BDOS ;CHARACTER CLEARED
01CE B7C9               ORA A! RET
                ;
                CSELECT:
                        ;GET THE CURRENTLY SELECTED DRIVE NUMBER TO REG A
01D0 0E19C30500         MVI C,CSELF! JMP BDOS
                ;
                SETDMABUFF:
                        ;SET DEFAULT BUFFER DMA ADDRESS
01D5 110000             LXI D,BUFF ;(DROP THROUGH)
                ;
                SETDMA
                        ;SET DMA ADDRESS TO D E
01D8 0E1AC30500         MVI C,DMAF! JMP BDOS
                ;
                DEL$SUB:
                        ;DELETE THE SUBMIT FILE  AND SET SUBMIT FLAG TO FALSE
01DD 21AB077FB7         LXI H,SUBMIT! MOV A,M! ORA A! RZ ;RETURN IF NO SUB FILE
01E3 3600               MVI M,0 ;SUBMIT FLAG IS SET TO FALSE
01E5 AFCDBD00           XRA A! CALL SELECT ;ON DRIVE A TO ERASE FILE
01E9 11AC07CDEF         LXI D,SUBFCB! CALL DELETE
01EF 3AEF07C3BD         LDA CDISK! JMP SELECT ;BACK TO ORIGINAL DRIVE
                ;
                SERIALIZE:
                        ;CHECK SERIALIZATION
01F5 112E032100         LXI D,SERIAL! LXI H,BDOSL! MVI B,6 ;CHECK SIX BYTES
01FD 1ABEC2CF03         SER0:   LDAX D! CMP M! JNZ BADSERIAL
0202 132305C2FD                 INX D! INX H! DCR B! JNZ SER0
0208 C9                         RET ;SERIAL NUMBER IS OK
                ;
                COMERR:
                        ;ERROR IN COMMAND STRING STARTING AT POSITION
                        ;'STADDR' AND ENDING WITH FIRST DELIMITER
0209 CD9E00             CALL CRLF ;SPACE TO NEXT LINE
```



```
CP/M MACRO ASSEM 2.0     #006     console command processor (CCP), ver 2.0

020C 2A8A00             LHLD STADDR ;H L ADDRESS FIRST TO PRINT
                        COMERR0: ;PRINT CHARACTERS UNTIL BLANK OR ZERO
020F 7EFE20CA22                 MOV A,M! CPI ' '! JZ COMERR1; NOT BLANK
0215 B7CA2202                   ORA A! JZ COMERR1; NOT ZERO, SO PRINT IT
0219 E5CD8C00E1                 PUSH H! CALL PRINTCHAR! POP H! INX H
021F C30F02                     JMP COMERR0; FOR ANOTHER CHARACTER
                        COMERR1: ;PRINT QUESTION MARK,AND DELETE SUB FILE
0222 3E3FCD8C00                 MVI A '?'! CALL PRINTCHAR
0227 CD9E00CDDD                 CALL CRLF! CALL DEL$SUB
022D C38203                     JMP CCP ;RESTART WITH NEXT COMMAND
                ;
                ; FCB SCAN AND FILL SUBROUTINE (ENTRY IS AT FILLFCB BELOW)
                        ;FILL THE COMFCB, INDEXED BY A (0 OR 16)
                        ;SUBROUTINES
                        DELIM:  ;LOOK FOR A DELIMITER
0230 1AB7C8                     LDAX D! ORA A! RZ ;NOT THE LAST ELEMENT
0233 FE20DA0902                 CPI ' '! JC COMERR ;NON GRAPHIC
0238 C8                         RZ ;TREAT BLANK AS DELIMITER
0239 FE3DC8                     CPI '='! RZ
023C FE5FC8                     CPI LA!  RZ ;LEFT ARROW
023F FE2EC8                     CPI '.'! RZ
0242 FE3AC8                     CPI ':'! RZ
0245 FE3BC8                     CPI ';'! RZ
0248 FE3CC8                     CPI '<'! RZ
024B FE3EC8                     CPI '>'! RZ
024E C9                         RET     ;DELIMITER NOT FOUND
                ;
                        DEBLANK: ;DEBLANK THE INPUT LINE
024F 1AB7C8                     LDAX D! ORA A! RZ ;TREAT END OF LINE AS BLANK
0252 FE20C013C3                 CPI ' '! RNZ! INX D! JMP DEBLANK
                ;
                        ADDH: ;ADD A TO H L
0259 856FD0                     ADD L! MOV L,A! RNC
025C 24C9                       INR H! RET
                                ;
                FILLFCB0:
                        ;EQUIVALENT TO FILLFCB(0)
025E 3E00               MVI A 0
                ;
                FILLFCB:
0260 21CD07CD59         LXI H,COMFCB! CALL ADDH! PUSH H! PUSH H ;FCB RESCANNED A
0268 AF32F007           XRA A! STA SDISK ;CLEAR SELECTED DISK (IN CASE A:...)
026C 2A8E00EB           LHLD COMADDR! XCHG ;COMMAND ADDRESS IN D,E
0270 CD4F02             CALL DEBLANK ;TO FIRST NON BLANK CHARACTER
0273 EB228A00           XCHG! SHLD STADDR ;IN CASE OF ERRORS
0277 EBE1               XCHG! POP H ;D E HAS COMMAND  H,L HAS FCB ADDRESS
                        ;LOOK FOR PRECEDING FILE NAME A: B: ...
0279 1AB7CA8902         LDAX D! ORA A! JZ SETCUR0 ;USE CURRENT DISK IF EMPTY COM
027E DE4047             SBI 'A'-1! MOV B,A ;DISK NAME HELD IN B IF : FOLLOWS
0281 131AFE3ACA         INX D! LDAX D! CPI ':'! JZ SETDSK ;SET DISK NAME IF :
                        ;
                        SETCUR: ;SET CURRENT DISK
0288 1B                         DCX D ;BACK TO FIRST CHARACTER OF COMMAND
                        SETCUR0:
0289 3AEF0777C3                 LDA CDISK! MOV M,A! JMP SETNAME
                        ;
```

```
CP/M MACRO ASSEM 2.0     #007    console command processor (CCP), ver 2.0

                         SETDSK  ;SET DISK TO NAME IN REGISTER B
0290 7832F007                    MOV A,B! STA SDISK ;MARK AS DISK SELECTED
0294 7013                        MOV M,B! INX D ;PAST THE :
                         ;
                         SETNAME: ;SET THE FILE NAME FIELD
0296 0608                        MVI B,8 ;FILE NAME LENGTH (MAX)
0298 CD3002CAB9                  SETNAM0: CALL DELIM! JZ PADNAME ;NOT A DELIMITER
029E 23FE2AC2A9                          INX H! CPI '*'! JNZ SETNAM1 ;MUST BE ?'S
02A4 363FC3AB02                          MVI M,'?'! JMP SETNAM2 ;TO DEC COUNT
                                 ;
02A9 7713                        SETNAM1: MOV M,A ;STORE CHARACTER TO FCB! INX D
02AB 05C29802                    SETNAM2: DCR B ;COUNT DOWN LENGTH! JNZ SETNAM0
                                 ;
                         ;END OF NAME.  TRUNCATE REMAINDER
02AF CD3002CAC0          TRNAME: CALL DELIM! JZ SETTY ;SET TYPE FIELD IF DELIMITE
02B5 13C3AF02                    INX D! JMP TRNAME
                                 ;
02B9 2336200SC2          PADNAME: INX H! MVI M ' '! DCR B! JNZ PADNAME
                                 ;
                         SETTY: ;SET THE TYPE FIELD
02C0 0603FE2EC2                  MVI B,3! CPI '.'! JNZ PADTY ;SKIP THE TYPE FIELD
02C7 13                          INX D ;PAST THE .  TO THE FILE TYPE FIELD
                                 SETTY0: ;SET THE FIELD FROM THE COMMAND BUFFER
02C8 CD3002CAE9                          CALL DELIM! JZ PADTY! INX H! CPI '*'! JN
02D4 363FC3DB02                          MVI M,'?' ;SINCE * SPECIFIED! JMP SETTY2
                                         ;
                                 SETTY1: ;NOT A *, SO COPY TO TYPE FIELD
02D9 7713                                MOV M A! INX D
                                 SETTY2: ;DECREMENT COUNT AND GO AGAIN
02DB 05C2C802                            DCR B! JNZ SETTY0
                                         ;
                                 ;END OF TYPE FIELD, TRUNCATE
                         TRTYP: ;TRUNCATE TYPE FIELD
02DF CD3002CAF0                  CALL DELIM! JZ EFILL! INX D! JMP TRTYP
                                 ;
                                 PADTY:  ;PAD THE TYPE FIELD WITH BLANKS
02E9 2336200SC2                          INX H! MVI M,' '! DCR B! JNZ PADTY
                                 ;
                         EFILL: ;END OF THE FILENAME/FILETYPE FILL, SAVE COMMAND
                                 ;FILL THE REMAINING FIELDS FOR THE FCB
02F0 0603                        MVI B,3
02F2 2336000SC2                  EFILL0: INX H! MVI M,0! DCR B! JNZ EFILL0
02F9 EB22E800                    XCHG! SHLD COMADDR ;SET NEW STARTING POINT
                                 ;
                                 ;RECOVER THE START ADDRESS OF THE FCB AND COUNT
02FD E1010E00                    POP H! LXI B 11 ;B=0  C=8+3
0301 237EFE3FC2                  SCNQ: INX H! MOV A,M! CPI '?'! JNZ SCNQ0
0308 04                          ;? FOUND  COUNT IT IN B! INR B
0309 0DC20103                    SCNQ0: DCR C! JNZ SCNQ
                                 ;
                                 ;NUMBER OF ?'S IN C, MOVE TO A AND RETURN WITH F
030D 78B7C9                      MOV A,B! ORA A! RET
                 ;
                 INTVEC:
                         ;INTRINSIC FUNCTION NAMES (ALL ARE FOUR CHARACTERS
0310 44495220            DB      'DIR '
```

```
CP/M MACRO ASSEM 2.0     #008    console command processor (CCP), ver 2.0

0314 45524120            DB      'ERA '
0318 54595045            DB      'TYPE'
031C 53415645            DB      'SAVE'
0320 52454E20            DB      'REN '
0324 55534552            DB      'USER'
0006 =                   INTLEN EQU ($-INTVEC)/4 ;INTRINSIC FUNCTION LENGTH
0328 0000000000          SERIAL: DB 0,0,0,0,0,0
                 ;
                 ;
                 INTRINSIC:
                         ;LOOK FOR INTRINSIC FUNCTIONS (COMFCB HAS BEEN FILLED
032E 2110003E00          LXI H,INTVEC! MVI C 0 ;C COUNTS INTRINSICS AS SCANNED
0333 79FE06D0            INTRIN0: MOV A,C! CPI INTLEN ;DONE WITH SCAN?! RNC
                                  ;NO  MORE TO SCAN
0337 11CF07                       LXI D,COMFCB+1 ;BEGINNING OF NAME
033A 0604                         MVI B,4 ;LENGTH OF MATCH IS IN B
033C 1ABE                         INTRIN1: LDAX D! CMP M ;MATCH?
033E C24F03                               JNZ INTRIN2 ;SKIP IF NO MATCH
0341 132305                               INX D! INX H! DCR B
0344 C23C03                               JNZ INTRIN1 ;LOOP WHILE MATCHING
                                  ;
                                  ;COMPLETE MATCH ON NAME.  CHECK FOR BLANK IN FCB
0347 1AFE20C254                   LDAX D! CPI ' '! JNZ INTRIN3 ;OTHERWISE MATCHED
034D 79C9                         MOV A,C! RET ;WITH INTRINSIC NUMBER IN A
                                  ;
                                  INTRIN2: ;MISMATCH.  MOVE TO END OF INTRINSIC
034F 2305C24F03                           INX H! DCR B! JNZ INTRIN2
                                  ;
                                  INTRIN3: ;TRY NEXT INTRINSIC
0354 0C                                   INR C ;TO NEXT INTRINSIC NUMBER
0355 C33303                               JMP INTRIN0 ;FOR ANOTHER ROUND
                 ;
                 CCPCLEAR:
                         ;CLEAR THE COMMAND BUFFER
0358 AF                  XRA     A
0359 320700              STA     COMLEN
                         ;DROP THROUGH TO START CCP
                 CCPSTART:
                         ;ENTER HERE FROM BOOT LOADER
035C 31AB07C5            LXI SP,STACK! PUSH B ;SAVE INITIAL DISK NUMBER
                         ;(HIGH ORDER 4BITS=USER CODE, LOW 4BITS=DISK#)
0360 791F1F1F1F          MOV A,C! RAR! RAR! RAR! RAR! ANI 0FH ;USER CODE
0367 5FCD1501            MOV E,A! CALL SETUSER ;USER CODE SELECTED
                         ;INITIALIZE FOR THIS USER, GET $ FLAG
036B CDB800              CALL INITIALIZE ;0FFH IN ACCUM IF $ FILE PRESENT
036E 32AB07              STA SUBMIT ;SUBMIT FLAG SET IF $ FILE PRESENT
0371 C1                  POP B ;RECALL USER CODE AND DISK NUMBER
0372 79E60F              MOV A,C! ANI 0FH ;DISK NUMBER IN ACCUMULATOR
0375 320400              STA DISKA ;CLEARS LOW MEMORY USER CODE NIBBLE
0378 CDBD00              CALL SELECT ;PROPER DISK IS SELECTED  NOW CHECK SUB FILE
                         ;CHECK FOR INITIAL COMMAND
037B 3A0700B7C2          LDA COMLEN! ORA A! JNZ CCP0      ;ASSUME TYPED ALREADY
                 ;
                 CCP:
                         ;ENTER HERE ON EACH COMMAND OR ERROR CONDITION
0382 31AB07              LXI SP,STACK
```

```
CP/M MACRO ASSEM 2.0    #009    console command processor (CCP), ver 2.0

0385 CD9E00             CALL CRLF ;PRINT D> PROMPT, WHERE D IS DISK NAME
0388 CDDD01             CALL CSELECT ;GET CURRENT DISK NUMBER
038B C641CD8C00         ADI 'A'! CALL PRINTCHAR
0390 3E3ECD8C00         MVI A,'>'! CALL PRINTCHAR
0395 CD3901             CALL READCOM ;COMMAND BUFFER FILLED
                CCP0:   ;(ENTER HERE FROM INITIALIZATION WITH COMMAND FULL)
039E 118000CDD8         LXI D,BUFF! CALL SETDMA ;DEFAULT DMA ADDRESS AT BUFF
03AE CDD901325F         CALL CSELECT! STA CDISK ;CURRENT DISK NUMBER SAVED
03A4 CD5E02             CALL FILLFCB0 ;COMMAND FCB FILLED
03A7 C40902             CNZ COMERR ;THE NAME CANNOT BE AN AMBIGUOUS REFERENCE
03AA 3AF007B7C2         LDA SDISK! ORA A! JNZ USERFUNC
                        ;CHECK FOR AN INTRINSIC FUNCTION
03B1 CD2E03             CALL INTRINSIC
03B4 21C103             LXI H,JMPTAB ;INDEX IS IN THE ACCUMULATOR
03B7 5F16001919         MOV E,A! MVI D 0! DAD D! DAD D ;INDEX IN D E
03BC 7E23666FE9         MOV A,M! INX H! MOV H,M! MOV L,A! PCHL
                        ;PC CHANGES TO THE PROPER INTRINSIC OR USER FUNC
                        JMPTAB:
03C1 7704                       DW      DIRECT  ;DIRECTORY SEARCH
03C3 1F05                       DW      ERASE   ;FILE ERASE
03C5 5D05                       DW      TYPE    ;TYPE FILE
03C7 AD05                       DW      SAVE    ;SAVE MEMORY IMAGE
03C9 1006                       DW      RENAME  ;FILE RENAME
03CB 8F06                       DW      USER    ;USER NUMBER
03CD A506                       DW      USERFUNC;USER-DEFINED FUNCTION
                        BADSERIAL:
03CF 21F376                     LXI H,0 DI OR (HLT SHL 8)
03D2 2200002100                 SHLD CCPLOC! LXI H,CCPLOC! PCHL
                                ;

                ;
                ;UTILITY SUBROUTINES FOR INTRINSIC HANDLERS
                READERR:
                        ;PRINT THE READ ERROR MESSAGE
03D9 01DF03C3A7                 LXI B,RDMSG! JMP PRINT
03DF 5245414420                 RDMSG: DB 'READ ERROR',0
                ;
                NOFILE:
                        ;PRINT NO FILE MESSAGE
03EA 01F003C3A7                 LXI B,NOFMSG! JMP PRINT
03F0 4E4F204649                 NOFMSG: DB 'NO FILE',0
                ;
                GETNUMBER: ;READ A NUMBER FROM THE COMMAND LINE
03F8 CD5E02             CALL FILLFCB0 ;SHOULD BE NUMBER
03FB 3AF007B7C2         LDA SDISK! ORA A! JNZ COMERR ;CANNOT BE PREFIXED
                        ;CONVERT THE BYTE VALUE IN COMFCB TO BINARY
0402 21CE07010B         LXI H,COMFCB+1! LXI B,11 ;(B=0, C=11)
                        ;VALUE ACCUMULATED IN B  C COUNTS NAME LENGTH TO
040B 7EFE20CA33         CONV0:  MOV A,M! CPI ' '! JZ CONV1
                                ;MORE TO SCAN  CONVERT CHAR TO BINARY AN
040E 23D630FE0A                 INX H! SUI '0'! CPI 10! JNC COMERR ;VALI
0416 5778                       MOV D,A ;SAVE VALUE! MOV A,B ;MULT BY 10
0418 E6F0C20902                 ANI 1110$0000B! JNZ COMERR
041D 78                         MOV A,B ;RECOVER VALUE
041E 070707                     RLC! RLC! RLC ;*8
0421 80DA0902                   ADD B! JC COMERR
0425 80DA0902                   ADD B! JC COMERR ;*8+*2 = *10
```



```
CP/M MACRO ASSEM 2.0    #010    console command processor (CCP), ver 2.0

0429 82DA0902                   ADD D! JC COMERR ;+DIGIT
042D 470EC20804                 MOV B,A! DCR C! JNZ CONV0 ;FOR ANOTHER D
0432 C9                         RET
                        CONV1:  ;END OF DIGITS, CHECK FOR ALL BLANKS
0433 7EFE20C209                 MOV A,M! CPI ' '! JNZ COMERR ;BLANKS?
0439 230DC23304                 INX H! DCR C! JNZ CONV1
043E 78C9                       MOV A,B ;RECOVER VALUE! RET
                        ;
                MOVENAME:
                        ;MOVE 3 CHARACTERS FROM H L TO D.F ADDRESSES
0440 0603               MVI B,3
0442 7E122313           MOVE0: MOV A,M! STAX D! INX H! INX D
0446 05C24204                   DCR B! JNZ MOVE0
044A C9                 RET
                ;
                ADDHCF: ;BUFF + A + C TO H.L FOLLOWED BY FETCH
044B 21e000e1CD         LXI H,BUFF! ADD C! CALL ADDH! MOV A,M! RET
                ;
                SETDISK:
                        ;CHANGE DISKS FOR THIS COMMAND, IF REQUESTED
0454 AF32CD07           XRA A! STA COMFCB ;CLEAR DISK NAME FROM FCB
0458 3AF007B7C8         LDA SDISK! ORA A! RZ ;NO ACTION IF NOT SPECIFIED
045D 3D21EF07BE         DCR A! LXI H,CDISK! CMP M! RZ ;ALREADY SELECTED
0463 C3BD00             JMP SELECT
                ;
                RESETDISK:
                        ;RETURN TO ORIGINAL DISK AFTER COMMAND
0466 3AF007B7C8         LDA SDISK! ORA A! RZ ;NO ACTION IF NOT SELECTED
046B 3D21EF07BE         DCR A! LXI H,CDISK! CMP M! RZ ;SAME DISK
0471 3AEF07C3BD         LDA CDISK! JMP SELECT
                        ;
                ;INDIVIDUAL INTRINSICS FOLLOW
                DIRECT:
                ;DIRECTORY SEARCH
0477 CD5E02             CALL FILLFCB0 ;COMFCB GETS FILE NAME
047A CD5404             CALL SETDISK ;CHANGE DISK DRIVES IF REQUESTED
047D 21CE077E           LXI H,COMFCB+1! MOV A,M ;MAY BE EMPTY REQUEST
0481 FE20C28F04         CPI ' '! JNZ DIR1 ;SKIP FILL OF ??? IF NOT BLANK
                        ;SET COMFCB TO ALL ??? FOR CURRENT DISK
0486 060B               MVI B,11 ;LENGTH OF FILL ????????.???
0488 363F2305C2         DIR0: MVI M,'?'! INX H! DCR B! JNZ DIR0
                ;NOT A BLANK REQUEST  MUST BE IN COMFCB
048F 1E00D5             DIR1:   MVI E,0! PUSH D ;E COUNTS DIRECTORY ENTRIES
0492 CDE900             CALL SEARCHCOM ;FIRST ONE HAS BEEN FOUND
0495 CCEA03             CZ NOFILE ;NOT FOUND MESSAGE
0498 CA1B05             DIR2:   JZ ENDIR
                        ;FOUND, BUT MAY BE SYSTEM FILE
049B 3AEE07             LDA DCNT ;GET THE LOCATION OF THE ELEMENT
049E 0F0F0FE660         RRC! RRC! RRC! ANI 110$0000B! MOV C,A
                        ;C CONTAINS BASE INDEX INTO BUFF FOR DIR ENTRY
04A4 3E0ACD4B04         MVI A,SYSFILE! CALL ADDHCF ;VALUE TO A
04A9 17DA0F05           RAL! JC DIR6 ;SKIP IF SYSTEM FILE
                        ;C HOLDS INDEX INTO BUFFER
                        ;ANOTHER FCB FOUND  NEW LINE?
04AD D17B1CD5           POP D! MOV A,E! INR E! PUSH D
                        ;E=0 1 2,3....NEW LINE IF MOD 4 = 0
```

```
CP/M MACRO ASSEM 2.0    #011    console command processor (CCP), ver 2.0

04B1 E607F5                     ANI 11B! PUSH PSW ;AND SAVE THE TEST
04B4 C2CC04                             JNZ DIRHDR0 ;HEADER ON CURRENT LINE
04B7 CD9E00                             CALL CRLF
04BA C5CDD001C1                         PUSH B! CALL CSELECT! POP B
                                        ;CURRENT DISK IN A
04BF C641CD9200                         ADI  'A'! CALL PRINTBC
04C4 3E3ACD9200                         MVI A,':'! CALL PRINTBC
04C9 C3D404                             JMP DIRHDR1 ;SKIP CURRENT LINE HDR
04CC CDA200                     DIRHDR0:CALL BLANK ;AFTER LAST ONE
04CF 3E3ACD9200                         MVI A,':'! CALL PRINTBC
                                DIRHDR1:
04D4 CDA200                             CALL BLANK
                                ;COMPUTE POSITION OF NAME IN BUFFER
04D7 0601                       MVI B,1 ;START WITH FIRST CHARACTER OF NAME
04D9 78CD4B04                   DIR3:   MOV A,B! CALL ADDHCF ;BUFF+A+C FETCHED
04DD E67F                               ANI 7FH ;MASK FLAGS
                                        ;MAY DELETE TRAILING BLANKS
04DF FE20C2F904                         CPI ' '! JNZ DIR4 ;CHECK FOR BLANK TYPE
04E4 F1F5                               POP PSW! PUSH PSW ;MAY BE 3RD ITEM
04E6 FE03C2F704                         CPI 3! JNZ DIRB ;PLACE BLANK AT END IF N
04EB 3E09CD4B04                         MVI A,9! CALL ADDHCF ;FIRST CHAR OF TYPE
04F0 E67FFE20CA                         ANI 7FH! CPI ' '! JZ DIR5
                                        ;NOT A BLANK IN THE FILE TYPE FIELD
04F7 3E20                       DIRB:   MVI A,' ' ;RESTORE TRAILING FILENAME CHR
                                DIR4:
04F9 CD9200                             CALL PRINTBC ;CHAR PRINTED
04FC 0478FE0CD2                         INR B! MOV A,B! CPI 12! JNC DIR5
                                        ;CHECK FOR BREAK BETWEEN NAMES
0503 FE09C2D904                         CPI 9! JNZ DIR3 ;FOR ANOTHER CHAR
                                        ;PRINT A BLANK BETWEEN NAMES
0508 CDA200C3D9                         CALL BLANK! JMP DIR3
                                ;
                        DIR5:   ;END OF CURRENT ENTRY
050E F1                         POP PSW ;DISCARD THE DIRECTORY COUNTER (MOD 4
050F CDC201             DIR6:   CALL BREAK$KEY ;CHECK FOR INTERRUPT AT KEYBOARD
0512 C21B05                     JNZ ENDIR ;ABORT DIRECTORY SEARCH
0515 CDE400C398                 CALL SEARCHN! JMP DIR2 ;FOR ANOTHER ENTRY
                        ENDIR:  ;END OF DIRECTORY SCAN
051B D1                         POP D ;DISCARD DIRECTORY COUNTER
051C C38E07                     JMP RETCOM
                ;
                ;
                ERASE:
051F CD5E02             CALL FILLFCB0 ;CANNOT BE ALL ???'S
0522 FE0B               CPI 11
0524 C24205             JNZ ERASEFILE
                                ;ERASING ALL OF THE DISK
0527 015205CDA7                 LXI B,ERMSG! CALL PRINT!
052D CD3901                     CALL READCOM
0530 2107003SC2                 LXI H,COMLEN! DCR M! JNZ CCP ;BAD INPUT
0537 237EFE59C2                 INX H! MOV A,M! CPI 'Y'! JNZ CCP
                                ;OK, ERASE THE ENTIRE DISKETTE
053E 23228800                   INX H! SHLD COMADDR ;OTHERWISE ERROR AT RETCOM
                        ERASEFILE:
0542 CD5404                     CALL SETDISK
0545 11CD07CDEF                 LXI D,COMFCB! CALL DELETE
```



```
CP/M MACRO ASSEM 2.0    #012    console command processor (CCP), ver 2.0

                                INR A ;255 RETURNED IF NOT FOUND
054C CCFA03                     CZ NOFILE ;NO FILE MESSAGE IF SO
054F C38607                     JMP RETCOM
                ;
0552 414C4C2028         ERMSG:  DB      'ALL (Y/N)?',0
                ;
                TYPE:
055D CD5E02C209         CALL FILLFCB0! JNZ COMERR ;DON'T ALLOW ?'S IN FILE NAME
0563 CD5404CDD0         CALL SETDISK! CALL OPENC ;OPEN THE FILE
0569 CAA705             JZ TYPERR ;ZERO FLAG INDICATES NOT FOUND
                                ;FILE OPENED. READ 'TIL EOF
056C CD9E0021F1                 CALL CRLF! LXI H,BPTR! MVI M,255 ;READ FIRST BUF
                                TYPE0:  ;LOOP ON BPTR
0574 21F1077EFE                         LXI H,BPTR! MOV A,M! CPI 128 ;END BUFFER
057A DA8705E5                           JC TYPE1! PUSH H ;CARRY IF 0,1,... 127
                                        ;READ ANOTHER BUFFER FULL
057E CDFE00E1                           CALL DISKREADC! POP H ;RECOVER ADDRESS O
0582 C2A005                             JNZ TYPEOF ;HARD END OF FILE
0585 AF77                               XRA A! MOV M,A ;BPTR = 0
                                TYPE1:  ;READ CHARACTER AT BPTR AND PRINT
0587 34                                 INR M ;BPTR = BPTR + 1
0588 216000CD59                         LXI H,BUFF! CALL ADDH ;H,L ADDRESSES CHA
058E 7EFE1ACA86                         MOV A,M! CPI EOFILE! JZ RETCOM
0594 CD0C00                             CALL PRINTCHAR
0597 CDC201C286                         CALL BREAK$KEY! JNZ RETCOM ;ABORT IF BRE
059D C37405                             JMP TYPE0 ;FOR ANOTHER CHARACTER
                                ;
                                TYPEOF: ;END OF FILE. CHECK FOR ERRORS
05A0 3DCA8607                           DCR A! JZ RETCOM
05A4 CDD903                             CALL READERR
05A7 CD6604C309                 TYPERR: CALL RESETDISK! JMP COMERR
                ;
                SAVE:
05AD CDF803             CALL GETNUMBER; VALUE TO REGISTER A
05B0 F5                 PUSH PSW ;SAVE IT FOR LATER
                        ;
                        ;SHOULD BE FOLLOWED BY A FILE TO SAVE THE MEMORY
05B1 CD5E02             CALL FILLFCB0
05B4 C20902             JNZ COMERR ;CANNOT BE AMBIGUOUS
05B7 CD5404             CALL SETDISK ;MAY BE A DISK CHANGE
05BA 11CD07D5CD         LXI D,COMFCB! PUSH D! CALL DELETE ;EXISTING FILE
05C1 D1CD0901           POP D! CALL MAKE ;CREATE A NEW FILE ON DISK
05C5 CAFB05             JZ SAVERR ;NO DIRECTORY SPACE
05C8 AF32ED07           XRA A! STA COMREC; CLEAR NEXT RECORD FIELD
05CC F1                 POP PSW ;#PAGES TO WRITE IS IN A, CHANGE TO #SEC
05CD 6F260029           MOV L,A! MVI H,0! DAD H!
05D1 110001             LXI D,TRAN ;H,L IS SECTOR COUNT, D,E IS LOAD ADD
                SAVE0:  ;CHECK FOR SECTOR COUNT ZERO
05D4 7CB5CAF105         MOV A,H! ORA L! JZ SAVE1 ;MAY BE COMPLETED
05D9 2B                 DCX H ;SECTOR COUNT = SECTOR COUNT - 1
05DA E5                 PUSH H ;SAVE IT FOR NEXT TIME AROUND
05DB 21800019E5         LXI H,128! DAD D! PUSH H ;NEXT DMA ADDRESS SAVED
05E0 CDD801             CALL SETDMA ;CURRENT DMA ADDRESS SET
05E3 11CD07CD04         LXI D,COMFCB! CALL DISKWRITE
05E9 D1E1               POP D! POP H ;DMA ADDRESS, SECTOR COUNT
05EB C2FB05             JNZ SAVERR ;MAY BE DISK FULL CASE
```

```
05EF C3D405                     JMP SAVE0 ;FOR ANOTHER SECTOR
                                ;
                        SAVE1:  ;END OF DUMP, CLOSE THE FILE
05F1 11CD07CDDA                 LXI D,COMFCB! CALL CLOSE
05F7 3C                         INR A; 255 BECOMES 00 IF ERROR
05F8 C20106                     JNZ RETSAVE ;FOR ANOTHER COMMAND
                        SAVERR: ;MUST BE FULL OR READ ONLY DISK
05FB 010706CDA7                 LXI B,FULLMSG! CALL PRINT
                        RETSAVE:
                                ;RESET DMA BUFFER
0601 CDD501                     CALL SETDMABUFF
0604 C38607                     JMP RETCOM
0607 4E4F205350         FULLMSG: DB 'NO SPACE',0
                                ;

                ;
                RENAME:
                        ;RENAME A FILE ON A SPECIFIC DISK
0610 CD5E02C209                 CALL FILLFCB0! JNZ COMERR ;MUST BE UNAMBIGUOUS
0616 3AF007F5                   LDA SDISK! PUSH PSW ;SAVE FOR LATER COMPARE
061A CD5404                     CALL SETDISK ;DISK SELECTED
061D CDE900                     CALL SEARCHCOM ;IS NEW NAME ALREADY THERE?
0620 C27906                     JNZ RENERR3
                                ;FILE DOESN'T EXIST  MOVE TO SECOND HALF OF FCB
0623 21CD0711DD                 LXI H,COMFCB! LXI D,COMFCB+16! MVI B,16! CALL MO
                                ;CHECK FOR = OR LEFT ARROW
062F 2AE800EBCD                 LHLD COMADDR! XCHG! CALL DEBLANK
0635 FE3DCA3F06                 CPI '='! JZ REN1 ;OK IF =
063A FE5FC27306                 CPI LA! JNZ RENERR2
063F EB23228800         REN1:   XCHG! INX H! SHLD COMADDR ;PAST DELIMITER
                                ;PROPER DELIMITER FOUND
0644 CD5E02C273                 CALL FILLFCB0! JNZ RENERR2
                                ;CHECK FOR DRIVE CONFLICT
064A F147                               POP PSW! MOV B,A ;PREVIOUS DRIVE NUMBER
064C 21F0077EB7                         LXI H,SDISK! MOV A,M! ORA A! JZ REN2
                                        ;DRIVE NAME WAS SPECIFIED.  SAME ONE?
0654 B870C27306                         CMP B! MOV M,B! JNZ RENERR2
0659 70                 REN2:   MOV M,B ;STORE THE NAME IN CASE DRIVES SWITCHED
065A AF32CD07CD                 XRA A! STA COMFCB! CALL SEARCHCOM ;IS OLD FILE T
0661 CA6D06                     JZ RENERR1
                                ;
                                ;EVERYTHING IS OK  RENAME THE FILE
0664 11CD07CD0E                 LXI D,COMFCB! CALL RENAM
066A C38607                     JMP RETCOM
                                ;
                        RENERR1:; NO FILE ON DISK
066D CDEA03C366                 CALL NOFILE! JMP RETCOM
                        RENERR2:; AMBIGOUS REFERENCE/NAME CONFLICT
0673 CD6604C309                 CALL RESETDISK! JMP COMERR
                        RENERR3:; FILE ALREADY EXISTS
0679 018206CDA7                 LXI B,RENMSG! CALL PRINT! JMP RETCOM
0682 46494C4520         RENMSG: DB 'FILE EXISTS',0
                ;
                USER:
                        ;SET USER NUMBER
068E CDF803             CALL GETNUMBER; LEAVES THE VALUE IN THE ACCUMULATOR
0691 FE10D20902         CPI 16! JNC COMERR; MUST BE BETWEEN 0 AND 15
```



```
0696 5F                 MOV E,A ;SAVE FOR SETUSER CALL
0697 3ACE07FE20         LDA COMFCB+1! CPI ' '! JZ COMERR
069F CD1501             CALL SETUSER ;NEW USER NUMBER SET
06A2 C38907             JMP ENDCOM
                ;
                USERFUNC:
06A5 CDF501             CALL SERIALIZE ;CHECK SERIALIZATION
                        ;LOAD USER FUNCTION AND SET UP FOR EXECUTION
06A8 3ACE07FE20         LDA COMFCB+1! CPI ' '! JNZ USER0
                                ;NO FILE NAME, BUT MAY BE DISK SWITCH
06B0 3AF007B7CA                 LDA SDISK! ORA A! JZ ENDCOM ;NO DISK NAME IF 0
06B7 3D32EF07CD                 DCR A! STA CDISK! CALL SETDISKA ;SET USER/DISK
06BE CDBD00C389                 CALL SELECT! JMP ENDCOM
                        USER0:  ;FILE NAME IS PRESENT
06C4 11D6071AFE                 LXI D,COMFCB+9! LDAX D! CPI ' '! JNZ COMERR ;TYP
06CD D5CD5404D1                 PUSH D! CALL SETDISK! POP D! LXI H,COMTYPE ;.COM
06D5 CD4004                     CALL MOVENAME ;FILE TYPE IS SET TO .COM
06D8 CDD900CA6B                 CALL OPENC! JZ USERER
                                ;FILE OPENED PROPERLY, READ IT INTO MEMORY
06DE 210001                     LXI H,TRAN ;TRANSIENT PROGRAM BASE
06E1 E5                         LOAD0:  PUSH H ;SAVE DMA ADDRESS
06E2 EBCDD801                           XCHG! CALL SETDMA
06E6 11CD07CDF9                         LXI D,COMFCB! CALL DISKREAD! JNZ LOAD1
                                        ;SECTOR LOADED, SET NEW DMA ADDRESS AND
06EF E111800019                         POP H! LXI D,128! DAD D
06F4 110000                             LXI D TRANM ;HAS THE LOAD OVERFLOWED?
06F7 7D937C9AD2                         MOV A,L! SUB E! MOV A,H! SBB D! JNC LOAD
06FE C3E106                             JMP LOAD0 ;FOR ANOTHER SECTOR
                                        ;
0701 E13DC27107                 LOAD1:  POP H! DCR A! JNZ LOADERR ;END FILE IS 1
0706 CD6604                             CALL RESETDISK ;BACK TO ORIGINAL DISK
0709 CD5E0221F0                         CALL FILLFCB0! LXI H,SDISK! PUSH H
0710 7E32CD07                           MOV A,M! STA COMFCB ;DRIVE NUMBER SET
0714 3E10CD6002                         MVI A 16! CALL FILLFCB ;MOVE ENTIRE FCB
0719 E17E32DD07                         POP H! MOV A,M! STA COMFCB+16
071E AF32ED07                           XRA A! STA COMREC ;RECORD NUMBER SET TO
0722 115C0021CD                         LXI D,FCB! LXI H,COMFCB! MVI B,33! CALL
                                        ;MOVE COMMAND LINE TO BUFF
072D 210080                             LXI H,COMBUF
0730 7EB7CA3E07                 BMOVE0: MOV A,M! ORA A! JZ BMOVE1! CPI ' '! JZ B
073A 23C33007                           INX H! JMP BMOVE0 ;FOR ANOTHER SCAN
                                        ;FIRST BLANK POSITION FOUND
073E 0600118100                 BMOVE1: MVI B,0! LXI D,BUFF+1! ;READY FOR THE MO
0743 7E12B7CA4F                 BMOVE2: MOV A,M! STAX D! ORA A! JZ BMOVE3
                                        ;MORE TO MOVE
0749 042313C343                         INR B! INX H! INX D! JMP BMOVE2
                                BMOVE3: ;B HAS CHARACTER COUNT
074F 78328000                           MOV A,B! STA BUFF
0753 CD9B00                             CALL CRLF
                                        ;NOW GO TO THE LOADED PROGRAM
0756 CDD501                             CALL SETDMABUFF ;DEFAULT DMA
0759 CD1A01                             CALL SAVEUSER ;USER CODE SAVED
                                        ;LOW MEMORY DISKA CONTAINS USER CODE
075C CD0001                             CALL TRAN ;GONE TO THE LOADED PROGRAM
075F 31AB07                             LXI SP,STACK ;MAY COME BACK HERE
0762 CD2901CDBD                         CALL SETDISKA! CALL SELECT
```

```
CP/M MACRO ASSEM 2.0    #015    console command processor (CCP), ver 2.0

 0760 C38203                            JMP CCP
                                ;
                                USERER: ;ARRIVE HERE ON COMMAND ERROR
 0763 CD6604C309                        CALL RESETDISK! JMP COMERR
                                        ;
                                LOADERR:;CANNOT LOAD THE PROGRAM
 0771 017A07CDA7                        LXI B LOADMSG! CALL PRINT
 0777 C38607                            JMP RETCOM
 077A 424144204C                        LOADMSG: DB 'BAD LOAD',0
 0783 434F4D                    COMTYPE:        DB 'COM' ;FOR COM FILES
                ;
                ;
                RETCOM: ;RESET DISK BEFORE END OF COMMAND CHECK
 0786 CD6604            CALL RESETDISK
                ;
                ENDCOM: ;END OF INTRINSIC COMMAND
 0789 CD5102            CALL FILLFCB0 ;TO CHECK FOR GARBAGE AT END OF LINE
 078C 3ACE07D620        LDA COMFCB+1! SUI ' '! LXI H,SDISK! ORA M
                        ;0 IN ACCUMULATOR IF NO DISK SELECTED, AND BLANK FCB
 0795 C20902            JNZ COMERR
 0798 C38203            JMP CCP
                ;
                ;
                ;
                ;       DATA AREAS
 079B                   DS      16      ;8 LEVEL STACK
                STACK:
                ;
                ;       'SUBMIT' FILE CONTROL BLOCK
 07AB 00        SUBMIT: DB      0       ;00 IF NO SUBMIT FILE, FF IF SUBMITTING
 07AC 0024242420SUBFCB: DB      0,'$$$      '   ;FILE NAME IS $$$
 07B5 5355420000        DB      'SUB',0,0       ;FILE TYPE IS SUB
 07BA 00        SUBMOD: DB      0       ;MODULE NUMBER
 07BB           SUBRC:  DS      1       ;RECORD COUNT FILED
 07BC                   DS      16      ;DISK MAP
 07CC           SUBCR:  DS      1       ;CURRENT RECORD TO READ
                ;
                ;       COMMAND FILE CONTROL BLOCK
 07CD           COMFCB: DS      32      ;FIELDS FILLED IN LATER
 07ED           COMREC: DS      1       ;CURRENT RECORD TO READ/WRITE
 07EE           DCNT:   DS      1       ;DISK DIRECTORY COUNT (USED FOR ERROR CO
 07EF           CDISK:  DS      1       ;CURRENT DISK
 07F0           SDISK:  DS      1       ;SELECTED DISK FOR CURRENT OPERATION
                                        ;NONE=0, A=1  B=2 ...
 07F1           BPTR:   DS      1       ;BUFFER POINTER
 07F2                   END     CCPLOC
```

```
0259 ADDH       044B ADDHCF     03CF BADSERIAL  0005 BDOS       00F4 BDOSCOND
00C3 BDOSINR    0800 BDOSL      00A2 BLANK      0730 BMOVE0     073E BMOVE1
0743 BMOVE2     074F BMOVE3     07F1 BPTR       000B BREAKF     01C2 BREAKKEY
0080 BUFF       0398 CCP0       0358 CCPCLEAR   0382 CCP        0000 CCPLOC
035C CCPSTART   07EF CDISK      00DA CLOSE      0010 CLOSEF     0088 COMADDR
0008 COMBUF     0209 COMERR     020F COMERR0    0222 COMERR1    07CD COMFCB
0007 COMLEN     07FD COMREC     0783 COMTYPE    0408 CONV0      0433 CONV1
000D CR         0098 CRLF       01D0 CSELECT    0019 CSELF      07EE DCNT
024F DEBLANK    00FF DELETE     0013 DELF       0230 DELIM      01DD DELSUB
0488 DIR0       048F DIR1       0498 DIR2       04D9 DIR3       04F9 DIR4
050B DIR5       050F DIR6       04F7 DIRB       0477 DIRECT     04CC DIRHDR0
04D4 DIRHDR1    0004 DISKA      00F9 DISKREAD   00FE DISKREADC  0104 DISKWRITE
001A DMAF       0014 DREADF     0015 DWRITF     02F0 EFILL      02F2 EFILL0
0789 ENDCOM     051B ENDIR      001A EOFILE     051F ERASE      0542 ERASEFILE
0552 ERMSG      0000 FALSE      005C FCB        025E FILLFCB0   0260 FILLFCB
0607 FULLMSG    03F8 GETNUMBER  0113 GETUSER    000D INITF      00B8 INITIALIZE
0006 INTLEN     0333 INTRIN0    033C INTRIN1    034F INTRIN2    0354 INTRIN3
032E INTRINSIC  0310 INTVEC     03C1 JMPTAB     005F LA         000A LF
000C LIFTF      06E1 LOAD0      0701 LOAD1      0771 LOADERR    077A LOADMSG
0018 LOGF       0016 MAKEF      0109 MAKE       0006 MAXLEN     0442 MOVE0
0440 MOVENAME   03EA NOFILE     03F0 NOFMSG     01A7 NOREAD     0196 NOSUB
00D0 OPENC      000F OPENF      00CB OPEN       02B9 PADNAME    02E9 PADTY
0009 PBUFF      0002 PCHARF     00AC PRIN0      0092 PRINTBC    00A7 PRINT
008C PRINTCHAR  00EA RBUFF      0001 RCHARF     03DF RDMSG      0139 READCOM
01AB READCOM0   01BA READCOM1   03D9 READERR    063F REN1       0659 REN2
010E RENAM      0610 RENAME     066D RENERR1    0673 RENERR2    0679 RENERR3
0017 RENF       0682 RENMSG     0466 RESETDISK  0786 RETCOM     0601 RETSAVE
0009 ROFILE     05AD SAVE       05D4 SAVE0      05F1 SAVE1      05FB SAVERR
011A SAVEUSER   0301 SCNQ       0309 SCNQ0      07F0 SDISK      00DF SEARCH
00E9 SEARCHCOM  00E4 SEARCHN    0011 SEARF      0012 SEARNF     00BD SELECT
000E SELF       01FD SER0       01F5 SERIALIZE  0328 SERIAL     0289 SETCUR0
0288 SETCUR     0129 SETDISKA   0454 SETDISK   01D5 SETDMABUFF 01D8 SETDMA
0290 SETDSK     0298 SETNAM0    02A9 SETNAM1    02AB SETNAM2    0296 SETNAME
02C0 SETTY      02C8 SETTY0     02D9 SETTY1     02DB SETTY2     0115 SETUSER
07AB STACK      008A STADDR     07CC SUBCR      07AC SUBFCB     07AB SUBMIT
07BA SUBMOD     07BB SUBRC      000A SYSFILE    0000 TESTING    0100 TRAN
0000 TRANM      0130 TRANSLATE  02AF TRNAME     02DF TRTYP      FFFF TRUE
055D TYPE       0574 TYPE0      0587 TYPE1      05A0 TYPEOF     05A7 TYPERR
06C4 USER0      076B USERER     0020 USERF      068E USER       06A5 USERFUNC
```

```
                        TITLE   'Bdos Interface  Bdos Version 2.0 Aug, 1979'
                ;***************************************************************
                ;***************************************************************
                ;**
                ;**   B A S I C   D I S K   O P E R A T I N G   S Y S T E M
                ;**            I N T E R F A C E   M O D U L E
                ;**
                ;***************************************************************
                ;***************************************************************
                ;
                ;       COPYRIGHT (C) 1978, 1979
                ;       DIGITAL RESEARCH
                ;       BOX 579, PACIFIC GROVE
                ;       CALIFORNIA
                ;
FFFF =          ON      EQU     0FFFFH
0000 =          OFF     EQU     00000H
0000 =          TEST    EQU     OFF
                ;
                        IF      TEST
                        ORG     3C00H
                        ELSE
0800                    ORG     0800H
                        ENDIF
                ;       BIOS VALUE DEFINED AT END OF MODULE
                ;
0018 =          SSIZE   EQU     24              ;24 LEVEL STACK
                ;
                ;       LOW MEMORY LOCATIONS
0000 =          REBOOT  EQU     0000H           ;REBOOT SYSTEM
0003 =          IOLOC   EQU     0003H           ;I/O BYTE LOCATION
0006 =          BDOSA   EQU     0006H           ;ADDRESS FIELD OF JMP BDOS
                ;
                ;       BIOS ACCESS CONSTANTS
1600 #          BOOTF   SET     BIOS+3*0        ;COLD BOOT FUNCTION
1603 #          WBOOTF  SET     BIOS+3*1        ;WARM BOOT FUNCTION
1606 #          CONSTF  SET     BIOS+3*2        ;CONSOLE STATUS FUNCTION
1609 #          CONINF  SET     BIOS+3*3        ;CONSOLE INPUT FUNCTION
160C #          CONOUTF SET     BIOS+3*4        ;CONSOLE OUTPUT FUNCTION
160F #          LISTF   SET     BIOS+3*5        ;LIST OUTPUT FUNCTION
1612 #          PUNCHF  SET     BIOS+3*6        ;PUNCH OUTPUT FUNCTION
1615 #          READERF SET     BIOS+3*7        ;READER INPUT FUNCTION
1618 #          HOMEF   SET     BIOS+3*8        ;DISK HOME FUNCTION
161B #          SELDSKF SET     BIOS+3*9        ;SELECT DISK FUNCTION
161E #          SETTRKF SET     BIOS+3*10       ;SET TRACK FUNCTION
1621 #          SETSECF SET     BIOS+3*11       ;SET SECTOR FUNCTION
1624 #          SETDMAF SET     BIOS+3*12       ;SET DMA FUNCTION
1627 #          READF   SET     BIOS+3*13       ;READ DISK FUNCTION
162A #          WRITEF  SET     BIOS+3*14       ;WRITE DISK FUNCTION
162D #          LISTSTF SET     BIOS+3*15       ;LIST STATUS FUNCTION
1630 #          SECTRAN SET     BIOS+3*16       ;SECTOR TRANSLATE
                ;
                ;       EQUATES FOR NON GRAPHIC CHARACTERS
0003 =          CTLC    EQU     03H     ;CONTROL C
0005 =          CTLE    EQU     05H     ;PHYSICAL EOL
0008 =          CTLH    EQU     08H     ;BACKSPACE
```



```
0010 =          CTLP    EQU     10H     ;PRNT TOGGLE
0012 =          CTLR    EQU     12H     ;REPEAT LINE
0013 =          CTLS    EQU     13H     ;STOP/START SCREEN
0015 =          CTLU    EQU     15H     ;LINE DELETE
0018 =          CTLX    EQU     18H     ;=CTL U
001A =          CTLZ    EQU     1AH     ;END OF FILE
007F =          RUBOUT  EQU     7FH     ;CHAR DELETE
0009 =          TAB     EQU     09H     ;TAB CHAR
000D =          CR      EQU     0DH     ;CARRIAGE RETURN
000A =          LF      EQU     0AH     ;LINE FEED
005E =          CTL     EQU     5EH     ;UP ARROW
                ;
0800 0000000000         DB      0,0,0,0,0,0
                ;
                ;       ENTER HERE FROM THE USER'S PROGRAM WITH FUNCTION NUMBER
                ;       AND INFORMATION ADDRESS IN D,E
0806 C31108             JMP     BDOSE   ;PAST PARAMETER BLOCK
                ;
                ;       ************************************************
                ;       *** RELATIVE LOCATIONS 0009 - 000E           ***
                ;       ************************************************
                ;
0809 DB08       PERERR: DW      PERSUB  ;PERMANENT ERROR SUBROUTINE
080B E708       SELERR: DW      SELSUB  ;SELECT ERROR SUBROUTINE
080D ED08       RODERR: DW      RODSUB  ;RO DISK ERROR SUBROUTINE
080F F308       ROFERR: DW      ROFSUB  ;RO FILE ERROR SUBROUTINE
                ;
                ;
                BDOSE:  ;ARRIVE HERE FROM USER PROGRAMS
0811 EB22490B0EB        XCHG! SHLD INFO! XCHG ;INFO=DE, DE=INFO
0816 7B329315           MOV A,E! STA LINFO ;LINFO = LOW(INFO) - DON'T EQU
081A 210000224B         LXI H,0! SHLD ARET ;RETURN VALUE DEFAULTS TO 0000
                        ;SAVE USER'S STACK POINTER, SET TO LOCAL STACK
0820 3922150B           DAD SP! SHLD ENTSP ;ENTSP = STACKPTR
0824 31470B             LXI SP,LSTACK ;LOCAL STACK SETUP
0827 AF329D1532         XRA A! STA FCBDSK! STA RESEL ;FCBDSK,RESEL=FALSE
082E 214215             LXI H,GOBACK ;RETURN HERE AFTER ALL FUNCTIONS
0831 E5                 PUSH H ;JMP GOBACK EQUIVALENT TO RET
0832 79FE25D0           MOV A,C! CPI NFUNCS! RNC ;SKIP IF INVALID #
0836 4B                 MOV C,E ;POSSIBLE OUTPUT CHARACTER TO C
0837 2144085F16         LXI H,FUNCTAB! MOV E,A! MVI D,0 ;DE=FUNC, HL=.CIOTAB
083D 19195F2356         DAD D! DAD D! MOV E,M! INX H! MOV D,M ;DE=FUNCTAB(FUNC)
0842 EBE9               XCHG! PCHL ;DISPATCHED
                ;
                ;       DISPATCH TABLE FOR FUNCTIONS
                FUNCTAB:
0844 00168E0894         DW      BOOTF, FUNC1  FUNC2  FUNC3
084C 9E08A208A6         DW      FUNC4, FUNC5, FUNC6, FUNC7
0854 C208C708D0         DW      FUNC8  FUNC9  FUNC10 FUNC11
000C =          DISKF   EQU     ($-FUNCTAB)/2   ;DISK FUNCS
085C 5E1464147D         DW      FUNC12 FUNC13 FUNC14 FUNC15
0864 8914BF14A8         DW      FUNC16,FUNC17,FUNC18,FUNC19
086C C014C714CE         DW      FUNC20 FUNC21 FUNC22 FUNC23
0874 E014E714FE         DW      FUNC24,FUNC25,FUNC26,FUNC27
087C FE14011508         DW      FUNC28 FUNC29 FUNC30 FUNC31
0884 18152D1533         DW      FUNC32,FUNC33,FUNC34,FUNC35
088C 3F15               DW      FUNC36
```

```
0025 =          NFUNCS  EQU     ($-FUNCTAB)/2
                ;
                FUNC1:
                        ;RETURN CONSOLE CHARACTER WITH ECHO
088E CD4B09             CALL CONECH
0891 C3D708             JMP STA$RET
                ;
                FUNC2:
                        ;WRITE CONSOLE CHARACTER WITH TAB EXPANSION
0894 CDD509             CALL TABOUT
0897 C9                 RET ;JMP GOBACK
                ;
                FUNC3:
                        ;RETURN READER CHARACTER
089E CD1516             CALL READERF
089B C3D708             JMP STA$RET
                ;
                FUNC4:
                        ;WRITE PUNCH CHARACTER
089E CD1216             CALL PUNCHF
08A1 C9                 RET ;JMP GOBACK
                ;
                FUNC5:
                        ;WRITE LIST CHARACTER
                        ;WRITE TO LIST DEVICE
08A2 CD0F16             CALL LISTF
08A5 C9                 RET ;JMP GOBACK
                ;
                FUNC6:
                        ;DIRECT CONSOLE I/O - READ IF 0FFH
08A6 793CCAAF08         MOV A,C! INR A! JZ DIRINP ;0FFH => 00H, MEANS INPUT MODE
                                ;DIRECT OUTPUT FUNCTION
08AB CD0C16                     CALL CONOUTF
08AE C9                         RET ;JMP GOBACK
                        DIRINP:
08AF CD0616                     CALL CONSTF ;STATUS CHECK
08B2 B7CA5F15                   ORA A! JZ RETMON ;SKIP, RETURN 00 IF NOT READY
                                ;CHARACTER IS READY  GET IT
08B6 CD0916                     CALL CONINF ;TO A
08B9 C3D708                     JMP STA$RET
                ;
                FUNC7:
                        ;RETURN IO BYTE
08BC 3A0300             LDA IOLOC
08BF C3D708             JMP STA$RET
                ;
                FUNC8:
                        ;SET I/O BYTE
08C2 210300             LXI H,IOLOC
08C5 71                 MOV M,C
08C6 C9                 RET ;JMP GOBACK
                ;
                FUNC9:
                        ;WRITE LINE UNTIL $ ENCOUNTERED
08C7 2A490B4D44         LHLD INFO! MOV C,L! MOV B,H ;BC=STRING ADDRESS
08CC CD1E0A             CALL PRINT ;OUT TO CONSOLE
```



```
08CF C9                 RET ;JMP GOBACK
                ;
                FUNC10:
                        ;READ A BUFFERED CONSOLE LINE
08D0 CD290A             CALL READ
08D3 C9                 RET ;JMP GOBACK
                ;
                FUNC11:
                        ;CHECK CONSOLE STATUS
08D4 CD6809             CALL CONBRK
                        ;(DROP THROUGH TO STA$RET)
                STA$RET:
                        ;STORE THE A REGISTER TO ARET
08D7 324B0BC9           STA ARET! RET ;JMP GOBACK
                ;
                ;       ERROR SUBROUTINES
                PERSUB: ;REPORT PERMANENT ERROR
08DB 212509CDFC         LXI H,PERMSG! CALL ERRFLG ;TO REPORT THE ERROR
08E1 FE03CA0000         CPI CTLC! JZ REBOOT ;REBOOT IF RESPONSE IS CTLC
08E6 C9                 RET ;AND IGNORE THE ERROR
                ;
                SELSUB: ;REPORT SELECT ERROR
08E7 213009C3F6         LXI H,SELMSG! JMP WAIT$ERR ;WAIT CONSOLE BEFORE BOOT
                ;
                RODSUB: ;REPORT WRITE TO READ/ONLY DISK
08ED 213C09C3F6         LXI H,RODMSG! JMP WAIT$ERR ;WAIT CONSOLE
                ;
                ROFSUB: ;REPORT READ/ONLY FILE
08F3 213709             LXI H,ROFMSG ;DROP THROUGH TO WAIT FOR CONSOLE
                ;
                WAIT$ERR:
                        ;WAIT FOR RESPONSE BEFORE BOOT
08F6 CDFC08C300         CALL ERRFLG! JMP REBOOT
                ;
                ERRFLG:
                        ;REPORT ERROR TO CONSOLE  MESSAGE ADDRESS IN HL
08FC E5CD110A           PUSH H! CALL CRLF ;STACK MSSG ADDRESS, NEW LINE
0900 3A480BC641         LDA CURDSK! ADI 'A'! STA DSKERR ;CURRENT DISK NAME
0908 011509CD1B         LXI B,DSKMSG! CALL PRINT ;THE ERROR MESSAGE
090E C1CD1B0A           POP B! CALL PRINT ;ERROR MESSAGE TAIL
0912 C34009             JMP CONIN ;TO GET THE INPUT CHARACTER
                        ;RET
                ;
                ;       ERROR MESSAGES
0915 42646F7320DSKMSG:  DB      'Bdos Err On '
0921 203A2024   DSKERR: DB      ': $'   ;FILLED IN BY ERRFLG
0925 4261642053PERMSG:  DB      'Bad Sector$'
0930 53656C6563SELMSG:  DB      'Select$'
0937 46696C6520ROFMSG:  DB      'File '
093C 522F4F24   RODMSG: DB      'R/O$'
                ;
                ;
                ;       CONSOLE HANDLERS
                CONIN:
                        ;READ CONSOLE CHARACTER TO A
0940 21140B7E36         LXI H,KBCHAR! MOV A,M! MVI M,0! ORA A! RNZ
```

```
                        ;NO PREVIOUS KEYBOARD CHARACTER READY
0948 C30916             JMP CONINF ;GET CHARACTER EXTERNALLY
                        ;RET
                ;
                CONECH:
                        ;READ CHARACTER WITH ECHO
091B CD4009CD59         CALL CONIN! CALL ECHOC! RC ;ECHO CHARACTER?
                        ;CHARACTER MUST BE ECHOED BEFORE RETURN
0952 F54FCDD509         PUSH PSW! MOV C,A! CALL TABOUT! POP PSW
0958 C9                 RET ;WITH CHARACTER IN A
                ;
                ECHOC:
                        ;ECHO CHARACTER IF GRAPHIC
                        ;CR  LF, TAB, OR BACKSPACE
0959 FE0DC8             CPI CR! RZ ;CARRIAGE RETURN?
095C FE0AC8             CPI LF! RZ ;LINE FEED?
095F FE09C8             CPI TAB! RZ ;TAB?
0962 FE08C8             CPI CTLH! RZ ;BACKSPACE?
0965 FE20C9             CPI ' '! RET ;CARRY SET IF NOT GRAPHIC
                ;
                CONBRK: ;CHECK FOR CHARACTER READY
0968 3A140BB7C2         LDA KBCHAR! ORA A! JNZ CONB1 ;SKIP IF ACTIVE KBCHAR
                                ;NO ACTIVE KBCHAR, CHECK EXTERNAL BREAK
096F CD0616E601                 CALL CONSTF! ANI 1! RZ ;RETURN IF NO CHAR READY
                                ;CHARACTER READY, READ IT
0975 CD0916                     CALL CONINF ;TO A
0978 FE13C28709                 CPI CTLS! JNZ CONB0 ;CHECK STOP SCREEN FUNCTION
                                ;FOUND CTLS  READ NEXT CHARACTER
097D CD0916                     CALL CONINF ;TO A
0980 FE03CA0000                 CPI CTLC! JZ REBOOT ;CTLC IMPLIES RE-BOOT
                                ;NOT A REBOOT, ACT AS IF NOTHING HAS HAPPENED
0985 AFC9                       XRA A! RET ;WITH ZERO IN ACCUMULATOR
                        CONB0:
                                ;CHARACTER IN ACCUM, SAVE IT
0987 32140B                     STA KBCHAR
                        CONB1:
                                ;RETURN WITH TRUE SET IN ACCUMULATOR
098A 3E01C9                     MVI A,1! RET
                ;
                CONOUT:
                        ;COMPUTE CHARACTER POSITION/WRITE CONSOLE CHAR FROM C
                        ;COMPCOL = TRUE IF COMPUTING COLUMN POSITION
098D 3A100BB7C2         LDA COMPCOL! ORA A! JNZ COMPOUT
                                ;WRITE THE CHARACTER  THEN COMPUTE THE COLUMN
                                ;WRITE CONSOLE CHARACTER FROM C
0994 C5CD6809                   PUSH B! CALL CONBRK ;CHECK FOR SCREEN STOP FUNCT
0998 C1C5                       POP B! PUSH B ;RECALL/SAVE CHARACTER
099A CD0C16                     CALL CONOUTF ;EXTERNALLY  TO CONSOLE
099D C1C5                       POP B! PUSH B ;RECALL/SAVE CHARACTER
                                ;MAY BE COPYING TO THE LIST DEVICE
099F 3A130BB7C4                 LDA LISTCP! ORA A! CNZ LISTF ;TO PRINTER, IF SO
09A6 C1                         POP B ;RECALL THE CHARACTER
                        COMPOUT:
09A7 79                         MOV A,C ;RECALL THE CHARACTER
                                ;AND COMPUTE COLUMN POSITION
09A8 21120B                     LXI H,COLUMN ;A = CHAR, HL = .COLUMN
```



```
09AB FE7F              CPI RUBOUT! RZ ;NO COLUMN CHANGE IF NULLS
09AE 34                INR M ;COLUMN = COLUMN + 1
09AF FE20D0            CPI ' '! RNC ;RETURN IF GRAPHIC
                       ;NOT GRAPHIC, RESET COLUMN POSITION
09B2 35                DCR M ;COLUMN = COLUMN - 1
09B3 7EB7C8            MOV A,M! ORA A! RZ ;RETURN IF AT ZERO
                       ;NOT AT ZERO, MAY BE BACKSPACE OR END LINE
09B6 79                MOV A,C ;CHARACTER BACK TO A
09B7 FE08C2BE09        CPI CTLH! JNZ NOTBACKSP
                               ;BACKSPACE CHARACTER
                               DCR M ;COLUMN = COLUMN - 1
09BC 35                        RET
09BD C9                NOTBACKSP:
                               ;NOT A BACKSPACE CHARACTER, EOL?
09BE FE0AC0                    CPI LF! RNZ ;RETURN IF NOT
                               ;END OF LINE, COLUMN = 0
09C1 3600                      MVI M,0 ;COLUMN = 0
09C3 C9                RET
                ;
                CTLOUT:
                        ;SEND C CHARACTER WITH POSSIBLE PRECEDING UP-ARROW
09C4 79CD5909           MOV A,C! CALL ECHOC ;CY IF NOT GRAPHIC (OR SPECIAL CASE
09C8 D2D509             JNC TABOUT ;SKIP IF GRAPHIC  TAB, CR, LF, OR CTLH
                               ;SEND PRECEDING UP ARROW
09CB F50E5ECD8D                PUSH PSW! MVI C,CTL! CALL CONOUT ;UP ARROW
09D1 F1F640                    POP PSW! ORI 40H ;BECOMES GRAPHIC LETTER
09D4 4F                        MOV C,A ;READY TO PRINT
                               ;(DROP THROUGH TO TABOUT)
                ;
                TABOUT:
                        ;EXPAND TABS TO CONSOLE
09D5 79FE09C28D         MOV A,C! CPI TAB! JNZ CONOUT ;DIRECT TO CONOUT IF NOT
                               ;TAB ENCOUNTERED, MOVE TO NEXT TAB POSITION
                        TAB0:
09DB 0E20CD8D09                MVI C,' '! CALL CONOUT ;ANOTHER BLANK
09E0 3A120BE607                LDA COLUMN! ANI 111B ;COLUMN MOD 8 = 0 ?
09E5 C2DB09                    JNZ TAB0 ;BACK FOR ANOTHER IF NOT
09E8 C9                 RET
                ;
                PCTLH:
                        ;SEND CTLH TO CONSOLE WITHOUT AFFECTING COLUMN COUNT
09E9 0E08C30C16         MVI C,CTLH! JMP CONOUTF
                        ;RET
                ;
                BACKUP:
                        ;BACK-UP ONE SCREEN POSITION
09EE CDE9090E20         CALL PCTLH! MVI C,' '! CALL CONOUTF! JMP PCTLH
                        ;
                CRLFP:
                        ;PRINT #, CR, LF FOR CTLX, CTLU, CTLR FUNCTIONS
                        ;THEN MOVE TO STRTCOL (STARTING COLUMN)
09F9 0E23CD8D09         MVI C,'#'! CALL CONOUT
09FE CD110A             CALL CRLF
                        ;COLUMN = 0, MOVE TO POSITION STRTCOL
                        CRLFP0:
0A01 3A120B2111                LDA COLUMN! LXI H,STRTCOL
```

```
CP/M MACRO ASSEM 2.0    #007    Bdos Interface, Bdos, Version 2.0 Aug, 1979

0A07 BED0                       CMP M! RNC ;STOP WHEN COLUMN REACHES STRTCOL
0A09 0E20CD8D09                 MVI C,' '! CALL CONOUT ;PRINT BLANK
0A0E C3010A                     JMP CRLFP0
                ;;
                ;
                CRLF:
                        ;CARRIAGE RETURN LINE FEED SEQUENCE
0A11 0E0DCD8D09         MVI C,CR! CALL CONOUT! MVI C,LF! JMP CONOUT
                        ;RET
                ;
                PRINT:
                        ;PRINT MESSAGE UNTIL M(BC) = '$'
0A1B 0AFE24C8           LDAX B! CPI '$'! RZ ;STOP ON $
                                ;MORE TO PRINT
0A1F 03C54F                     INX B! PUSH B! MOV C,A ;CHAR TO C
0A22 CDD509                     CALL TABOUT ;ANOTHER CHARACTER PRINTED
0A25 C1C31B0A                   POP B! JMP PRINT
                ;
                READ:   ;READ TO INFO ADDRESS (MAX LENGTH, CURRENT LENGTH, BUFFE
0A29 3A120B3211         LDA COLUMN! STA STRTCOL ;SAVE START FOR CTL-X, CTL H
0A2F 2A490B4E23         LHLD INFO! MOV C,M! INX H! PUSH H! MVI B,0
                        ;B = CURRENT BUFFER LENGTH,
                        ;C = MAXIMUM BUFFER LENGTH,
                        ;HL= NEXT TO FILL - 1
                        READNX:
                                ;READ NEXT CHARACTER, BC, HL ACTIVE
0A37 C5E5                       PUSH B! PUSH H ;BLEN, CMAX, HL SAVED
                                READN0:
0A39 CD4009                             CALL CONIN ;NEXT CHAR IN A
0A3C E67F                               ANI 7FH ;MASK PARITY BIT
0A3E E1C1                               POP H! POP B ;REACTIVATE COUNTERS
0A40 FE0DCA090B                         CPI CR! JZ READEN ;END OF LINE?
0A45 FE0ACA090B                         CPI LF! JZ READEN ;ALSO END OF LINE
0A4A FE08C25E0A                         CPI CTLH! JNZ NOTH ;BACKSPACE?
                                        ;DO WE HAVE ANY CHARACTERS TO BACK OVER?
0A4F 78B7CA370A                         MOV A,B! ORA A! JZ READNX
                                        ;CHARACTERS REMAIN IN BUFFER  BACKUP ONE
0A54 05                                 DCR B ;REMOVE ONE CHARACTER
0A55 3A120B3210                         LDA COLUMN! STA COMPCOL ;COL > 0
                                        ;COMPCOL > 0 MARKS REPEAT AS LENGTH COMP
0A5B C3B80A                             JMP LINELEN ;USES SAME CODE AS REPEAT
                                NOTH:
                                        ;NOT A BACKSPACE
0A5E FE7FC26E0A                         CPI RUBOUT! JNZ NOTRUB ;RUBOUT CHAR?
                                        ;RUBOUT ENCOUNTERED  RUBOUT IF POSSIBLE
0A63 78B7CA370A                         MOV A,B! ORA A! JZ READNX ;SKIP IF LEN=0
                                        ;BUFFER HAS CHARACTERS  RESEND LAST CHAR
0A68 7E2B                               MOV A,M! DCR B! DCX H ;A = LAST CHAR
                                        ;BLEN BLEN-1, NEXT TO FILL = 1 DECREMENT
0A6B C3F10A                             JMP RDECHO ;ACT LIKE THIS IS AN ECHO
                ;
                                NOTRUB:
                                        ;NOT A RUBOUT CHARACTER, CHECK END LINE
0A6E FE05C27F0A                         CPI CTLE! JNZ NOTE ;PHYSICAL END LINE?
                                        ;YES, SAVE ACTIVE COUNTERS AND FORCE EOL
0A73 C5E5CD110A                         PUSH B! PUSH H! CALL CRLF
```

```
CP/M MACRO ASSEM 2.0    #008    Bdos Interface, Bdos  Version 2.0 Aug, 1979

0A78 AF32110B                   XRA A! STA STRTCOL ;START POSITION = 00
0A7C C3390A                     JMP READN0 ;FOR ANOTHER CHARACTER
                        NOTE:
                                ;NOT END OF LINE, LIST TOGGLE?
0A7F FE10C2900A                 CPI CTLP! JNZ NOTP ;SKIP IF NOT CTLP
                                ;LIST TOGGLE = CHANGE PARITY
0A84 E5                         PUSH H ;SAVE NEXT TO FILL - 1
0A85 21130B                     LXI H,LISTCP ;HL=.LISTCP FLAG
0A88 3E0196                     MVI A,1! SUB M ;TRUE-LISTCP
0A8B 77                         MOV M,A ;LISTCP = NOT LISTCP
0A8C E1C3370A                   POP H! JMP READNX ;FOR ANOTHER CHAR
                        NOTP:
                                ;NOT A CTLP  LINE DELETE?
0A90 FE18C2A70A                 CPI CTLX! JNZ NOTX
0A95 E1                         POP H ;DISCARD START POSITION
                                ;LOOP WHILE COLUMN > STRTCOL
                                BACKX:
0A96 3A110B2112                         LDA STRTCOL! LXI H,COLUMN
0A9C BED2290A                           CMP M! JNC READ ;START AGAIN
0AA0 35                                 DCR M ;COLUMN = COLUMN - 1
0AA1 CDEE09                             CALL BACKUP ;ONE POSITION
0AA4 C3960A                             JMP BACKX
                        NOTX:
                                ;NOT A CONTROL X, CONTROL U?
                                ;NOT CONTROL-X, CONTROL-U?
0AA7 FE15C2B30A                 CPI CTLU! JNZ NOTU ;SKIP IF NOT
                                ;DELETE LINE (CTLU)
0AAC CDF909                     CALL CRLFP ;PHYSICAL EOL
0AAF E1                         POP H ;DISCARD STARTING POSITION
0AB0 C3290A                     JMP READ ;TO START ALL OVER
                        NOTU:
                                ;NOT LINE DELETE, REPEAT LINE?
0AB3 FE12C2EE0A                 CPI CTLR! JNZ NOTR
                        LINELEN:
                                ;REPEAT LINE, OR COMPUTE LINE LEN (CTLH)
                                ;IF COMPCOL > 0
0AB8 C5CDF909                   PUSH B! CALL CRLFP ;SAVE LINE LENGTH
0ABC C1E1E5C5                   POP B! POP H! PUSH H! PUSH B
                                ;BCUR  CMAX ACTIVE  BEGINNING BUFF AT HL
                        REP0:
0AC0 78B7CAD20A                 MOV A,B! ORA A! JZ REP1 ;COUNT LEN TO 00
0AC5 234E                       INX H! MOV C,M ;NEXT TO PRINT
0AC7 05C5E5                     DCR B! PUSH B! PUSH H ;COUNT LENGTH DOWN
0ACA CDC409                     CALL CTLOUT ;CHARACTER ECHOED
0ACD E1C1                       POP H! POP B ;RECALL REMAINING COUNT
0ACF C3C00A                     JMP REP0 ;FOR THE NEXT CHARACTER
                        REP1:
                                ;END OF REPEAT, RECALL LENGTHS
                                ;ORIGINAL BC STILL REMAINS PUSHED
0AD2 E5                         PUSH H ;SAVE NEXT TO FILL
0AD3 3A1[illegible]B87          LDA COMPCOL! ORA A ;=0 IF COMPUTING LENG
0AD7 CA390A                     JZ READN0 ;FOR ANOTHER CHAR IF SO
                                ;COLUMN POSITION COMPUTED FOR CTLH
0ADA 21120B96                   LXI H,COLUMN! SUB M ;DIFF > 0
0ADE 32100B                     STA COMPCOL ;COUNT DOWN BELOW
                                ;MOVE BACK COMPCOL-COLUMN SPACES
```

```
                        BACKSP:
                                ;MOVE BACK ONE MORE SPACE
0AE1 CDEE09                     CALL BACKUP ;ONE SPACE
0AE4 21100B35                   LXI H COMPCOL! DCR M
0AE8 C2E10A                     JNZ BACKSP
0AEB C3390A                     JMP READN0 ;FOR NEXT CHARACTER
                        NOTR:
                                ;NOT A CTLR, PLACE INTO BUFFER
                        RDECHO:
0AEE 2377                       INX H! MOV M A ;CHARACTER FILLED TO MEM
0AF0 04                         INR B ;BLEN = BLEN + 1
                        RDECH1:
                                ;LOOK FOR A RANDOM CONTROL CHARACTER
0AF1 C5E5                       PUSH B! PUSH H ;ACTIVE VALUES SAVED
0AF3 4F                         MOV C A ;READY TO PRINT
0AF4 CDC409                     CALL CTLOUT ;MAY BE UP-ARROW C
0AF7 E1C17E                     POP H! POP B! MOV A,M ;RECALL CHAR
0AFA FE03                       CPI CTLC ;SET FLAGS FOR REBOOT TEST
0AFC 78                         MOV A B ;MOVE LENGTH TO A
0AFD C2050B                     JNZ NOTC ;SKIP IF NOT A CONTROL C
0B00 FE01                       CPI 1 ;CONTROL C, MUST BE LENGTH 1
0B02 CA0000                     JZ REBOOT ;REBOOT IF BLEN = 1
                                ;LENGTH NOT ONE  SO SKIP REBOOT
                        NOTC:
                                ;NOT REBOOT  ARE WE AT END OF BUFFER?
0B05 B9DA370A                   CMP C! JC READNX ;GO FOR ANOTHER IF NOT
                        READEN:
                                ;END OF READ OPERATION, STORE BLEN
0B09 E170                       POP H! MOV M,B ;M=CURRENT LEN) = B
0B0B 0E0DC38D09                 MVI C CR! JMP CONOUT ;RETURN CARRIAGE
                                ;RET
                ;
                ;
                ;       DATA AREAS
                ;
0B10 00         COMPCOL:DB      0       ;TRUE IF COMPUTING COLUMN POSITION
0B11 00         STRTCOL:DB      0       ;STARTING COLUMN POSITION AFTER READ
0B12 00         COLUMN: DB      0       ;COLUMN POSITION
0B13 00         LISTCP: DB      0       ;LISTING TOGGLE
0B14 00         KBCHAR: DB      0       ;INITIAL KEY CHAR = 00
0B15            ENTSP:  DS      2       ;ENTRY STACK POINTER
0B17                    DS      SSIZE*2 ;STACK SIZE
                LSTACK:
                ;       END OF BASIC I/O SYSTEM
                ;
                ;***************************************************************
                ;***************************************************************
                ;
                ;       COMMON VALUES SHARED BETWEEN BDOSI AND BDOS
0B47 00         USRCODE:DB      0       ;CURRENT USER NUMBER
0B48 00         CURDSK: DB      0       ;CURRENT DISK NUMBER
0B49            INFO:   DS      2       ;INFORMATION ADDRESS
0B4B            ARET:   DS      2       ;ADDRESS VALUE TO RETURN
0B4B =          LRET    EQU     ARET    ;LOW(ARET)
                ;
                ;***************************************************************
```



```
                ;***************************************************************
                ;**
                ;**   B A S I C   D I S K   O P E R A T I N G   S Y S T E M
                ;**
                ;***************************************************************
                ;***************************************************************
                ;
0020 =          DVERS   EQU     20H     ;VERSION 2.0
                ;       MODULE ADDRESSES
                ;
                ;       LITERAL CONSTANTS
00FF =          TRUE    EQU     0FFH    ;CONSTANT TRUE
0000 =          FALSE   EQU     000H    ;CONSTANT FALSE
FFFF =          ENDDIR  EQU     0FFFFH  ;END OF DIRECTORY
0001 =          BYTE    EQU     1       ;NUMBER OF BYTES FOR "BYTE" TYPE
0002 =          WORD    EQU     2       ;NUMBER OF BYTES FOR "WORD" TYPE
                ;
                ;       FIXED ADDRESSES IN LOW MEMORY
005C =          TFCB    EQU     005CH   ;DEFAULT FCB LOCATION
0080 =          TBUFF   EQU     0080H   ;DEFAULT BUFFER LOCATION
                ;
                ;       FIXED ADDRESSES REFERENCED IN BIOS MODULE ARE
                ;       PERERR (0009), SELERR (000C), RODERR (000E)
                ;
                ;       ERROR MESSAGE HANDLERS
                GOERR:
                        ;HL = .ERRORHANDLER, CALL SUBROUTINE
0B4D 5E2356             MOV E,M! INX H! MOV D,M ;ADDRESS OF ROUTINE IN DE
0B50 EBE9               XCHG! PCHL ;TO SUBROUTINE
                ;
                PER$ERROR:
                        ;REPORT PERMANENT ERROR TO USER
0B52 21090EC34D         LXI H,PERERR! JMP GOERR
                ;
                SEL$ERROR:
                        ;REPORT SELECT ERROR
0B58 210B0EC34D         LXI H,SELERR! JMP GOERR
                ;
                ROD$ERROR:
                        ;REPORT READ/ONLY DISK ERROR
0B5F 210D0EC34D         LXI H,RODERR! JMP GOERR
                ;
                ROF$ERROR:
                        ;REPORT READ/ONLY FILE ERROR
0B64 210F0EC34D         LXI H,ROFERR! JMP GOERR
                ;
                ;
                ;       LOCAL SUBROUTINES FOR BIOS INTERFACE
                ;
                MOVE:
                        ;MOVE DATA LENGTH OF LENGTH C FROM SOURCE DE TO
                        ;DESTINATION GIVEN BY HL
0B6A 0C                 INR C ;IN CASE IT IS ZERO
                        MOVE0:
0B6B 0DC8                       DCR C! RZ ;MORE TO MOVE
0B6D 1A77                       LDAX D! MOV M A ;ONE BYTE MOVED
```

```
0B6F 1323                       INX D! INX H ;TO NEXT BYTE
0B71 C36B0B                     JMP MOVE0
                ;
                SELECTDISK:
                        ;SELECT THE DISK DRIVE GIVEN BY CURDSK  AND FILL
                        ;THE BASE ADDRESSES CURTRKA - ALLOCA, THEN FILL
                        ;THE VALUES OF THE DISK PARAMETER BLOCK
0B74 3A4E0B4F           LDA CURDSK! MOV C,A ;CURRENT DISK# TO C
                        ;LSB OF E = 0 IF NOT YET LOGGED - IN
0B78 CD1B1E             CALL SELDSKF ;HL FILLED BY CALL
                        ;HL = 0000 IF ERROR, OTHERWISE DISK HEADERS
0B7B 7CB5C8             MOV A,H! ORA L! RZ ;RETURN WITH 0000 IN HL AND Z FLAG
                        ;DISK HEADER BLOCK ADDRESS IN HL
0B7F 5E235623           MOV E,M! INX H! MOV D,M! INX H ;DE=.TRAN
0B82 2270152323         SHLD CDRMAXA! INX H! INX H ;.CDRMAX
0B87 2272152323         SHLD CURTRKA! INX H! INX H ;HL=.CURREC
0B8C 2274152323         SHLD CURRECA! INX H! INX H ;HL=.BUFFA
                        ;DE STILL CONTAINS .TRAN
0B91 EB22ED15           XCHG! SHLD TRANV ;.TRAN VECTOR
0B95 217615             LXI H,BUFFA ;DE= SOURCE FOR MOVE, HL=DEST
0B98 0E0ECD6A0B         MVI C,ADDLIST! CALL MOVE ;ADDLIST FILLED
                        ;NOW FILL THE DISK PARAMETER BLOCK
0B9D 2A7E15EB           LHLD DPBADDR! XCHG ;DE IS SOURCE
0BA1 217E15             LXI H SECTPT ;HL IS DESTINATION
0BA4 0E0FCD6A0B         MVI C,DPBLIST! CALL MOVE ;DATA FILLED
                        ;NOW SET SINGLE-DOUBLE MAP MODE
0BA9 2A8315             LHLD MAXALL ;LARGEST ALLOCATION NUMBER
0BAC 7C                 MOV A,H ;00 INDICATES < 255
0BAD 219A1536FF         LXI H,SINGLE! MVI M,TRUE ;ASSUME A-00
0BB2 B7CABB0B           ORA A! JZ RETSELECT
                        ;HIGH ORDER OF MAXALL NOT ZERO, USE DOUBLE DM
0BB6 3600               MVI M,FALSE
                RETSELECT:
0BB8 3EFFB7C9           MVI A,TRUE! ORA A! RET ;SELECT DISK FUNCTION OK
                ;
                HOME:
                        ;MOVE TO HOME POSITION, THEN OFFSET TO START OF DIR
0BBC CD1816             CALL HOMEF ;MOVE TO TRACK 00, SECTOR 00 REFERENCE
0BBF 218B154E23         LXI H,OFFSET! MOV C,M! INX H! MOV B,M! CALL SETTRKF
                        ;FIRST DIRECTORY POSITION SELECTED
0BC8 AF                 XRA A ;CONSTANT ZERO TO ACCUMULATOR
0BC9 2A72157723         LHLD CURTRKA! MOV M,A! INX H! MOV M,A ;CURTRK=0000
0BCF 2A74157723         LHLD CURRECA! MOV M,A! INX H! MOV M,A ;CURREC=0000
                        ;CURTRK, CURREC BOTH SET TO 0000
0BD5 C9                 RET
                ;
                RDBUFF:
                        ;READ BUFFER AND CHECK CONDITION
0BD6 CD2716             CALL READF ;CURRENT DRIVE, TRACK, SECTOR, DMA
0BD9 B7C2520B           ORA A! JNZ PERRORR
0BDD C9                 RET
                ;
                WRBUFF:
                        ;WRITE BUFFER AND CHECK CONDITION
                        ;WRITE TYPE (WRTYPE) IS IN REGISTER C
                        ;WRTYPE = 0 => NORMAL WRITE OPERATION
```



```
                        ;WRTYPE = 1 => DIRECTORY WRITE OPERATION
                        ;WRTYPE = 2 => START OF NEW BLOCK
0BDF CD2A16             CALL WRITEF ;CURRENT DRIVE, TRACK, SECTOR, DMA
0BE2 B7C2520B           ORA A! JNZ PERRORR
0BE6 C9                 RET
                ;
                SEEK:
                ;SEEK THE TRACK GIVEN BY ARECORD (ACTUAL RECORD)
                ;LOCAL EQUATES FOR REGISTERS
0001 =          ARECH   EQU B! ARECL   EQU C ;ARECORD = BC
0003 =          CRECH   EQU D! CRECL   EQU E ;CURREC  = DE
0005 =          CTRKH   EQU H! CTRKL   EQU L ;CURTRK  = HL
0005 =          TCRECH EQU H! TCRECL EQU L ;TCURREC = HL
                ;LOAD THE REGISTERS FROM MEMORY
0BE6 21A2154E23 LXI H,ARECORD! MOV ARECL,M! INX H! MOV ARECH,M
0BEC 2A74155E23 LHLD CURRECA ! MOV CRECL M! INX H! MOV CRECH M
0BF2 2A72157E23 LHLD CURTRKA ! MOV A,M! INX H! MOV CTRKH,M! MOV CTRKL,A
                ;LOOP WHILE ARECORD < CURREC
                SEEK0:
0BF9 7993789A           MOV A,ARECL! SUB CRECL! MOV A ARECH! SBB CRECH
0BFD D20F0C             JNC SEEK1 ;SKIP IF ARECORD >= CURREC
                                ;CURREC = CURREC - SECTPT
0C00 E52A7E15                   PUSH CTRKH! LHLD SECTPT
0C04 7B955F                     MOV A,CRECL! SUB L! MOV CRECL A
0C07 7A9C57                     MOV A,CRECH! SBB H! MOV CRECH,A
0C0A E1                         POP CTRKH
                                ;CURTRK = CURTRK - 1
0C0B 2B                         DCX CTRKH
0C0C C3F90E             JMP SEEK0 ;FOR ANOTHER TRY
                SEEK1:
                ;LOOK WHILE ARECORD >= (T:=CURREC + SECTPT)
0C0F E5                 PUSH CTRKH
0C10 2A7E1519           LHLD SECTPT! DAD CRECH ;HL = CURREC+SECTPT
0C14 7995789C           MOV A ARECL! SUB TCRECL! MOV A ARECH! SBB TCRECH
0C18 DA210C             JC SEEK2 ;SKIP IF T > ARECORD
                                ;CURREC = T
0C1B EB                         XCHG
                                ;CURTRK = CURTRK + 1
0C1C E123                       POP CTRKH! INX CTRKH
0C1E C30F0C             JMP SEEK1 ;FOR ANOTHER TRY
0C21 E1                 SEEK2:  POP CTRKH
                ;ARRIVE HERE WITH UPDATED VALUES IN EACH REGISTER
0C22 C5D5E5             PUSH ARECH! PUSH CRECH! PUSH CTRKH ;TO STACK FOR LATER
                ;STACK CONTAINS (LOWEST) BC=ARECORD  DE=CURREC  HL=CURTR
0C25 EB2A8B1519         XCHG! LHLD OFFSET! DAD D ;HL = CURTRK+OFFSET
0C2A 444DCD1E16         MOV B,H! MOV C L! CALL SETTRKF ;TRACK SET UP
                ;NOTE THAT BC = CURTRK IS DIFFERENCE TO MOVE IN BIOS
0C2F D1                 POP D ;RECALL CURTRK
0C30 2A72157323         LHLD CURTRKA! MOV M,E! INX H! MOV M D ;CURTRK UPDATED
                ;NOW COMPUTE SECTOR AS ARECORD-CURREC
0C36 D1                 POP CRECH ;RECALL CURREC
0C37 2A74157323         LHLD CURRECA! MOV M,CRECL! INX H! MOV M,CRECH
0C3D C1                 POP ARECH ;BC-ARECORD  DE-CURREC
0C3E 79934F             MOV A,ARECL! SUB CRECL! MOV ARECL,A
0C41 789A47             MOV A ARECH! SBB CRECH! MOV ARECH A
0C44 2A8D15EB           LHLD TRANV! XCHG ;BC-SECTOR#, DE=.TRAN
```

```
0C48 CD3016             CALL SECTRAN ;HL = TRAN(SECTOR)
0C4B 4D44               MOV C,L! MOV B,H ;BC = TRAN(SECTOR)
0C4D C32116             JMP SETSECF ;SECTOR SELECTED
                        ;RET
                ;
                ;       FILE CONTROL BLOCK (FCB) CONSTANTS
00E5 =          EMPTY   EQU     0E5H    ;EMPTY DIRECTORY ENTRY
007F =          LSTREC  EQU     127     ;LAST RECORD# IN EXTENT
0080 =          RECSIZ  EQU     128     ;RECORD SIZE
0020 =          FCBLEN  EQU     32      ;FILE CONTROL BLOCK SIZE
0004 =          DIRREC  EQU     RECSIZ/FCBLEN   ;DIRECTORY ELTS / RECORD
0002 =          DSKSHF  EQU     2       ;LOG2(DIRREC)
0003 =          DSKMSK  EQU     DIRREC-1
0005 =          FCBSHF  EQU     5       ;LOG2(FCBLEN)
                ;
000C =          EXTNUM  EQU     12      ;EXTENT NUMBER FIELD
001F =          MAXEXT  EQU     31      ;LARGEST EXTENT NUMBER
000D =          UBYTES  EQU     13      ;UNFILLED BYTES FIELD
000E =          MODNUM  EQU     14      ;DATA MODULE NUMBER
000F =          MAXMOD  EQU     15      ;LARGEST MODULE NUMBER
0080 =          FWFMSK  EQU     80H     ;FILE WRITE FLAG IS HIGH ORDER MODNUM
000F =          NAMLEN  EQU     15      ;NAME LENGTH
000F =          RECCNT  EQU     15      ;RECORD COUNT FIELD
0010 =          DSKMAP  EQU     16      ;DISK MAP FIELD
001F =          LSTFCB  EQU     FCBLEN-1
0020 =          NXTREC  EQU     FCBLEN
0021 =          RANREC  EQU     NXTREC+1;RANDOM RECORD FIELD (2 BYTES)
                ;
                ;       RESERVED FILE INDICATORS
0009 =          ROFILE  EQU     9       ;HIGH ORDER OF FIRST TYPE CHAR
000A =          INVIS   EQU     10      ;INVISIBLE FILE IN DIR COMMAND
                ;       EQU     11      ;RESERVED
                ;
                ;       UTILITY FUNCTIONS FOR FILE ACCESS
                ;
                DM$POSITION:
                        ;COMPUTE DISK MAP POSITION FOR VRECORD TO HL
0C50 2180154E           LXI H,BLKSHF! MOV C M ;SHIFT COUNT TO C
0C54 3AA015             LDA VRECORD ;CURRENT VIRTUAL RECORD TO A
                        DMPOS0:
0C57 B71F0DC257                 ORA A! RAR! DCR C! JNZ DMPOS0
                        ;A = SHR(VRECORD,BLKSHF) = VRECORD/2**(SECT/BLOCK)
0C5D 47                 MOV B,A ;SAVE IT FOR LATER ADDITION
0C5E 3E0896             MVI A,8! SUB M ;8-BLKSHF TO ACCUMULATOR
0C61 4F                 MOV C,A ;EXTENT SHIFT COUNT IN REGISTER C
0C62 3A9F15             LDA EXTVAL ;EXTENT VALUE ANI EXTMSK
                        DMPOS1:
                                ;BLKSHF = 3,4,5,6,7, C=5,4,3,2,1
                                ;SHIFT IS 4,3,2,1,0
0C65 0DCA6E0C                   DCR C! JZ DMPOS2
0C69 B717C3650C                 ORA A! RAL! JMP DMPOS1
                        DMPOS2:
                        ;ARRIVE HERE WITH A = SHL(EXT AND EXTMSK,7-BLKSHF)
0C6E 80                 ADD B ;ADD THE PREVIOUS SHR(VRECORD,BLKSHF) VALUE
                        ;A IS ONE OF THE FOLLOWING VALUES, DEPENDING UPON ALLOC
                        ;BKS BLKSHF
```



```
                        ;1K    3       V/8 + EXTVAL * 16
                        ;2K    4       V/16+ EXTVAL * 8
                        ;4K    5       V/32+ EXTVAL * 4
                        ;8K    6       V/64+ EXTVAL * 2
                        ;16K   7       V/128+EXTVAL * 1
0C6F C9                 RET ;WITH DM$POSITION IN A
                ;
                GETDM:
                        ;RETURN DISK MAP VALUE FROM POSITION GIVEN BY BC
0C70 2A490B             LHLD INFO ;BASE ADDRESS OF FILE CONTROL BLOCK
0C73 11100019           LXI D,DSKMAP! DAD D ;HL =.DISKMAP
0C77 09                 DAD B ;INDEX BY A SINGLE BYTE VALUE
0C78 3A9A15             LDA SINGLE ;SINGLE BYTE/MAP ENTRY?
0C7B B7CA830C           ORA A! JZ GETDMD ;GET DISK MAP SINGLE BYTE
0C7F 6E2600C9                   MOV L,M! MVI H,0! RET ;WITH HL=00BB
                        GETDMD:
0C83 09                         DAD B ;HL=.FCB(DM+I*2)
                                ;DOUBLE PRECISION VALUE RETURNED
0C84 5E2356EBC9                 MOV E,M! INX H! MOV D,M! XCHG! RET
                ;
                INDEX:
                        ;COMPUTE DISK BLOCK NUMBER FROM CURRENT FCB
0C89 CD500C             CALL DM$POSITION ;0...15 IN REGISTER A
0C8C 4F0600CD70         MOV C,A! MVI B,0! CALL GETDM ;VALUE TO HL
0C92 22A215C9           SHLD ARECORD! RET
                ;
                ALLOCATED:
                        ;CALLED FOLLOWING INDEX TO SEE IF BLOCK ALLOCATED
0C96 2AA2157DB4         LHLD ARECORD! MOV A,L! ORA H! RET
                ;
                ATRAN:
                        ;COMPUTE ACTUAL RECORD ADDRESS, ASSUMING INDEX CALLED
0C9C 3A8015             LDA BLKSHF ;SHIFT COUNT TO REG A
0C9F 2AA215             LHLD ARECORD
                        ATRAN0:
0CA2 293DC2A20C                 DAD H! DCR A! JNZ ATRAN0 ;SHL(ARECORD,BLKSHF)
0CA7 3A8115AF           LDA BLKMSK! MOV C,A ;MASK VALUE TO C
0CAB 3AA015A1           LDA VRECORD! ANA C ;MASKED VALUE IN A
0CAF B56F               ORA L! MOV L,A ;TO HL
0CB1 22A215             SHLD ARECORD ;ARECORD-HL OR (VRECORD AND BLKMSK)
0CB4 C9                 RET
                ;
                GETEXTA:
                        ;GET CURRENT EXTENT FIELD ADDRESS TO A
0CB5 2A490B110C         LHLD INFO! LXI D,EXTNUM! DAD D ;HL=.FCB(EXTNUM)
0CBC C9                 RET
                ;
                GETFCBA:
                        ;COMPUTE RECCNT AND NXTREC ADDRESSES FOR GET/SETFCB
0CBD 2A490B110F         LHLD INFO! LXI D,RECCNT! DAD D! XCHG ;DE=.FCB(RECCNT)
0CC5 21110019           LXI H,(NXTREC-RECCNT)! DAD D ;HL=.FCB(NXTREC)
0CC9 C9                 RET
                ;
                GETFCB:
                        ;SET VARIABLES FROM CURRENTLY ADDRESSED FCB
0CCA CDBD0C             CALL GETFCBA ;ADDRESSES IN DE, HL
```

```
0C48 CD3016             CALL SECTRAN ;HL = TRAN(SECTOR)
0C4B 4D44               MOV C,L! MOV B,H ;BC = TRAN(SECTOR)
0C4D C32116             JMP SETSEC# ;SECTOR SELECTED
                        ;RET
                ;
                ;       FILE CONTROL BLOCK (FCB) CONSTANTS
00E5 =          EMPTY   EQU     0E5H    ;EMPTY DIRECTORY ENTRY
007F =          LSTREC  EQU     127     ;LAST RECORD# IN EXTENT
0080 =          RECSIZ  EQU     128     ;RECORD SIZE
0020 =          FCBLEN  EQU     32      ;FILE CONTROL BLOCK SIZE
0004 =          DIRREC  EQU     RECSIZ/FCBLEN   ;DIRECTORY ELTS / RECORD
0002 =          DSKSHF  EQU     2       ;LOG2(DIRREC)
0003 =          DSKMSK  EQU     DIRREC-1
0005 =          FCBSHF  EQU     5       ;LOG2(FCBLEN)
                ;
000C =          EXTNUM  EQU     12      ;EXTENT NUMBER FIELD
001F =          MAXEXT  EQU     31      ;LARGEST EXTENT NUMBER
000D =          UBYTES  EQU     13      ;UNFILLED BYTES FIELD
000E =          MODNUM  EQU     14      ;DATA MODULE NUMBER
000F =          MAXMOD  EQU     15      ;LARGEST MODULE NUMBER
0080 =          FWFMSK  EQU     80H     ;FILE WRITE FLAG IS HIGH ORDER MODNUM
000F =          NAMLEN  EQU     15      ;NAME LENGTH
000F =          RECCNT  EQU     15      ;RECORD COUNT FIELD
0010 =          DSKMAP  EQU     16      ;DISK MAP FIELD
001F =          LSTFCB  EQU     FCBLEN-1
0020 =          NXTREC  EQU     FCBLEN
0021 =          RANREC  EQU     NXTREC+1;RANDOM RECORD FIELD (2 BYTES)
                ;
                ;       RESERVED FILE INDICATORS
0009 =          ROFILE  EQU     9       ;HIGH ORDER OF FIRST TYPE CHAR
000A =          INVIS   EQU     10      ;INVISIBLE FILE IN DIR COMMAND
                ;       EQU     11      ;RESERVED
                ;
                ;       UTILITY FUNCTIONS FOR FILE ACCESS
                ;
                DM$POSITION:
                        ;COMPUTE DISK MAP POSITION FOR VRECORD TO HL
0C50 2180154E           LXI H,BLKSHF! MOV C,M ;SHIFT COUNT TO C
0C54 3AA015             LDA VRECORD ;CURRENT VIRTUAL RECORD TO A
                        DMPOS0:
0C57 B71F0DC257                 ORA A! RAR! DCR C! JNZ DMPOS0
                        ;A = SHR(VRECORD,BLKSHF) = VRECORD/2**(SECT/BLOCK)
0C5D 47                 MOV B,A ;SAVE IT FOR LATER ADDITION
0C5E 3E0896             MVI A,8! SUB M ;8-BLKSHF TO ACCUMULATOR
0C61 4F                 MOV C,A ;EXTENT SHIFT COUNT IN REGISTER C
0C62 3A9F15             LDA EXTVAL ;EXTENT VALUE AND EXTMSK
                        DMPOS1:
                                ;BLKSHF = 3,4,5,6,7, C=5,4,3,2,1
                                ;SHIFT IS 4,3,2,1,0
0C65 0DCA6E0C                   DCR C! JZ DMPOS2
0C69 B717C3650C                 ORA A! RAL! JMP DMPOS1
                        DMPOS2:
                        ;ARRIVE HERE WITH A = SHL(EXT AND EXTMSK,7-BLKSHF)
0C6E 80                 ADD B ;ADD THE PREVIOUS SHR(VRECORD,BLKSHF) VALUE
                        ;A IS ONE OF THE FOLLOWING VALUES, DEPENDING UPON ALLOC
                        ;BKS BLKSHF
```



```
                        ;1K   3       V/8 + EXTVAL * 16
                        ;2K   4       V/16+ EXTVAL * 8
                        ;4K   5       V/32+ EXTVAL * 4
                        ;8K   6       V/64+ EXTVAL * 2
                        ;16K  7       V/128+EXTVAL * 1
0C6F C9                 RET ;WITH DM$POSITION IN A
                ;
                GETDM:
                        ;RETURN DISK MAP VALUE FROM POSITION GIVEN BY BC
0C70 2A490B             LHLD INFO ;BASE ADDRESS OF FILE CONTROL BLOCK
0C73 11100019           LXI D,DSKMAP! DAD D ;HL = .DISKMAP
0C77 09                 DAD B ;INDEX BY A SINGLE BYTE VALUE
0C78 3A9A15             LDA SINGLE ;SINGLE BYTE/MAP ENTRY?
0C7B B7CA830C           ORA A! JZ GETDMD ;GET DISK MAP SINGLE BYTE
0C7F 6F2600C9                   MOV L,M! MVI H,0! RET ;WITH HL = 00BB
                        GETDMD:
0C83 09                         DAD B ;HL = .FCB(DM+I*2)
                                ;DOUBLE PRECISION VALUE RETURNED
0C84 5E2356EBC9                 MOV E,M! INX H! MOV D,M! XCHG! RET
                ;
                INDEX:
                        ;COMPUTE DISK BLOCK NUMBER FROM CURRENT FCB
0C89 CD500C             CALL DM$POSITION ;0...15 IN REGISTER A
0C8C 4F0600CD70         MOV C,A! MVI B,0! CALL GETDM ;VALUE TO HL
0C92 22A215C9           SHLD ARECORD! RET
                ;
                ALLOCATED:
                        ;CALLED FOLLOWING INDEX TO SEE IF BLOCK ALLOCATED
0C96 2AA2157DB4         LHLD ARECORD! MOV A,L! ORA H! RET
                ;
                ATRAN:
                        ;COMPUTE ACTUAL RECORD ADDRESS, ASSUMING INDEX CALLED
0C9C 3A8015             LDA BLKSHF ;SHIFT COUNT TO REG A
0C9F 2AA215             LHLD ARECORD
                        ATRAN0:
0CA2 293DC2A20C                 DAD H! DCR A! JNZ ATRAN0 ;SHL(ARECORD,BLKSHF)
0CA7 3A81154F           LDA BLKMSK! MOV C,A ;MASK VALUE TO C
0CAB 3AA015A1           LDA VRECORD! ANA C ;MASKED VALUE IN A
0CAF B56F               ORA L! MOV L,A ;TO HL
0CB1 22A215             SHLD ARECORD ;ARECORD=HL OR (VRECORD AND BLKMSK)
0CB4 C9                 RET
                ;
                GETEXTA:
                        ;GET CURRENT EXTENT FIELD ADDRESS TO A
0CB5 2A490B110C         LHLD INFO! LXI D,EXTNUM! DAD D ;HL = .FCB(EXTNUM)
0CBC C9                 RET
                ;
                GETFCBA:
                        ;COMPUTE RECCNT AND NXTREC ADDRESSES FOR GET/SETFCB
0CBD 2A490B110F         LHLD INFO! LXI D,RECCNT! DAD D! XCHG ;DE = .FCB(RECCNT)
0CC5 21110019           LXI H,(NXTREC-RECCNT)! DAD D ;HL = .FCB(NXTREC)
0CC9 C9                 RET
                ;
                GETFCB:
                        ;SET VARIABLES FROM CURRENTLY ADDRESSED FCB
0CCA CDBD0C             CALL GETFCBA ;ADDRESSES IN DE, HL
```

```
CP/M MACRO ASSEM 2.0     #015    Bdos Interface, Bdos, Version 2.0 Aug, 1979

0CCE 7E32A015           MOV A,M! STA VRECORD ;VRECORD=FCB(NXTREC)
0CD1 EB7E329E15         XCHG! MOV A M! STA RCOUNT ;RCOUNT=FCB(RECCNT)
0CD6 CDB50C             CALL GETEXTA ;HL = .FCB(EXTNUM)
0CD9 3A8215             LDA EXTMSK ;EXTENT MASK TO A
0CDC A6                 ANA M ;FCB(EXTNUM) AND EXTMSK
0CDD 329F15             STA EXTVAL
0CE0 C9                 RET
                ;
                SETFCB:
                        ;PLACE VALUES BACK INTO CURRENT FCB(SEQ,REC #)
0CE1 CDBD0C             CALL GETFCBA ;ADDRESSES TO DE, HL
0CE4 3A92154F           LDA SEQIO! MOV C,A ;=1 IF SEQUENTIAL I/O
0CE8 3AA0159177         LDA VRECORD! ADD C! MOV M,A ;FCB(NXTREC)=VRECORD+SEQIO
0CED EB3A9E1577         XCHG! LDA RCOUNT! MOV M A ;FCB(RECCNT)=RCOUNT
0CF2 C9                 RET
                ;
                HLROTR:
                        ;HL ROTATE RIGHT BY AMOUNT C
0CF3 0C                 INR C ;IN CASE ZERO
0CF4 0DC8               HLROTR0: DCR C! RZ ;RETURN WHEN ZERO
0CF6 7CB71F67                   MOV A,H! ORA A! RAR! MOV H,A ;HIGH BYTE
0CFA 7D1F6F                     MOV A,L! RAR! MOV L A ;LOW BYTE
0CFD C3F40C                     JMP HLROTR0
                ;
                SEEKDIR:
                        ;SEEK THE RECORD CONTAINING THE CURRENT DIR ENTRY
0D00 2AA515             LHLD DCNT ;DIRECTORY COUNTER TO HL
0D03 0E02CDF30C         MVI C,DSKSHF! CALL HLROTR ;VALUE TO HL
0D08 22A21522A7         SHLD ARECORD! SHLD DREC ;READY FOR SEEK
0D0F C3E60B             JMP SEEK
                        ;RET
                ;
                COMPUTE$CS:
                        ;COMPUTE CHECKSUM FOR CURRENT DIRECTORY BUFFER
0D11 0E80               MVI C,RECSIZ ;SIZE OF DIRECTORY BUFFER
0D13 2A7615             LHLD BUFFA ;CURRENT DIRECTORY BUFFER
0D16 AF                 XRA A ;CLEAR CHECKSUM VALUE
                        COMPUTECS0:
0D17 86230D                     ADD M! INX H! DCR C ;CS=CS+BUFF(RECSIZ-C)
0D1A C2170D                     JNZ COMPUTECS0
0D1D C9                 RET ;WITH CHECKSUM IN A
                ;
                HLROTL:
                        ;ROTATE THE MASK IN HL BY AMOUNT IN C
0D1E 0C                 INR C ;MAY BE ZERO
0D1F 0DC8               HLROTL0: DCR C! RZ ;RETURN IF ZERO
0D21 29C31F0D                   DAD H! JMP HLROTL0
                ;
                SET$CDISK:
                        ;SET A "1" VALUE IN CURDSK POSITION OF BC
0D25 C5                 PUSH B ;SAVE INPUT PARAMETER
0D26 3A480B4F           LDA CURDSK! MOV C,A ;READY PARAMETER FOR SHIFT
0D2A 210100             LXI H,1 ;NUMBER TO SHIFT
0D2D CD1E0D             CALL HLROTL ;HL = MASK TO INTEGRATE
0D30 C1                 POP B ;ORIGINAL MASK
0D31 79B56F             MOV A,C! ORA L! MOV L A
```



```
CP/M MACRO ASSEM 2.0     #016    Bdos Interface  Bdos  Version 2.0 Aug. 1979

0D34 78B467             MOV A,B! ORA H! MOV H,A ;HL = MASK OR ROL(1,CURDSK)
0D37 C9                 RET
                ;
                NOWRITE:
                        ;RETURN TRUE IF DIR CHECKSUM DIFFERENCE OCCURRED
0D38 2A6A153A48         LHLD RODSK! LDA CURDSK! MOV C,A! CALL HLROTR
0D42 7DE601C9           MOV A,L! ANI 1B! RET ;NON ZERO IF NOWRITE
                ;
                SET$RO:
                        ;SET CURRENT DISK TO READ ONLY
0D46 216A154E23         LXI H,RODSK! MOV C,M! INX H! MOV B,M
0D4C CD250D             CALL SET$CDISK ;SETS BIT TO 1
0D4F 226A15             SHLD RODSK
                        ;HIGH WATER MARK IN DIRECTORY GOES TO MAX.
0D52 2A8515EB           LHLD DIRMAX! XCHG ;DE = DIRECTORY MAX
0D56 2A7015             LHLD CDRMAXA ;HL = .CDRMAX
0D59 732372             MOV M,E! INX H! MOV M,D ;CDRMAX = DIRMAX
0D5C C9                 RET
                ;
                CHECK$ROFILE:
                        ;CHECK CURRENT BUFF(DPTR) OR FCB(0) FOR R/O STATUS
0D5D 11090019           LXI D,ROFILE! DAD D ;OFFSET TO RO BIT
0D61 7E17D0             MOV A,M! RAL! RNC ;RETURN IF NOT SET
0D64 C3640B             JMP ROF$ERROR ;EXIT TO READ ONLY DISK MESSAGE
                ;
                CHECK$RODIR:
                        ;CHECK CURRENT DIRECTORY ELEMENT FOR READ/ONLY STATUS
0D67 CD790D             CALL GETDPTRA ;ADDRESS OF ELEMENT
0D6A C35D0D             JMP CHECK$ROFILE ;SHARE CODE
                ;
                CHECK$WRITE:
                        ;CHECK FOR WRITE PROTECTED DISK
0D6D CD3B0DC8           CALL NOWRITE! RZ ;OK TO WRITE IF NOT RODSK
0D71 C35E0B             JMP ROD$ERROR ;READ ONLY DISK ERROR
                ;
                ADDH:
                        ;HL = HL + A
0D74 856FD0             ADD L! MOV L A! RNC
                        ;OVERFLOW TO H
0D77 24C9               INR H! RET
                ;
                GETDPTRA:
                        ;COMPUTE THE ADDRESS OF A DIRECTORY ELEMENT AT
                        ;POSITON DPTR IN THE BUFFER
0D79 2A76153AA4         LHLD BUFFA! LDA DPTR! JMP ADDH
                ;
                GETMODNUM:
                        ;COMPUTE THE ADDRESS OF THE MODULE NUMBER
                        ;BRING MODULE NUMBER TO ACCUMULATOR
                        ;(HIGH ORDER BIT IS FWF (FILE WRITE FLAG)
0D82 2A490B110E         LHLD INFO! LXI D,MODNUM! DAD D ;HL=.FCB(MODNUM)
0D89 7EC9               MOV A,M! RET ;A=FCB(MODNUM)
                ;
                CLRMODNUM:
                        ;CLEAR THE MODULE NUMBER FIELD FOR USER OPEN MAKE
0D8B CD820D3600         CALL GETMODNUM! MVI M 0 ;FCB(MODNUM)=0
```

```
0D90 C9                 RET
                ;
                SETFWF:
0D91 CD820D             CALL GETMODNUM ;HL=.FCB(MODNUM), A=FCB(MODNUM)
                        ;SET FWF (FILE WRITE FLAG) TO "1"
0D94 F68077             ORI FWFMSK! MOV M,A ;FCB(MODNUM) = FCB(MODNUM) OR 80H
                        ;ALSO RETURNS NON ZERO IN ACCUMULATOR
0D97 C9                 RET
                ;
                SETLRET1:
                        ;SET LRET = 1
0D98 3E01324B0B         MVI A,1! STA LRET1 RET
                ;
                COMPCDR:
                        ;RETURN CY IF CDRMAX > DCNT
0D9E 2AA515EB           LHLD DCNT! XCHG ;DE = DIRECTORY COUNTER
0DA2 2A7015             LHLD CDRMAXA ;HL=.CDRMAX
0DA5 7B96               MOV A,E! SUB M ;LOW(DCNT) - LOW(CDRMAX)
0DA7 23                 INX H ;HL = .CDRMAX+1
0DA8 7A9E               MOV A,D! SBB M ;HIG(DCNT) - HIG(CDRMAX)
                        ;CONDITION DCNT - CDRMAX  PRODUCES CY IF CDRMAX DCNT
0DAA C9                 RET
                ;
                SETCDR:
                        ;IF NOT (CDRMAX > DCNT) THEN CDRMAX = DCNT+1
0DAB CD9E0D             CALL COMPCDR
0DAE D8                 RC ;RETURN IF CDRMAX  DCNT
                        ;OTHERWISE  HL = .CDRMAX+1, DE = DCNT
0DAF 13722B73           INX D! MOV M,D! DCX H! MOV M,E
0DB3 C9                 RET
                ;
                SUBDH:
                        ;COMPUTE HL = DE - HL
0DB4 7B956F7A9C         MOV A,E! SUB L! MOV L,A! MOV A,D! SBB H! MOV H,A
0DBA C9                 RET
                ;
                NEWCHECKSUM:
0DBB 0EFF               MVI C,TRUE ;DROP THROUGH TO COMPUTE NEW CHECKSUM
                CHECKSUM:
                        ;COMPUTE CURRENT CHECKSUM RECORD AND UPDATE THE
                        ;DIRECTORY ELEMENT IF C=TRUE, OR CHECK FOR = IF NOT
                        ;DREC < CHKSIZ?
0DBD 2AA715EB2A         LHLD DREC! XCHG! LHLD CHKSIZ! CALL SUBDH ;DE-HL
0DC7 D0                 RNC ;SKIP CHECKSUM IF PAST CHECKSUM VECTOR SIZE
                                ;DREC < CHKSIZ, SO CONTINUE
0DC8 C5                         PUSH B ;SAVE INIT FLAG
0DC9 CD110D                     CALL COMPUTE$CS ;CHECK SUM VALUE TO A
0DCC 2A7A15                     LHLD CHECKA ;ADDRESS OF CHECK SUM VECTOR
0DCF EB                         XCHG
0DD0 2AA715                     LHLD DREC ;VALUE OF DREC
0DD3 19                         DAD D ;HL = .CHECK(DREC)
0DD4 C1                         POP B ;RECALL TRUE=0FFH OR FALSE=00 TO C
0DD5 0C                         INR C ;0FFH PRODUCES ZERO FLAG
0DD6 CAE30D                     JZ INITIAL$CS
                                        ;NOT INITIALIZING, COMPARE
0DD9 BE                                 CMP M ;COMPUTE$CS=CHECK(DREC)?
```



```
0DDA C8                                 RZ ;NO MESSAGE IF OK
                                        ;CHECKSUM ERROR, ARE WE BEYOND
                                        ;THE END OF THE DISK?
0DDB CD9E0D                             CALL COMPCDR
0DDE D0                                 RNC ;NO MESSAGE IF SO
0DDF CD460D                             CALL SET$RO ;READ/ONLY DISK SET
0DE2 C9                                 RET
                                INITIAL$CS:
                                        ;INITIALIZING THE CHECKSUM
0DE3 77C9                               MOV M,A! RET
                ;
                SETDMA:
                        ;HL .DMA ADDRESS TO SET (I.E., BUFFA OR DMAAD)
0DE5 4E2346             MOV C,M! INX H! MOV B,M ;PARAMETER READY
0DE8 C32416             JMP SETDMAF
                ;
                SETDATA:
                        ;SET DATA DMA ADDRESS
0DEB 216F15C3E5         LXI H,DMAAD! JMP SETDMA ;TO COMPLETE THE CALL
                ;
                SETDIR:
                        ;SET DIRECTORY DMA ADDRESS
0DF1 217615C3E5         LXI H,BUFFA! JMP SETDMA ;TO COMPLETE THE CALL
                ;
                WRDIR:
                        ;WRITE THE CURRENT DIRECTORY ENTRY, SET CHECKSUM
0DF7 CDBB0D             CALL NEWCHECKSUM ;INITIALIZE ENTRY
0DFA CDF10D             CALL SETDIR ;DIRECTORY DMA
0DFD 0E01               MVI C,1 ;INDICATES A WRITE DIRECTORY OPERATION
0DFF CDDE0B             CALL WRBUFF ;WRITE THE BUFFER
0E02 C3EB0D             JMP SETDATA ;TO DATA DMA ADDRESS
                        ;RET
                ;
                RD$DIR:
                        ;READ A DIRECTORY ENTRY INTO THE DIRECTORY BUFFER
0E05 CDF10D             CALL SETDIR ;DIRECTORY DMA
0E08 CDD60B             CALL RDBUFF ;DIRECTORY RECORD LOADED
0E0B C3EB0D             JMP SETDATA ;TO DATA DMA ADDRESS
                        ;RET
                ;
                DIR$TO$USER:
                        ;COPY THE DIRECTORY ENTRY TO THE USER BUFFER
                        ;AFTER CALL TO SEARCH OR SEARCHN BY USER CODE
0E0E 2A7615EB           LHLD BUFFA! XCHG ;SOURCE IS DIRECTORY BUFFER
0E12 2A6E15             LHLD DMAAD ;DESTINATION IS USER DMA ADDRESS
0E15 0E80               MVI C,RECSIZ ;COPY ENTIRE RECORD
0E17 C36A0B             JMP MOVE
                        ;RET
                ;
                END$OF$DIR:
                        ;RETURN ZERO FLAG IF AT END OF DIRECTORY, NON ZERO
                        ;IF NOT AT END (END OF DIR IF DCNT = 0FFFFH)
0E1A 21A5157E           LXI H,DCNT! MOV A,M ;MAY BE 0FFH
0E1E 23BE               INX H! CMP M ;LOW(DCNT) = HIGH(DCNT)?
0E20 C0                 RNZ ;NON ZERO RETURNED IF DIFFERENT
                        ;HIGH AND LOW THE SAME = 0FFH?
```

```
0E21 3C                 INR A ;0FFH BECOMES 00 IF SO
0E22 C9                 RET
                ;
                SET$END$DIR:
                        ;SET DCNT TO THE END OF THE DIRECTORY
0E23 21FFFF22A5         LXI H,ENDDIR! SHLD DCNT! RET
                ;
                READ$DIR:
                        ;READ NEXT DIRECTORY ENTRY, WITH C=TRUE IF INITIALIZING
0E2A 2A8515EB           LHLD DIRMAX! XCHG ;IN PREPARATION FOR SUBTRACT
0E2E 2AA5152322         LHLD DCNT! INX H! SHLD DCNT ;DCNT=DCNT+1
                        ;CONTINUE WHILE DIRMAX >= DCNT (DIRMAX-DCNT NO CY)
0E35 CDB40D             CALL SUBDH ;DE-HL
0E38 D23F0E             JNC READ$DIR0
                                ;YES, SET DCNT TO END OF DIRECTORY
0E3B CD230E                     CALL SET$END$DIR
0E3E C9                         RET
                        READ$DIR0:
                                ;NOT AT END OF DIRECTORY, SEEK NEXT ELEMENT
                                ;INITIALIZATION FLAG IS IN C
0E3F 3AA515E603                 LDA DCNT! ANI DSKMSK ;LOW(DCNT) AND DSKMSK
0E44 0605                       MVI B,FCBSHF ;TO MULTIPLY BY FCB SIZE
                                READ$DIR1:
0E46 8705C2460E                         ADD A! DCR B! JNZ READ$DIR1
                                ;A = (LOW(DCNT) AND DSKMSK) SHL FCBSHF
0E4B 32A415                     STA DPTR ;READY FOR NEXT DIR OPERATION
0E4E B7C0                       ORA A! RNZ ;RETURN IF NOT A NEW RECORD
0E50 C5                         PUSH B ;SAVE INITIALIZATION FLAG C
0E51 CD000D                     CALL SEEK$DIR ;SEEK PROPER RECORD
0E54 CD050E                     CALL RD$DIR ;READ THE DIRECTORY RECORD
0E57 C1                         POP B ;RECALL INITIALIZATION FLAG
0E58 C3BD0D                     JMP CHECKSUM ;CHECKSUM THE DIRECTORY ELT
                                ;RET
                ;
                ROTR:
                        ;BYTE VALUE FROM ALLOC IS IN REGISTER A, WITH SHIFT COUN
                        ;IN REGISTER C (TO PLACE BIT BACK INTO POSITION), AND
                        ;TARGET ALLOC POSITION IN REGISTERS HL.  ROTATE AND REPLA
0E5B 0F15C25B0E         RRC! DCR D! JNZ ROTR ;BACK INTO POSITION
0E60 77                 MOV M,A ;BACK TO ALLOC
0E61 C9                 RET
                ;
                GETALLOCBIT:
                        ;GIVEN ALLOCATION VECTOR POSITION BC.  RETURN WITH BYTE
                        ;CONTAINING BC SHIFTED SO THAT THE LEAST SIGNIFICANT
                        ;BIT IS IN THE LOW ORDER ACCUMULATOR POSITION.  HL IS
                        ;THE ADDRESS OF THE BYTE FOR POSSIBLE REPLACEMENT IN
                        ;MEMORY UPON RETURN, AND D CONTAINS THE NUMBER OF SHIFTS
                        ;REQUIRED TO PLACE THE RETURNED VALUE BACK INTO POSITION
0E62 79E6073C5F         MOV A,C! ANI 111B! INR A! MOV E,A! MOV D,A
                        ;D AND E BOTH CONTAIN THE NUMBER OF BIT POSITIONS TO SHI
0E68 790F0F0FE6         MOV A,C! RRC! RRC! RRC! ANI 11111B! MOV C,A ;C SHR 3 TO
0E6F 7887878787         MOV A,B! ADD A! ADD A! ADD A! ADD A! ADD A ;B SHL 5
0E75 B14F               ORA C! MOV C,A ;BBBCCCCC TO C
0E77 780F0F0FE6         MOV A,B! RRC! RRC! RRC! ANI 11111B! MOV B,A ;BC SHR 3 TO
0E7F 2A7C15             LHLD ALLOCA ;BASE ADDRESS OF ALLOCATION VECTOR
```



```
0E81 097E               DAD B! MOV A,M ;BYTE TO A, HL = .ALLOC(BC SHR 3)
                        ;NOW MOVE THE BIT TO THE LOW ORDER POSITION OF A
0E83 071DC2830E         ROTL: RLC! DCR E! JNZ ROTL! RET
                ;
                ;
                SETALLOCBIT:
                        ;BC IS THE BIT POSITION OF ALLOC TO SET OR RESET.  THE
                        ;VALUE OF THE BIT IS IN REGISTER E.
0E89 D5CD620E           PUSH D! CALL GETALLOCBIT ;SHIFTED VAL A, COUNT IN D
0E8D E6FE               ANI 1111$1110B ;MASK LOW BIT TO ZERO  MAY BE SET)
0E8F C1B1               POP B! ORA C ;LOW BIT OF C IS MASKED INTO A
0E91 C35B0E             JMP ROTR ;TO ROTATE BACK INTO PROPER POSITION
                        ;RET
                ;
                SCANDM:
                        ;SCAN THE DISK MAP ADDRESSED BY DPTR FOR NON-ZERO
                        ;ENTRIES, THE ALLOCATION VECTOR ENTRY CORRESPONDING
                        ;TO A NON-ZERO ENTRY IS SET TO THE VALUE OF C (0,1)
0E94 CD790D             CALL GETDPTRA ;HL = BUFFA + DPTR
                        ;HL ADDRESSES THE BEGINNING OF THE DIRECTORY ENTRY
0E97 11100019           LXI D,DSKMAP! DAD D ;HL NOW ADDRESSES THE DISK MAP
0E9B C5                 PUSH B ;SAVE THE 0/1 BIT TO SET
0E9C 0E11               MVI C,FCBLEN-DSKMAP+1 ;SIZE OF SINGLE BYTE DISK MAP + 1
                        SCANDM0:
                                ;LOOP ONCE FOR EACH DISK MAP ENTRY
0E9E D1                         POP D ;RECALL BIT PARITY
0E9F 0DC8                       DCR C! RZ ;ALL DONE SCANNING?
                                ;NO, GET NEXT ENTRY FOR SCAN
0EA1 D5                         PUSH D ;REPLACE BIT PARITY
0EA2 3A9A15B7CA                 LDA SINGLE! ORA A! JZ SCANDM1
                                                ;SINGLE BYTE SCAN OPERATION
0EA9 C5                                         PUSH B ;SAVE COUNTER
0EAA E5                                         PUSH H ;SAVE MAP ADDRESS
0EAB 4E0600                                     MOV C,M! MVI B,0 ;BC=BLOCK#
0EAF C3B70E                                     JMP SCANDM2
                                                SCANDM1:
                                                ;DOUBLE BYTE SCAN OPERATION
0EB1 0D                                         DCR C ;COUNT FOR DOUBLE BYTE
0EB2 C5                                         PUSH B ;SAVE COUNTER
0EB3 4E2346                                     MOV C,M! INX H! MOV B,M ;BC=BLOCK#
0EB6 E5                                         PUSH H ;SAVE MAP ADDRESS
                                                SCANDM2:
                                                ;ARRIVE HERE WITH BC=BLOCK#, E=0/1
0EB7 79B0                                       MOV A,C! ORA B ;SKIP IF = 0000
0EB9 C4890E                                     CNZ SET$ALLOC$BIT
                                                ;BIT SET TO 0/1
0EBC E123                                       POP H! INX H ;TO NEXT BIT POSITION
0EBE C1                                         POP B ;RECALL COUNTER
0EBF C39E0E                                     JMP SCANDM0 ;FOR ANOTHER ITEM
                ;
                INITIALIZE:
                        ;INITIALIZE THE CURRENT DISK
                        ;LRET = FALSE ;SET TO TRUE IF $ FILE EXISTS
                        ;COMPUTE THE LENGTH OF THE ALLOCATION VECTOR - 2
0EC2 2A83150E03         LHLD MAXALL! MVI C,3 ;PERFORM MAXALL 8
                        ;NUMBER OF BYTES IN ALLOC VECTOR IS (MAXALL/8)+1
```

```
0EC7 CDF30C23           CALL HLROTR! INX H ;HL = MAXALL/8+1
0ECB 444D               MOV B,H! MOV C,L ;COUNT DOWN BC TIL ZERO
0ECD 2A7C15             LHLD ALLOCA ;BASE OF ALLOCATION VECTOR
                        ;FILL THE ALLOCATION VECTOR WITH ZEROS
                        INITIAL0:
0ED0 360023                     MVI M,0! INX H ;ALLOC(I)=0
0ED3 0B                         DCX B ;COUNT LENGTH DOWN
0ED4 78B1C2D00E                 MOV A,B! ORA C! JNZ INITIAL0
                        ;SET THE RESERVED SPACE FOR THE DIRECTORY
0ED9 2A8715EB           LHLD DIRBLK! XCHG
0EDD 2A7C15             LHLD ALLOCA ;HL= ALLOC()
0EE0 732372             MOV M,E! INX H! MOV M,D ;SETS RESERVED DIRECTORY BLKS
                        ;ALLOCATION VECTOR INITIALIZED, HOME DISK
0EE3 CDBC0B             CALL HOME
                        ;CDRMAX = 3 (SCANS AT LEAST ONE DIRECTORY RECORD)
0EE6 2A70153603         LHLD CDRMAXA! MVI M,3! INX H! MVI M,0
                        ;CDRMAX = 0000
0EEB CD230E             CALL SET$END$DIR ;DCNT = ENDDIR
                        ;READ DIRECTORY ENTRIES AND CHECK FOR ALLOCATED STORAGE
                        INITIAL2:
0EF1 0EFFCD2A0E                 MVI C,TRUE! CALL READ$DIR
0EF6 CD1A0EC8                   CALL END$OF$DIR! RZ ;RETURN IF END OF DIRECTORY
                                ;NOT END OF DIRECTORY, VALID ENTRY?
0EFA CD790D                     CALL GETDPTRA ;HL = BUFFA + DPTR
0EFD 3EE5BE                     MVI A,EMPTY! CMP M
0F00 CAF10E                     JZ INITIAL2 ;GO GET ANOTHER ITEM
                                ;NOT EMPTY, USER CODE THE SAME?
0F03 3A470B                     LDA USRCODE
0F06 BEC2150F                   CMP M! JNZ PDOLLAR
                                ;SAME USER CODE, CHECK FOR '$' SUBMIT
0F0A 237E                       INX H! MOV A,M ;FIRST CHARACTER
0F0C D624                       SUI '$' ;DOLLAR FILE?
0F0E C2150F                     JNZ PDOLLAR
                                ;DOLLAR FILE FOUND, MARK IN LRET
0F11 3D324B0B                   DCR A! STA LRET ;LRET = 255
                        PDOLLAR:
                                ;NOW SCAN THE DISK MAP FOR ALLOCATED BLOCKS
0F15 0E01                       MVI C,1 ;SET TO ALLOCATED
0F17 CD940E                     CALL SCANDM
0F1A CDAB0D                     CALL SETCDR ;SET CDRMAX TO DCNT
0F1D C3F10E                     JMP INITIAL2 ;FOR ANOTHER ENTRY
                ;
                COPY$DIRLOC:
                        ;COPY DIRECTORY LOCATION TO LRET FOLLOWING
                        ;DELETE, RENAME, ... OPS
0F20 3A91153248         LDA DIRLOC! STA LRET
0F26 C9                 RET
                ;
                COMPEXT:
                        ;COMPARE EXTENT# IN A WITH THAT IN C, RETURN NONZERO
                        ;IF THEY DO NOT MATCH
0F27 C5                 PUSH B ;SAVE C'S ORIGINAL VALUE
0F28 F53A82152F         PUSH PSW! LDA EXTMSK! CMA! MOV B,A
                        ;B HAS NEGATED FORM OF EXTENT MASK
0F2E 79A04F             MOV A,C! ANA B! MOV C,A ;LOW BITS REMOVED FROM C
0F31 F1A0               POP PSW! ANA B ;LOW BITS REMOVED FROM A
```



```
0F33 91E61F             SUB C! ANI MAXEXT ;SET FLAGS
0F36 C1                 POP B ;RESTORE ORIGINAL VALUES
0F37 C9                 RET
                ;
                SEARCHN:
                        ;SEARCH FOR THE NEXT DIRECTORY ELEMENT, ASSUMING
                        ;A PREVIOUS CALL ON SEARCH WHICH SETS SEARCHA AND
                        ;SEARCHL
0F38 0E00CD2A0E         MVI C,FALSE! CALL READ$DIR ;READ NEXT DIR ELEMENT
0F3D CD1A0ECA9F         CALL END$OF$DIR! JZ SEARCH$FIN ;SKIP TO END IF SO
                                ;NOT END OF DIRECTORY, SCAN FOR MATCH
0F43 2A9615EB                   LHLD SEARCHA! XCHG ;DE=BEGINNING OF USER FCB
0F47 1A                         LDAX D ;FIRST CHARACTER
0F48 FEE5                       CPI EMPTY ;KEEP SCANNING IF EMPTY
0F4A CA550F                     JZ SEARCHNEXT
                                ;NOT EMPTY, MAY BE END OF LOGICAL DIRECTORY
0F4D D5                         PUSH D ;SAVE SEARCH ADDRESS
0F4E CD9E0D                     CALL COMPCDR ;PAST LOGICAL END?
0F51 D1                         POP D ;RECALL ADDRESS
0F52 D29F0F                     JNC SEARCH$FIN ;ARTIFICIAL STOP
                        SEARCHNEXT:
0F55 CD790D                     CALL GETDPTRA ;HL = BUFFA+DPTR
0F58 3A95154F                   LDA SEARCHL! MOV C,A ;LENGTH OF SEARCH TO C
0F5C 0600                       MVI B,0 ;B COUNTS UP, C COUNTS DOWN
                                SEARCHLOOP:
0F5E 79B7CA8E0F                 MOV A,C! ORA A! JZ ENDSEARCH
0F63 1AFE3FCA87                 LDAX D! CPI '?'! JZ SEARCHOK ;? MATCHES
                                ;SCAN NEXT CHARACTER IF NOT UBYTES
0F69 78FE0DCA87                 MOV A,B! CPI UBYTES! JZ SEARCHOK
                                ;NOT THE UBYTES FIELD, EXTENT FIELD?
0F6F FE0C                       CPI EXTNUM ;MAY BE EXTENT FIELD
0F71 1A                         LDAX D ;FCB CHARACTER
0F72 CA7E0F                     JZ SEARCHEXT ;SKIP TO SEARCH EXTENT
0F75 96E67F                     SUB M! ANI 7FH ;MASK-OUT FLAGS, EXTENT MO
0F78 C2380F                     JNZ SEARCHN ;SKIP IF NOT MATCHED
0F7B C3870F                     JMP SEARCHOK ;MATCHED CHARACTER
                        SEARCHEXT:
                                ;A HAS FCB CHARACTER
                                ;ATTEMPT AN EXTENT # MATCH
0F7E C5                         PUSH B ;SAVE COUNTERS
0F7F 4E                         MOV C,M ;DIRECTORY CHARACTER TO C
0F80 CD270F                     CALL COMPEXT ;COMPARE USER/DIR CHAR
0F83 C1                         POP B ;RECALL COUNTERS
0F84 C2380F                     JNZ SEARCHN ;SKIP IF NO MATCH
                        SEARCHOK:
                                ;CURRENT CHARACTER MATCHES
0F87 1323040D                   INX D! INX H! INR B! DCR C
0F8B C35E0F                     JMP SEARCHLOOP
                        ENDSEARCH:
                                ;ENTIRE NAME MATCHES, RETURN DIR POSITIO
0F8F 3AA515E603                 LDA DCNT! ANI DSKMSK! STA LRET
                                ;LRET = LOW(DCNT) AND 11B
0F9E 2191157E17                 LXI H,DIRLOC! MOV A,M! RAL! RNC ;DIRLOC=
                                ;YES, CHANGE IT TO 0 TO MARK AS FOUND
0F9C AF77                       XRA A! MOV M,A ;DIRLOC=0
0F9E C9                         RET
```

```
CP/M MACRO ASSEM 2.0     #023     Bdos Interface  Bdos, Version 2.0 Aug, 1979

                                  SEARCH$FIN:
                                          ;END OF DIRECTORY, OR EMPTY NAME
0F9F CD230E                               CALL SET$END$DIR ;MAY BE ARTIFICAL END
0FA2 3EFF324B0B                           MVI A,255! STA LRET! RET
                         ;
                         SEARCH:
                                  ;SEARCH FOR DIRECTORY ELEMENT OF LENGTH C AT INFO
0FA8 3EFF329115                   MVI A,0FFH! STA DIRLOC ;CHANGED IF ACTUALLY FOUND
0FAD 21951571                     LXI H,SEARCHL! MOV M,C ;SEARCHL = C
0FB1 2A490B2296                   LHLD INFO! SHLD SEARCHA ;SEARCHA = INFO
0FB7 CD230E                       CALL SET$END$DIR ;DCNT = ENLDIR
0FBA CDBC0B                       CALL HOME ;TO START AT THE BEGINNING
0FBD C3380F                       JMP SEARCHN ;START THE SEARCH OPERATION
                         ;
                         DELETE:
                                  ;DELETE THE CURRENTLY ADDRESSED FILE
0FC0 CD6D0D                       CALL CHECK$WRITE ;WRITE PROTECTED?
0FC3 0E0CCDA80F                   MVI C,EXTNUM! CALL SEARCH ;SEARCH THROUGH FILE TYPE
                                  DELETE0:
                                          ;LOOP WHILE DIRECTORY MATCHES
0FC8 CD1A0EC8                             CALL END$OF$DIR! RZ ;STOP IF END
                                          ;SET EACH NON ZERO DISK MAP ENTRY TO 0
                                          ;IN THE ALLOCATION VECTOR
                                          ;MAY BE R/O FILE
0FCC CD670D                               CALL CHECK$RODIR ;RO DISK ERROR IF FOUND
0FCF CD790D                               CALL GETDPTRA ;HL=.BUFF DPTR)
0FD2 36E5                                 MVI M,EMPTY
0FD4 0E00CD940E                           MVI C,0! CALL SCANDM ;ALLOC ELTS SET TO 0
0FD9 CDF70D                               CALL WRDIR ;WRITE THE DIRECTORY
0FDC CD380F                               CALL SEARCHN ;TO NEXT ELEMENT
0FDF C3C80F                               JMP DELETE0 ;FOR ANOTHER RECORD
                         ;
                         GET$BLOCK:
                                  ;GIVEN ALLOCATION VECTOR POSITION BC, FIND THE ZERO BIT
                                  ;CLOSEST TO THIS POSITION BY SEARCHING LEFT AND RIGHT
                                  ;IF FOUND, SET THE BIT TO ONE AND RETURN THE BIT POSITIO
                                  ;IN HL.  IF NOT FOUND (I.E., WE PASS 0 ON THE LEFT  OR
                                  ;MAXALL ON THE RIGHT), RETURN 0000 IN HL
0FE2 5059                         MOV D,B! MOV E,C ;COPY OF STARTING POSITION TO DE
                                  LEFTTST:
0FE4 79B0CAF50F                           MOV A,C! ORA B! JZ RIGHTTST ;SKIP IF LEFT 0000
                                          ;LEFT NOT AT POSITION ZERO, BIT ZERO?
0FE9 0BD5C5                               DCX B! PUSH D! PUSH B ;LEFT,RIGHT PUSHED
0FEC CD620E                               CALL GETALLOCBIT
0FEF 1FD21010                             RAR! JNC RETBLOCK ;RETURN BLOCK NUMBER IF ZERO
                                          ;BIT IS ONE, SO TRY THE RIGHT
0FF3 C1D1                                 POP B! POP D ;LEFT, RIGHT RESTORED
                                  RIGHTTST:
0FF5 2A8315                               LHLD MAXALL ;VALUE OF MAXIMUM ALLOCATION#
0FF8 7B957A9C                             MOV A,E! SUB L! MOV A,D! SBB H ;RIGHT-MAXALL?
0FFC D21810                               JNC RETBLOCK0 ;RETURN BLOCK 0000 IF SO
0FFF 13C5D5                               INX D! PUSH B! PUSH D ;LEFT, RIGHT PUSHED
1002 424B                                 MOV B,D! MOV C,E ;READY RIGHT FOR CALL
1004 CD620F                               CALL GETALLOCBIT
1007 1FD21010                             RAR! JNC RETBLOCK ;RETURN BLOCK NUMBER IF ZERO
100B D1C1                                 POP D! POP B ;RESTORE LEFT AND RIGHT POINTERS
```



```
CP/M MACRO ASSEM 2.0     #024     Bdos Interface, Bdos, Version 2.0 Aug, 1979

100D C3E40F                               JMP LEFTTST ;FOR ANOTHER ATTEMPT
                                  RETBLOCK:
1010 173C                                 RAL! INR A ;BIT BACK INTO POSITION AND SET TO 1
                                          ;D CONTAINS THE NUMBER OF SHIFTS REQUIRED TO REP
1012 CD5B0E                               CALL ROTR ;MOVE BIT BACK TO POSITION AND STORE
1015 E1D1                                 POP H! POP D ;HL RETURNED VALUE, DE DISCARDED
1017 C9                                   RET
                                  RETBLOCK0:
                                          ;CANNOT FIND AN AVAILABLE BIT  RETURN 0000
101E 21000C9                              LXI H,0000H! RET
                         ;
                         COPY$DIR:
                                  ;COPY FCB INFORMATION STARTING AT C FOR E BYTES
                                  ;INTO THE CURRENTLY ADDRESSED DIRECTORY ENTRY
101C D5                           PUSH D ;SAVE LENGTH FOR LATER
101D 0600                         MVI B,0 ;DOUBLE INDEX TO BC
101F 2A490B                       LHLD INFO ;HL = SOURCE FOR DATA
1022 09EB                         DAD B! XCHG ;DE=.FCB(C), SOURCE FOR COPY
1024 CD790D                       CALL GETDPTRA ;HL=.BUFF(DPTR), DESTINATION
1027 C1                           POP B ;DE=SOURCE, HL=DEST, C=LENGTH
1028 CD6A0B                       CALL MOVE ;DATA MOVED
                                  SEEK$COPY:
                                  ;ENTER FROM CLOSE TO SEEK AND COPY CURRENT ELEMENT
102B CD000D                       CALL SEEK$DIR ;TO THE DIRECTORY ELEMENT
102E C3F70D                       JMP WRDIR ;WRITE THE DIRECTORY ELEMENT
                                  ;RET
                         ;
                         COPY$FCB:
                                  ;COPY THE ENTIRE FILE CONTROL BLOCK
1031 0E001F20                     MVI C,0! MVI E,FCBLEN ;START AT 0, TO FCBLEN-1
1035 C31C10                       JMP COPY$DIR
                         ;
                         RENAME:
                                  ;RENAME THE FILE DESCRIBED BY THE FIRST HALF OF
                                  ;THE CURRENTLY ADDRESSED FILE CONTROL BLOCK.  THE
                                  ;NEW NAME IS CONTAINED IN THE LAST HALF OF THE
                                  ;CURRENTLY ADDRESSED FILE CONROL BLOCK.  THE FILE
                                  ;NAME AND TYPE ARE CHANGED, BUT THE REEL NUMBER
                                  ;IS IGNORED.  THE USER NUMBER IS IDENTICAL
1038 CD6D0D                       CALL CHECK$WRITE ;MAY BE WRITE PROTECTED
                                  ;SEARCH UP TO THE EXTENT FIELD
103B 0E0CCDA80F                   MVI C,EXTNUM! CALL SEARCH
                                  ;COPY POSITION 0
1040 2A490B7F                     LHLD INFO! MOV A,M ;HL=.FCB(0), A=FCB(0)
1044 11100019                     LXI D,DSKMAP! DAD D;HL=.FCB(DSKMAP)
1048 77                           MOV M,A ;FCB(DSKMAP)=FCB(0)
                                  ;ASSUME THE SAME DISK DRIVE FOR NEW NAMED FILE
                                  RENAME0:
1049 CD1A0EC8                             CALL END$OF$DIR! RZ ;STOP AT END OF DIR
                                          ;NOT END OF DIRECTORY, RENAME NEXT ELEMENT
104D CD670D                               CALL CHECK$RODIR ;MAY BE READ-ONLY FILE
1050 0E101F0CCD                           MVI C,DSKMAP! MVI E,EXTNUM! CALL COPY$DIR
                                          ;ELEMENT RENAMED  MOVE TO NEXT
1057 CD380F                               CALL SEARCHN
105A C34910                               JMP RENAME0
                         ;
```

```
CP/M MACRO ASSEM 2.0     #025     Bdos Interface. Bdos  Version 2.0 Aug. 1979

                INDICATORS
                        ;SET FILE INDICATORS FOR CURRENT FCB
105D 0E0CCDA80F         MVI C,EXTNUM! CALL SEARCH ;THROUGH FILE TYPE
                        INDIC0:
1062 CD1A0EC8                   CALL END$OF$DIR! RZ ;STOP AT END OF DIR
                                ;NOT END OF DIRECTORY, CONTINUE TO CHANGE
1066 0E001E0C                   MVI C,0! MVI E,EXTNUM ;COPY NAME
106A CD1C10                     CALL COPY$DIR
106D CD380F                     CALL SEARCHN
1070 C36210                     JMP INDIC0
                ;
                OPEN:
                        ;SEARCH FOR THE DIRECTORY ENTRY, COPY TO FCB
1073 0E0FCDA80F         MVI C,NAMLEN! CALL SEARCH
1078 CD1A0EC8           CALL END$OF$DIR! RZ ;RETURN WITH LRET=255 IF END
                        ;NOT END OF DIRECTORY, COPY FCB INFORMATION
                OPEN$COPY:
                        ;REFERENCED BELOW TO COPY FCB INFO
107C CDB50C7EF5         CALL GETEXTA! MOV A,M! PUSH PSW! PUSH H ;SAVE EXTENT#
1082 CD790EFB           CALL GETDPTRA! XCHG ;DE = .BUFF(DPTR)
1086 2A490B             LHLD INFO ;HL = FCB(0)
1089 0E20               MVI C,NXTREC ;LENGTH OF MOVE OPERATION
108B D5                 PUSH D ;SAVE .BUFF(DPTR)
108C CD6A0B             CALL MOVE ;FROM .BUFF(DPTR) TO .FCB(0)
                        ;NOTE THAT ENTIRE FCB IS COPIED  INCLUDING INDICATORS
108F CD910D             CALL SETFWF ;SETS FILE WRITE FLAG
1092 D1210C0019         POP D! LXI H,EXTNUM! DAD D ;HL=.BUFF(DPTR+EXTNUM)
1097 4E                 MOV C,M ;C = DIRECTORY EXTENT NUMBER
1098 210F0019           LXI H,RECCNT! DAD D ;HL=.BUFF(DPTR+RECCNT)
109C 46                 MOV B,M ;B HOLDS DIRECTORY RECORD COUNT
109D E1F177             POP H! POP PSW! MOV M,A ;RESTORE EXTENT NUMBER
                        ;HL = .USER EXTENT#, B = DIR REC CNT, C = DIR EXTENT#
                        ;IF USER EXT < DIR EXT THEN USER := 128 RECORDS
                        ;IF USER EXT = DIR EXT THEN USER := DIR RECORDS
                        ;IF USER EXT > DIR EXT THEN USER := 0 RECORDS
10A0 79BE78                     MOV A,C! CMP M! MOV A,B ;READY DIR RECCNT
10A3 CAAD10                     JZ OPEN$RCNT ;IF SAME, USER GETS DIR RECCNT
10A6 3E00DAAD10                 MVI A,0! JC OPEN$RCNT ;USER IS LARGER
10AB 3E80                       MVI A,128 ;DIRECTORY IS LARGER
                        OPEN$RCNT: ;A HAS RECORD COUNT TO FILL
10AD 2A490B110F         LHLD INFO! LXI D,RECCNT! DAD D! MOV M,A
10B5 C9                 RET
                ;
                MERGEZERO:
                        ;HL = .FCB1(I), DE = .FCB2(I),
                        ;IF FCB1(I) = 0 THEN FCB1(I) := FCB2(I)
10B6 7E23B62BC0         MOV A,M! INX H! ORA M! DCX H! RNZ ;RETURN IF = 0000
10BB 1A771323           LDAX D! MOV M,A! INX D! INX H ;LOW BYTE COPIED
10BF 1A771B2B           LDAX D! MOV M,A! DCX D! DCX H ;BACK TO INPUT FORM
10C3 C9                 RET
                ;
                CLOSE:
                        ;LOCATE THE DIRECTORY ELEMENT AND RE-WRITE IT
10C4 AF324B0B           XRA A! STA LRET
10C8 CD380DC0           CALL NOWRITE! RNZ ;SKIP CLOSE IF R/O DISK
                        ;CHECK FILE WRITE FLAG - 0 INDICATES WRITTEN
```

```
CP/M MACRO ASSEM 2.0     #026     Bdos Interface, Bdos, Version 2.0 Aug, 1979

10CC CD820D             CALL GETMODNUM ;FCB(MODNUM) IN A
10CF E680C0             ANI FWFMSK! RNZ ;RETURN IF BIT REMAINS SET
10D2 0E0FCDA80F         MVI C,NAMLEN! CALL SEARCH ;LOCATE FILE
10D7 CD1A0EC8           CALL END$OF$DIR! RZ ;RETURN IF NOT FOUND
                        ;MERGE THE DISK MAP AT INFO WITH THAT AT BUFF(DPTR)
10DB 011000CD79         LXI B,DSKMAP! CALL GETDPTRA
10E1 09EB               DAD B! XCHG ;DE IS .BUFF(DPTR+16)
10E3 2A490B09           LHLD INFO! DAD B ;DE=.BUFF(DPTR+16), HL=.FCB(16)
10E7 0E10               MVI C,(FCBLEN-DSKMAP) ;LENGTH OF SINGLE BYTE DM
                        MERGE0:
10E9 3A9A15B7CA                 LDA SINGLE! ORA A! JZ MERGED ;SKIP TO DOUBLE
                                ;THIS IS A SINGLE BYTE MAP
                                ;IF FCB(I) = 0 THEN FCB(I) := BUFF(I)
                                ;IF BUFF(I) = 0 THEN BUFF(I) := FCB(I)
                                ;IF FCB(I) <> BUFF(I) THEN ERROR
10F0 7EB71AC2F7                 MOV A,M! ORA A! LDAX D! JNZ FCBNZERO
                                        ;FCB(I) = 0
10F6 77                         MOV M,A ;FCB(I) = BUFF(I)
                                FCBNZERO:
10F7 B7C2FD10                   ORA A! JNZ BUFFNZERO
                                        ;BUFF(I) = 0
10FB 7E12                       MOV A,M! STAX D ;BUFF(I) := FCB(I)
                                BUFFNZERO:
10FD BEC23C11           CMP M! JNZ MERGERR ;FCB(I) = BUFF(I)?
1101 C31911             JMP LMSET ;IF MERGE OK
                        MERGED:
                                ;THIS IS A DOUBLE BYTE MERGE OPERATION
1104 CDB610             CALL MERGEZERO ;BUFF = FCB IF BUFF 0000
1107 EBCDB610EB         XCHG! CALL MERGEZERO! XCHG ;FCB = BUFF IF FCB 00
                        ;THEY SHOULD BE IDENTICAL AT THIS POINT
110C 1ABEC23C11         LDAX D! CMP M! JNZ MERGERR ;LOW SAME?
1111 1323               INX D! INX H ;TO HIGH BYTE
1113 1ABEC23C11         LDAX D! CMP M! JNZ MERGERR ;HIGH SAME?
                        ;MERGE OPERATION OK FOR THIS PAIR
1118 0D                 DCR C ;EXTRA COUNT FOR DOUBLE BYTE
                        DMSET:
1119 1323               INX D! INX H ;TO NEXT BYTE POSITION
111B 0DC2E910           DCR C! JNZ MERGE0 ;FOR MORE
                        ;END OF DISK MAP MERGE, CHECK RECORD COUNT
                        ;DE = .BUFF(DPTR)+32, HL = .FCB(32)
111F 01ECFF09EB         LXI B,-(FCBLEN-EXTNUM)! DAD B! XCHG! DAD B
                        ;DE = .FCB(EXTNUM), HL = .BUFF(DPTR+EXTNUM)
1124 1A                 LDAX D ;CURRENT USER EXTENT NUMBER
                        ;IF FCB(EXT) >= BUFF(FCB) THEN
                        ;BUFF(EXT) := FCB(EXT), BUFF(REC) := FCB(REC)
1126 BEDA3311           CMP M! JC ENDMERGE
                        ;FCB EXTENT NUMBER >= DIR EXTENT NUMBER
112A 77                 MOV M,A ;BUFF(EXT) = FCB(EXT)
                        ;UPDATE DIRECTORY RECORD COUNT FIELD
112B 010700091EB        LXI B,(RECCNT-EXTNUM)! DAD B! XCHG! DAD B
                        ;DE=.BUFF(RECCNT), HL=.FCB(RECCNT)
1131 7E12               MOV A,M! STAX D ;BUFF(RECCNT)=FCB(RECCNT)
                        ENDMERGE:
1133 3EFF328F15         MVI A,TRUE! STA FCB$COPIED ;MARK AS COPIED
1138 CD2B10             CALL SEEK$COPY ;OK TO "WRDIR" HERE - 1.4 COMPAT
113B C9                 RET
```

```
                        MERGERR:
                                ;ELEMENTS DID NOT MERGE CORRECTLY
113C 214B0B35                   LXI H,LRET! DCR M ;=255 NON ZERO FLAG SET
1140 C9                         RET
                        ;
                        MAKE:
                                ;CREATE A NEW FILE BY CREATING A DIRECTORY ENTRY
                                ;THEN OPENING THE FILE
1141 CD6D0D                     CALL CHECK$WRITE ;MAY BE WRITE PROTECTED
1144 2A490BE5                   LHLD INFO! PUSH H ;SAVE FCB ADDRESS, LOOK FOR E5
1148 2169152249                 LXI H,EFCB! SHLD INFO ;INFO = .EMPTY
114E 0E01CDA80F                 MVI C,1! CALL SEARCH ;LENGTH 1 MATCH ON EMPTY ENTRY
1153 CD1A0E                     CALL END$OF$DIR ;ZERO FLAG SET IF NO SPACE
1156 E1                         POP H ;RECALL INFO ADDRESS
1157 22490B                     SHLD INFO ;IN CASE WE RETURN HERE
115A C8                         RZ ;RETURN WITH ERROR CONDITION 255 IF NOT FOUND
115B EB                         XCHG ;DE = INFO ADDRESS
                                ;CLEAR THE REMAINDER OF THE FCB
115C 210F0019                   LXI H,NAMLEN! DAD D ;HL=.FCB(NAMLEN)
1160 0E11                       MVI C,FCBLEN-NAMLEN ;NUMBER OF BYTES TO FILL
1162 AF                         XRA A ;CLEAR ACCUMULATOR TO 00 FOR FILL
                                MAKE0:
1163 77230DC263                         MOV M,A! INX HI DCR CI JNZ MAKE0
1169 210D0019                   LXI H,UBYTES! DAD D ;HL = .FCB(UBYTES)
116D 77                         MOV M,A ;FCB(UBYTES) = 0
116E CDAB0D                     CALL SETCDR ;MAY HAVE EXTENDED THE DIRECTORY
                                ;NOW COPY ENTRY TO THE DIRECTORY
1171 CD3110                     CALL COPY$FCB
                                ;AND SET THE FILE WRITE FLAG TO "1"
1174 C3910D                     JMP SETFWF
                                ;RET
                        ;
                        OPEN$REEL:
                                ;CLOSE THE CURRENT EXTENT, AND OPEN THE NEXT ONE
                                ;IF POSSIBLE.  RMF IS TRUE IF IN READ MODE
1177 AF328F15                           XRA A! STA FCB$COPIED ;SET TRUE IF ACTUALLY COPI
117B CDC410                             CALL CLOSE ;CLOSE CURRENT EXTENT
                                        ;LRET REMAINS AT ENDDIR IF WE CANNOT OPEN THE NE
117F CD1A0FC8                           CALL END$OF$DIR! RZ ;RETURN IF END
                                ;INCREMENT EXTENT NUMBER
1182 2A490B010C                 LHLD INFO! LXI B,EXTNUM! DAD B ;HL=.FCB(EXTNUM)
1189 7E3CE61F77                 MOV A,M! INR A! ANI MAXEXT! MOV M,A ;FCB(EXTNUM)=++1
118F CAA011                     JZ OPEN$MOD ;MOVE TO NEXT MODULE IF ZERO
                                ;MAY BE IN THE SAME EXTENT GROUP
1191 473A8215A0                 MOV B,A! LDA EXTMSK! ANA B
                                ;IF RESULT IS ZERO, THEN NOT IN THE SAME GROUP
1196 218F15                     LXI H,FCB$COPIED ;TRUE IF THE FCB WAS COPIED TO DIRECTOR
1199 A6                         ANA M ;PRODUCES A 00 IN ACCUMULATOR IF NOT WRITTEN
119A CAAB11                     JZ OPEN$REEL0 ;GO TO NEXT PHYSICAL EXTENT
                                ;RESULT IS NON ZERO, SO WE MUST BE IN SAME LOGICAL EXT
119D C3C911                     JMP OPEN$REEL1 ;TO COPY FCB INFORMATION
                                OPEN$MOD:
                                        ;EXTENT NUMBER OVERFLOW, GO TO NEXT MODULE
11A0 01020009                           LXI B,(MODNUM-EXTNUM)! DAD B ;HL=.FCB(MODNUM)
11A4 34                                 INR M ;FCB(MODNUM)=++1
                                        ;MODULE NUMBER INCREMENTED, CHECK FOR OVERFLOW
```



```
11A5 7EE60F                             MOV A,M! ANI MAXMOD ;MASK HIGH ORDER BITS
11A8 CAD411                             JZ OPEN$R$ERR ;CANNOT OVERFLOW TO ZERO
                                        ;OTHERWISE, OK TO CONTINUE WITH NEW MODULE
                                OPEN$REEL0:
11AB 0E0FCDA80F                         MVI C,NAMLEN! CALL SEARCH ;NEXT EXTENT FOUND?
11B0 C21A0EC2C9                         CALL END$OF$DIR! JNZ OPEN$REEL1
                                        ;END OF FILE ENCOUNTERED
11B6 3A901E3C                           LDA RMF! INR A ;0FFH BECOMES 00 IF READ
11BA CAD411                             JZ OPEN$R$ERR ;SETS LRET = 1
                                        ;TRY TO EXTEND THE CURRENT FILE
11BD CD4111                             CALL MAKE
                                        ;CANNOT BE END OF DIRECTORY
11C0 CD1A0E                             CALL END$OF$DIR
11C3 CAD411                             JZ OPEN$R$ERR ;WITH LRET = 1
11C6 C3CC11                             JMP OPEN$REEL2
                                OPEN$REEL1:
                                        ;NOT END OF FILE, OPEN
11C9 CD7C10                             CALL OPEN$COPY
                                OPEN$REEL2:
11CC CDCA0C                             CALL GETFCB ;SET PARAMETERS
11CF AF324B0B                           XRA A! STA LRET ;LRET = 0
11D3 C9                                 RET ;WITH LRET = 0
                        OPEN$R$ERR:
                                ;CANNOT MOVE TO NEXT EXTENT OF THIS FILE
11D4 CD980D                     CALL SETLRET1 ;LRET = 1
11D7 C3910D                     JMP SETFWF ;ENSURE THAT IT WILL NOT BE CLOSED
                                ;RET
                ;
                SEQDISKREAD:
                        ;SEQUENTIAL DISK READ OPERATION
11DA 3E01329215         MVI A,1! STA SEQIO
                        ;DROP THROUGH TO DISKREAD
                ;
                DISKREAD:       ;(MAY ENTER FROM SEQDISKREAD)
11DF 3EFF329015         MVI A,TRUE! STA RMF ;READ MODE FLAG = TRUE (OPEN$REEL)
                        ;READ THE NEXT RECORD FROM THE CURRENT FCB
11E4 CDCA0C             CALL GETFCB ;SETS PARAMETERS FOR THE READ
11E7 3AA015219E         LDA VRECORD! LXI H,RCOUNT! CMP M ;VRECORD-RCOUNT
                        ;SKIP IF RCOUNT > VRECORD
11EF DA0412             JC RECORDOK
                                ;NOT ENOUGH RECORDS IN THE EXTENT
                                ;RECORD COUNT MUST BE 128 TO CONTINUE
11F1 FE80                       CPI 128 ;VRECORD = 128?
11F3 C21A12                     JNZ DISKEOF ;SKIP IF VRECORD<>128
11F6 CD7711                     CALL OPEN$REEL ;GO TO NEXT EXTENT IF SO
11F9 AF32A015                   XRA A! STA VRECORD ;VRECORD=00
                                ;NOW CHECK FOR OPEN OK
11FD 3A4B0BB7C2                 LDA LRET! ORA A! JNZ DISKEOF ;STOP AT EOF
                        RECORDOK:
                                ;ARRIVE WITH FCB ADDRESSING A RECORD TO READ
1204 CD890C                     CALL INDEX
                                ;ERROR 2 IF READING UNWRITTEN DATA
                                ;(RETURNS 1 TO BE COMPATIBLE WITH 1.4)
1207 CD960C                     CALL ALLOCATED ;ARECORD=00007
120A CA1A12                     JZ DISKEOF
                                ;RECORD HAS BEEN ALLOCATED, READ IT
```

```
CP/M MACRO ASSEM 2.0     #029    Bdos Interface, Bdos, Version 2.0 Aug, 1979

120D CD9C0C                      CALL ATRAN ;ARECORD NOW A DISK ADDRESS
1210 CDE60B                      CALL SEEK ;TO PROPER TRACK,SECTOR
1213 CDD60B                      CALL RDBUFF ;TO DMA ADDRESS
1216 CDE10C                      CALL SETFCB ;REPLACE PARAMETERS
1219 C9                          RET
                         DISKEOF:
121A C3980D                      JMP SETLRET1 ;LRET = 1
                                 ;RET
                ;
                SEQDISKWRITE:
                         ;SEQUENTIAL DISK WRITE
121D 3E01329215          MVI A,1! STA SEQIO
                         ;DROP THROUGH TO DISKWRITE
                ;
                DISKWRITE:       ;(MAY ENTER HERE FROM SEQDISKWRITE ABOVE)
1222 3E00329015          MVI A,FALSE! STA RMF ;READ MODE FLAG
                         ;WRITE RECORD TO CURRENTLY SELECTED FILE
1227 CD6D0D              CALL CHECK$WRITE ;IN CASE WRITE PROTECTED
122A 2A490B              LHLD INFO ;HL = .FCB(0)
122D CD5D0D              CALL CHECK$ROFILE ;MAY BE A READ-ONLY FILE
1230 CDCA0C              CALL GETFCB ;TO SET LOCAL PARAMETERS
1233 3AA015FE80          LDA VRECORD! CPI LSTREC+1 ;VRECORD-128
                         ;SKIP IF VRECORD > LSTREC
1238 DA3F12              JC DISKWR0
                                 ;VRECORD = 128, CANNOT OPEN NEXT EXTENT
123B CD980DC9                    CALL SETLRET1! RET ;LRET=1
                         DISKWR0:
                         ;CAN WRITE THE NEXT RECORD, SO CONTINUE
123F CD890C              CALL INDEX
1242 CD9F0C              CALL ALLOCATED
1245 0E00                MVI C,0 ;MARKED AS NORMAL WRITE OPERATION FOR WRBUFF
1247 C29212              JNZ DISKWR1
                                 ;NOT ALLOCATED
                                 ;THE ARGUMENT TO GETBLOCK IS THE STARTING
                                 ;POSITION FOR THE DISK SEARCH, AND SHOULD BE
                                 ;THE LAST ALLOCATED BLOCK FOR THIS FILE, OR
                                 ;THE VALUE 0 IF NO SPACE HAS BEEN ALLOCATED
124A CD500C                      CALL DM$POSITION
124D 329415                      STA DMINX ;SAVE FOR LATER
1250 010000                      LXI B,0000H ;MAY USE BLOCK ZERO
1253 B7CA5E12                    ORA A! JZ NOPBLOCK ;SKIP IF NO PREVIOUS BLOCK
                                         ;PREVIOUS BLOCK EXISTS AT A
1257 4F0B                                MOV C,A! DCX B ;PREVIOUS BLOCK # IN BC
1259 CD700C                              CALL GETDM ;PREVIOUS BLOCK # TO HL
125C 444D                                MOV B,H! MOV C,L ;BC=PREV BLOCK#
                                 NOPBLOCK:
                                         ;BC = 0000, OR PREVIOUS BLOCK #
125F CDF20F                              CALL GET$BLOCK ;BLOCK # TO HL
                                 ;ARRIVE HERE WITH BLOCK# OR ZERO
1261 7DB4C26C12                  MOV A,L! ORA H! JNZ BLOCKOK
                                         ;CANNOT FIND A BLOCK TO ALLOCATE
1266 3E02324B0B                          MVI A,2! STA LRET! RET ;LRET=2
                                 BLOCKOK:
                                 ;ALLOCATED BLOCK NUMBER IS IN HL
126C 22A215                      SHLD ARECORD
126F EB                          XCHG ;BLOCK NUMBER TO DE
```



```
CP/M MACRO ASSEM 2.0     #030    Bdos Interface, Bdos, Version 2.0 Aug, 1979

1270 2A490B0110                  LHLD INFO! LXI B,DSKMAP! DAD B ;HL=.FCB(DSKMAP)
1273 3A9A15B7                    LDA SINGLE! ORA A ;SET FLAGS FOR SINGLE BYTE DM
1278 3A9415                      LDA DMINX ;RECALL DM INDEX
127B CA8812                      JZ ALLOCWD ;SKIP IF ALLOCATING WORD
                                         ;ALLOCATING A BYTE VALUE
127E CD740D73                            CALL ADDH! MOV M,E ;SINGLE BYTE ALLOC
1282 C39C12                              JMP DISKWRU ;TO CONTINUE
                                 ALLOCWD:
                                 ;ALLOCATE A WORD VALUE
1285 4F0600                              MOV C,A! MVI B,0 ;DOUBLE(DMINX)
1288 0909                                DAD B! DAD B ;HL=.FCB(DMINX*2)
128A 732372                              MOV M,E! INX H! MOV M,D ;DOUBLE WD
                                 DISKWRU:
                                 ;DISK WRITE TO PREVIOUSLY UNALLOCATED BLOCK
128D 0E02                        MVI C,2 ;MARKED AS UNALLOCATED WRITE
                         DISKWR1:
                         ;CONTINUE THE WRITE OPERATION OF NO ALLOCATION ERROR
                         ;C = 0 IF NORMAL WRITE, 2 IF TO PREV UNALLOC BLOCK
128F 3A4B0BB7C0          LDA LRET! ORA A! RNZ ;STOP IF NON ZERO RETURNED VALUE
1294 C5                  PUSH B ;SAVE WRITE FLAG
1295 CD9C0C              CALL ATRAN ;ARECORD SET
1298 CDE60B              CALL SEEK ;TO PROPER FILE POSITION
129B C1C5                POP B! PUSH B ;RESTORE/SAVE WRITE FLAG (C=2 IF NEW BLOCK
129D CDDE0B              CALL WRBUFF ;WRITTEN TO DISK
12A0 C1                  POP B ;C = 2 IF A NEW BLOCK WAS ALLOCATED, 0 IF NOT
                         ;INCREMENT RECORD COUNT IF RCOUNT<=VRECORD
12A1 3AA015219E          LDA VRECORD! LXI H,RCOUNT! CMP M ;VRECORD-RCOUNT
12A6 DAB212              JC DISKWR2
                                 ;RCOUNT <= VRECORD
12A9 7734                        MOV M,A! INR M ;RCOUNT = VRECORD+1
12AB 0E02                        MVI C,2 ;MARK AS RECORD COUNT INCREMENTED
                         DISKWR2:
                         ;A HAS VRECORD, C=2 IF NEW BLOCK OR NEW RECORD#
12AD 0D0DC2BF12          DCR C! DCR C! JNZ NOUPDATE
12B2 F5                          PUSH PSW ;SAVE VRECORD VALUE
12B3 CD820D                      CALL GETMODNUM ;HL=.FCB(MODNUM), A=FCB(MODNUM)
                                 ;RESET THE FILE WRITE FLAG TO MARK AS WRITTEN FC
12B6 E67F                        ANI (NOT FWFMSK) AND 0FFH ;BIT RESET
12B8 77                          MOV M,A ;FCB(MODNUM) = FCB(MODNUM) AND 7FH
12B9 F1                          POP PSW ;RESTORE VRECORD
                         NOUPDATE:
                         ;CHECK FOR END OF EXTENT, IF FOUND ATTEMPT TO OPEN
                         ;NEXT EXTENT IN PREPARATION FOR NEXT WRITE
12BA FE7F                CPI LSTREC ;VRECORD LSTREC?
12BC C2DF12              JNZ DISKWR3 ;SKIP IF NOT
                         ;MAY BE RANDOM ACCESS WRITE, IF SO WE ARE DONE
12BF 3A9215B7CA          LDA SEQIO! ORA A! JZ DISKWR3 ;SKIP NEXT EXTENT OPEN OP
                                 ;UPDATE CURRENT FCB BEFORE GOING TO NEXT EXTENT
12C4 CD9E10C                     CALL SETFCB
12C7 CD7711                      CALL OPEN$REEL ;RMF=FALSE
                                 ;VRECORD REMAINS AT LSTREC CAUSING EOF IF
                                 ;NO MORE DIRECTORY SPACE IS AVAILABLE
12CA 214B0B7EB7                  LXI H,LRET! MOV A,M! ORA A! JNZ NOSPACE
                                         ;SPACE AVAILABLE, SET VRECORD 255
12D1 3D32A015                            DCR A! STA VRECORD ;GOES TO 00 NEXT TIME
                                 NOSPACE:
```

```
12DD 3600               MVI M,0 ;LRET = 00 FOR RETURNED VALUE
                        DISKWR3:
12DF C3F10C             JMP SETFCB ;REPLACE PARAMETERS
                        ;RET
                ;
                RSEEK:
                        ;RANDOM ACCESS SEEK OPERATION, C=0FFH IF READ MODE
                        ;FCB IS ASSUMED TO ADDRESS AN ACTIVE FILE CONTROL BLOCK
                        ;(MODNUM HAS BEEN SET TO 1100$0000B IF PREVIOUS BAD SEEK
12E2 AF329215           XRA A! STA SEQIO ;MARKED AS RANDOM ACCESS OPERATION
12E6 C5                 PUSH B ;SAVE R/W FLAG
12E7 2A490BEB           LHLD INFO! XCHG ;DE WILL HOLD BASE OF FCB
12EB 21210019               LXI H,RANREC! DAD D ;HL=.FCB(RANREC)
12EF 7EF67FF5               MOV A,M! ANI 7FH! PUSH PSW ;RECORD NUMBER
12F3 7E17                   MOV A,M! RAL ;CY=LSB OF EXTENT#
12F5 237E17E61F             INX H! MOV A,M! RAL! ANI 11111B ;A EXT#
12FA 4F                     MOV C,A ;C HOLDS EXTENT NUMBER, RECORD STACKED
12FB 7E1F1F1F1F             MOV A,M! RAR! RAR! RAR! RAR! ANI 1111B ;MOD#
1302 47                     MOV B,A ;B HOLDS MODULE#, C HOLDS EXT#
1303 F1                     POP PSW ;RECALL SOUGHT RECORD #
                            ;CHECK TO INSURE THAT HIGH BYTE OF RAN REC = 00
1304 236E                   INX H! MOV L,M ;L=HIGH BYTE (MUST BE 00)
1306 2C2D2E06               INR L! DCR L! MVI L,6 ;ZERO FLAG, L=6
                            ;PRODUCE ERROR 6  SEEK PAST PHYSICAL EOD
130A C26B13                 JNZ SEEKERR
                            ;OTHERWISE, HIGH BYTE = 0, A = SOUGHT RECORD
130D 21200019               LXI H,NXTREC! DAD D ;HL = .FCB(NXTREC)
1311 77                     MOV M,A ;SOUGHT REC# STORED AWAY
                        ;ARRIVE HERE WITH B=MOD#, C=EXT#, DE=.FCB, REC STORED
                        ;THE R/W FLAG IS STILL STACKED.  COMPARE FCB VALUES
1312 210C001979             LXI H,EXTNUM! DAD D! MOV A,C ;A=SEEK EXT#
1317 96C22613               SUB M! JNZ RANCLOSE ;TESTS FOR = EXTENTS
                            ;EXTENTS MATCH, CHECK MOD#
131B 210E001978             LXI H,MODNUM! DAD D! MOV A,B ;B=SEEK MOD#
                            ;COULD BE OVERFLOW AT EOF, PRODUCING MODULE#
                            ;OF 90H OR 10H, SO COMPARE ALL BUT FWF
1320 96E67FCA5E             SUB M! ANI 7FH! JZ SEEKOK ;SAME?
                        RANCLOSE:
1326 C5D5                   PUSH B! PUSH D ;SAVE SEEK MOD#,EXT#, .FCB
1328 CDC410                 CALL CLOSE ;CURRENT EXTENT CLOSED
132B D1C1                   POP D! POP B ;RECALL PARAMETERS AND FILL
132D 2E03                   MVI L,3 ;CANNOT CLOSE ERROR #3
132F 3A4B0B3CCA             LDA LRET! INR A! JZ BADSEEK
1336 210C001971             LXI H,EXTNUM! DAD D! MOV M,C ;FCB(EXTNUM)=EXT#
133B 210F001970             LXI H,MODNUM! DAD D! MOV M,B ;FCB(MODNUM)=MOD#
1340 CD7310                 CALL OPEN ;IS THE FILE PRESENT?
1343 3A4B0B3CC2             LDA LRET! INR A! JNZ SEEKOK ;OPEN SUCCESSFUL?
                            ;CANNOT OPEN THE FILE, READ MODE?
134A C1                     POP B ;R/W FLAG TO C (=0FFH IF READ)
134B C5                     PUSH B ;EVERYONE EXPECTS THIS ITEM STACKED
134C 2E04                   MVI L,4 ;SEEK TO UNWRITTEN EXTENT #4
134E 0C                     INR C ;BECOMES 00 IF READ OPERATION
134F CA6413                 JZ BADSEEK ;SKIP TO ERROR IF READ OPERATION
                            ;WRITE OPERATION, MAKE NEW EXTENT
1352 CD4111                 CALL MAKE
1355 2E05                   MVI L,5 ;CANNOT CREATE NEW EXTENT #5
```



```
1357 3A4B0B3CCA             LDA LRET! INR A! JZ BADSEEK ;NO DIR SPACE
                            ;FILE MAKE OPERATION SUCCESSFUL
                        SEEKOK:
135E C1                     POP B ;DISCARD R/W FLAG
135F AF324B0BC9             XRA A! STA LRET! RET ;WITH ZERO SET
                        BADSEEK:
                            ;FCB NO LONGER CONTAINS A VALID FCB, MARK
                            ;WITH 1100$0000B IN MODNUM FIELD SO THAT IT
                            ;APPEARS AS OVERFLOW WITH FILE WRITE FLAG SET
1364 E5                     PUSH H ;SAVE ERROR FLAG
1365 CD820D                 CALL GETMODNUM ;HL = .MODNUM
1368 36C0                   MVI M,1100$0000B
136A E1                     POP H ;AND DROP THROUGH
                        SEEKERR:
136B C1                     POP B ;DISCARD R/W FLAG
136C 7D324B0B               MOV A,L! STA LRET ;LRET=#, NONZERO
                            ;SETFWF RETURNS NON-ZERO ACCUMULATOR FOR ERR
1370 C3910D                 JMP SETFWF ;FLAG SET, SO SUBSEQUENT CLOSE OK
                            ;RET
                ;
                RANDISKREAD
                        ;RANDOM DISK READ OPERATION
1373 0EFF               MVI C,TRUE ;MARKED AS READ OPERATION
1375 CDE212             CALL RSEEK
1378 CCDF11             CZ DISKREAD ;IF SEEK SUCCESSFUL
137B C9                 RET
                ;
                RANDISKWRITE:
                        ;RANDOM DISK WRITE OPERATION
137C 0E00               MVI C,FALSE ;MARKED AS WRITE OPERATION
137E CDE212             CALL RSEEK
1381 CC2212             CZ DISKWRITE ;IF SEEK SUCCESSFUL
1384 C9                 RET
                ;
                COMPUTE$RR:
                        ;COMPUTE RANDOM RECORD POSITION FOR GETFILESIZE/SETRANDO
1385 EB19               XCHG! DAD D
                        ;DE=.BUF(DPTR) OR .FCB(0), HL = .F(NXTREC/RECCNT)
1387 4E0600             MOV C,M! MVI B,0 ;BC = 0000 0000 ?RRR RRRR
138A 210C00197E         LXI H,EXTNUM! DAD D! MOV A,M! RRC! ANI 80H ;A=E000 0000
1392 814F3E0088         ADD C! MOV C,A! MVI A,0! ADC B! MOV B,A
                        ;BC = 0000 000? ERRRR RRRR
1398 7E0FE60F80         MOV A,M! RRC! ANI 0FH! ADD B! MOV B,A
                        ;BC = 000? EEEE ERRRR RRRR
139E 210E00197E         LXI H,MODNUM! DAD D! MOV A,M ;A=XXX? MMMM
13A3 87878787           ADD A! ADD A! ADD A! ADD A ;CY=? A=MMMM 0000
13A7 F58047             PUSH PSW! ADD B! MOV B,A
                        ;CY=?, BC = MMMM EEEE ERRR RRRR
13AA F5                 PUSH PSW ;POSSIBLE SECOND CARRY
13AB E1                 POP H ;CY = LSB OF L
13AC 7D                 MOV A,L ;CY = LSB OF A
13AD E1                 POP H ;CY = LSB OF L
13AE B5                 ORA L ;CY/CY = LSB OF A
13AF E601               ANI 1 ;A = 0000 000? POSSIBLE CARRY-OUT
13B1 C9                 RET
                ;
```

```
                GETFILESIZE
                        ;COMPUTE LOGICAL FILE SIZE FOR CURRENT FCB
13B2 0E0C               MVI C,EXTNUM
13B4 CDA20F             CALL SEARCH
                        ;ZERO THE RECEIVING RANREC FIELD
13B7 2A490B1121         LHLD INFO! LXI D,RANREC! DAD D! PUSH H ;SAVE POSITION
13BF 7223722372         MOV M,D! INX H! MOV M,D! INX H! MOV M,D; 00 00 00
                        GETSIZE:
13C4 CD1A0E                     CALL END$OF$DIR
13C7 CAEC13                     JZ SETSIZE
                                ;CURRENT FCB ADDRESSED BY DPTR
13CA CD790D110F                 CALL GETDPTRA! LXI D,RECCNT ;READY FOR COMPUTE S
13D0 CD8513                     CALL COMPUTE$RR
                                ;A=0000 0007 BC = MMMM EEEE ERRR RRRR
                                ;COMPARE WITH MEMORY, LARGER?
13D3 E1E5                       POP H! PUSH H ;RECALL, REPLACE .FCB(RANREC`
13D5 5F                         MOV E,A ;SAVE CY
13D6 799623                     MOV A,C! SUB M! INX H ;LS BYTE
13D9 789E23                     MOV A,B! SBB M! INX H ;MIDDLE BYTE
13DC 7B9F                       MOV A,E! SBB M ;CARRY IF .FCB(RANREC) > DIRECTOR
13DE DAE613                     JC GETNEXTSIZE ;FOR ANOTHER TRY
                                ;FCB IS LESS OR EQUAL, FILL FROM DIRECTORY
13E1 732B702B71                 MOV M,E! DCX H! MOV M,B! DCX H! MOV M,C
                        GETNEXTSIZE:
13E6 CD380F                     CALL SEARCHN
13E9 C3C413                     JMP GETSIZE
                        SETSIZE:
13EC E1                 POP H ;DISCARD .FCB(RANREC`
13ED C9                 RET
                ;
                SETRANDOM:
                        ;SET RANDOM RECORD FROM THE CURRENT FILE CONTROL BLOCK
13EE 2A490B1120         LHLD INFO! LXI D,NXTREC ;READY PARAMS FOR COMPUTESIZE
13F4 CD8513             CALL COMPUTE$RR ;DE=INFO, A=CY, BC=MMMM EEEE ERRR RRRR
13F7 21210019           LXI H,RANREC! DAD D ;HL = .FCB(RANREC)
13FB 7123702377         MOV M,C! INX H! MOV M,B! INX H! MOV M,A ;TO RANREC
1400 C9                 RET
                ;
                SELECT:
                        ;SELECT DISK INFO FOR SUBSEQUENT INPUT OR OUTPUT OPS
1401 2A6C153A48         LHLD DLOG! LDA CURDSK! MOV C,A! CALL HLROTR
1408 E5EB               PUSH H! XCHG ;SAVE IT FOR TEST BELOW, SEND TO SELDSK
140D CD740BE1           CALL SELECTDISK! POP H ;RECALL DLOG VECTOR
1411 CC580B             CZ SEL$ERROR ;RETURNS TRUE IF SELECT OK
                        ;IS THE DISK LOGGED IN?
1414 7D1FD8             MOV A,L! RAR! RC ;RETURN IF BIT IS SET
                        ;DISK NOT LOGGED IN, SET BIT AND INITIALIZE
1417 2A6C154D44         LHLD DLOG! MOV C,L! MOV B,H ;CALL READY
141C CD250D226C         CALL SET$CDISK! SHLD DLOG ;DLOG=SET$CDISK(DLOG)
1422 C3C20E             JMP INITIALIZE
                        ;RET
                ;
                CURSELECT:
1425 3A93152148         LDA LINFO! LXI H,CURDSK! CMP M! RZ ;SKIP IF LINFO=CURDSK
142D 77                 MOV M,A ;CURDSK=INFO
142F C30114             JMP SELECT
```



```
                        ;RET
                ;
                RESELECT:
                        ;CHECK CURRENT FCB TO SEE IF RESELECTION NECESSARY
1431 3EFF329B15         MVI A,TRUE! STA RESEL ;MARK POSSIBLE RESELECT
1436 2A490B7E           LHLD INFO! MOV A,M ;DRIVE SELECT CODE
143A E61F               ANI 1$11116 ;NON ZERO IS AUTO DRIVE SELECT
143C 3D                 DCR A ;DRIVE CODE NORMALIZED TO 0..30, OR 255
143D 329315             STA LINFO ;SAVE DRIVE CODE
1440 FE1ED25514         CPI 30! JNC NOSELECT
                                ;AUTO SELECT FUNCTION, SAVE CURDSK
1445 3A480B329C                 LDA CURDSK! STA OLDDSK ;OLDDSK=CURDSK
144B 7E329D15                   MOV A,M! STA FCBDSK ;SAVE DRIVE CODE
144F E6F077                     ANI 1110$0000B! MOV M,A ;PRESERVE HI BITS
1452 CD2514                     CALL CURSELECT
                        NOSELECT:
                                ;SET USER CODE
1455 3A470B                     LDA USRCODE ;0...31
1458 2A490BB677                 LHLD INFO! ORA M! MOV M,A
145D C9                         RET
                ;
                ;       INDIVIDUAL FUNCTION HANDLERS
                FUNC12:
                        ;RETURN VERSION NUMBER
145E 3E20324B0B         MVI A,DVERS! STA LRET ;LRET = DVERS (HIGH = 00)
1463 C9                 RET ;JMP GOBACK
                ;
                FUNC13:
                        ;RESET DISK SYSTEM - INITIALIZE TO DISK 0
1464 210000226A         LXI H,0! SHLD RODSK! SHLD DLOG
146D AF32480B           XRA A! STA CURDSK ;NOTE THAT USRCODE REMAINS UNCHANGED
1471 218000226E         LXI H,TBUFF! SHLD DMAAD ;DMAAD = TBUFF
1477 CDEB0D             CALL SETDATA ;TO DATA DMA ADDRESS
147A C30114             JMP SELECT
                        ;RET ;JMP GOBACK
                ;
                FUNC14:
                        ;SELECT DISK INFO
147D C32514             JMP CURSELECT
                        ;RET ;JMP GOBACK
                ;
                FUNC15:
                        ;OPEN FILE
1480 CD8B0D             CALL CLRMODNUM ;CLEAR THE MODULE NUMBER
1483 CD3114             CALL RESELECT
1486 C37310             JMP OPEN
                        ;RET ;JMP GOBACK
                ;
                FUNC16:
                        ;CLOSE FILE
1489 CD3114             CALL RESELECT
148C C3C410             JMP CLOSE
                        ;RET ;JMP GOBACK
                ;
                FUNC17:
                        ;SEARCH FOR FIRST OCCURRENCE OF A FILE
```

```
148F 0E00               MVI C,0 ;LENGTH ASSUMING '?' TRUE
1491 2A490B7EFE         LHLD INFO! MOV A,M! CPI '?' ;NO RESELECT IF ?
1497 CAA214             JZ QSELECT ;SKIP RESELECT IF SO
                                ;NORMAL SEARCH
149A CD8B0D                     CALL CLRMODNUM ;MODULE NUMBER ZEROED
149D CD3114                     CALL RESELECT
14A0 0E0F                       MVI C,NAMLEN
                        QSELECT:
14A2 CDA80F             CALL SEARCH
14A5 C30E0E             JMP DIR$TO$USER ;COPY DIRECTORY ENTRY TO USER
                        ;RET ;JMP GOBACK
                ;
                FUNC18:
                        ;SEARCH FOR NEXT OCCURRENCE OF A FILE NAME
14A8 2A96152249         LHLD SEARCHA! SHLD INFO
14AE CD3114CD38         CALL RESELECT! CALL SEARCHN
14B4 C30E0E             JMP DIR$TO$USER ;COPY DIRECTORY ENTRY TO USER
                        ;RET ;JMP GOBACK
                ;
                FUNC19:
                        ;DELETE A FILE
14B7 CD3114             CALL RESELECT
14BA CDC00F             CALL DELETE
14BD C3200F             JMP COPY$DIRLOC
                        ;RET ;JMP GOBACK
                ;
                FUNC20:
                        ;READ A FILE
14C0 CD3114             CALL RESELECT
14C3 CDDA11             CALL SEQDISKREAD
14C6 C9                 RET ;JMP GOBACK
                ;
                FUNC21:
                        ;WRITE A FILE
14C7 CD3114             CALL RESELECT
14CA CD1D12             CALL SEQDISKWRITE
14CD C9                 RET ;JMP GOBACK
                ;
                FUNC22:
                        ;MAKE A FILE
14CE CD8B0D             CALL CLRMODNUM
14D1 CD3114             CALL RESELECT
14D4 C34111             JMP MAKE
                        ;RET ;JMP GOBACK
                ;
                FUNC23:
                        ;RENAME A FILE
14D7 CD3114             CALL RESELECT
14DA CD3810             CALL RENAME
14DD C3200F             JMP COPY$DIRLOC
                        ;RET ;JMP GOBACK
                ;
                FUNC24:
                        ;RETURN THE LOGIN VECTOR
14E0 2A6C15224B         LHLD DLOG! SHLD ARET
14E6 C9                 RET ;JMP GOBACK
```



```
                ;
                FUNC25:
                        ;RETURN SELECTED DISK NUMBER
14E7 3A4F0B324B         LDA CURDSK! STA LRET
14ED C9                 RET ;JMP GOBACK
                ;
                FUNC26:
                        ;SET THE SUBSEQUENT DMA ADDRESS TO INFO
14EE 2A490B226E         LHLD INFO! SHLD DMAAD ;DMAAD = INFO
14F4 C3EB0D             JMP SETDATA ;TO DATA DMA ADDRESS
                        ;RET ;JMP GOBACK
                ;
                FUNC27:
                        ;RETURN THE LOGIN VECTOR ADDRESS
14F7 2A7C15224B         LHLD ALLOCA! SHLD ARET
14FD C9                 RET ;JMP GOBACK
                ;
                FUNC28:
                        ;WRITE PROTECT CURRENT DISK
14FE C34E0D             JMP SET$RO
                        ;RET ;JMP GOBACK
                ;
                FUNC29:
                        ;RETURN R/O BIT VECTOR
1501 2A6A15224B         LHLD RODSK! SHLD ARET
1507 C9                 RET ;JMP GOBACK
                ;
                FUNC30:
                        ;SET FILE INDICATORS
150E CD3114             CALL RESELECT
150B CD5D10             CALL INDICATORS
150E C3200F             JMP COPY$DIRLOC ;LRET=DIRLOC
                        ;RET ;JMP GOBACK
                ;
                FUNC31:
                        ;RETURN ADDRESS OF DISK PARAMETER BLOCK
1511 2A7815224B         LHLD DPBADDR! SHLD ARET
1517 C9                 RET ;JMP GOBACK
                ;
                FUNC32:
                        ;SET USER CODE
1518 3A9315FEFF         LDA LINFO! CPI 0FFH! JNZ SETUSRCODE
                                ;INTERROGATE USER CODE INSTEAD
1520 3A470B324B                 LDA USRCODE! STA LRET ;LRET=USRCODE
1526 C9                         RET ;JMP GOBACK
                        SETUSRCODE:
1527 E61F32470B                 ANI 1FH! STA USRCODE
152C C9                         RET ;JMP GOBACK
                ;
                FUNC33:
                        ;RANDOM DISK READ OPERATION
152D CD3114             CALL RESELECT
1530 C37313             JMP RANDISKREAD ;TO PERFORM THE DISK READ
                        ;RET ;JMP GOBACK
                ;
                FUNC34:
```

```
                        ;RANDOM DISK WRITE OPERATION
1533 CD3114             CALL RESELECT
1536 C37C13             JMP RANDISKWRITE ;TO PERFORM THE DISK WRITE
                        ;RET ;JMP GOBACK
                ;
                FUNC35:
                        ;RETURN FILE SIZE (0-65536)
1539 CD3114             CALL RESELECT
153C C3B213             JMP GETFILESIZE
                        ;RET ;JMP GOBACK
                ;
                FUNC36:
                        ;SET RANDOM RECORD
153F C3EE13             JMP SETRANDOM
                        ;RET ;JMP GOBACK
                ;
                GOBACK:
                        ;ARRIVE HERE AT END OF PROCESSING TO RETURN TO USER
1542 3A9B15B7CA         LDA RESEL! ORA A! JZ RETMON
                                ;RESELECTION MAY HAVE TAKEN PLACE
1549 2A490B3600                 LHLD INFO! MVI M,0 ;FCB(0)=0
154F 3A9D15B7CA                 LDA FCBDSK! ORA A! JZ RETMON
                                ;RESTORE DISK NUMBER
1555 77                         MOV M,A ;FCB(0)=FCBDSK
1556 3A9C153293                 LDA OLDDSK! STA LINFO! CALL CURSELECT
                ;
                ;       RETURN FROM THE DISK MONITOR
                RETMON:
155F 2A150BF9           LHLD ENTSP! SPHL ;USER STACK RESTORED
1563 2A4E0B7D44         LHLD ARET! MOV A,L! MOV B,H ;BA = HL = ARET
1568 C9                 RET
                ;
                ;       DATA AREAS
                ;
                ;       INITIALIZED DATA
1569 E5         EFCB:   DB      EMPTY   ;0E5 AVAILABLE DIR ENTRY
156A 0000       RODSK:  DW      0       ;READ ONLY DISK VECTOR
156C 0000       DLOG:   DW      0       ;LOGGED-IN DISKS
156E 8000       DMAAD   DW      TBUFF   ;INITIAL DMA ADDRESS
                ;
                ;       CURTRKA - ALLOCA ARE SET UPON DISK SELECT
                ;       (DATA MUST BE ADJACENT, DO NOT INSERT VARIABLES)
                ;       (ADDRESS OF TRANSLATE VECTOR, NOT USED)
1570            CDRMAXA:DS      WORD    ;POINTER TO CUR DIR MAX VALUE
1572            CURTRKA:DS      WORD    ;CURRENT TRACK ADDRESS
1574            CURRECA:DS      WORD    ;CURRENT RECORD ADDRESS
1576            BUFFA:  DS      WORD    ;POINTER TO DIRECTORY DMA ADDRESS
1578            DPBADDR:DS      WORD    ;CURRENT DISK PARAMETER BLOCK ADDRESS
157A            CHECKA: DS      WORD    ;CURRENT CHECKSUM VECTOR ADDRESS
157C            ALLOCA: DS      WORD    ;CURRENT ALLOCATION VECTOR ADDRESS
0008 =          ADDLIST EQU     $-BUFFA ;ADDRESS LIST SIZE
                ;
                ;       SECTPT - OFFSET OBTAINED FROM DISK PARM BLOCK AT DPBADDR
                ;       (DATA MUST BE ADJACENT, DO NOT INSERT VARIABLES)
157E            SECTPT: DS      WORD    ;SECTORS PER TRACK
1580            BLKSHF: DS      BYTE    ;BLOCK SHIFT FACTOR
```



```
1581            BLKMSK: DS      BYTE    ;BLOCK MASK
1582            EXTMSK: DS      BYTE    ;EXTENT MASK
1583            MAXALL: DS      WORD    ;MAXIMUM ALLOCATION NUMBER
1585            DIRMAX: DS      WORD    ;LARGEST DIRECTORY NUMBER
1587            DIRBLK: DS      WORD    ;RESERVED ALLOCATION BITS FOR DIRECTORY
1589            CHKSIZ: DS      WORD    ;SIZE OF CHECKSUM VECTOR
158B            OFFSET: DS      WORD    ;OFFSET TRACKS AT BEGINNING
000F =          DPBLIST EQU     $-SECTPT        ;SIZE OF AREA
                ;
                ;       LOCAL VARIABLES
158D            TRANV:  DS      WORD    ;ADDRESS OF TRANSLATE VECTOR
                FCB$COPIED
158F                    DS      BYTE    ;SET TRUE IF COPY$FCB CALLED
1590            RMF:    DS      BYTE    ;READ MODE FLAG FOR OPEN$REEL
1591            DIRLOC: DS      BYTE    ;DIRECTORY FLAG IN RENAME, ETC.
1592            SEQIO   DS      BYTE    ;1 IF SEQUENTIAL I/O
1593            LINFO:  DS      BYTE    ;LOW(INFO)
1594            DMINX:  DS      BYTE    ;LOCAL FOR DISKWRITE
1595            SEARCHL:DS      BYTE    ;SEARCH LENGTH
1596            SEARCHA:DS      WORD    ;SEARCH ADDRESS
1598            TINFO:  DS      WORD    ;TEMP FOR INFO IN "MAKE"
159A            SINGLE: DS      BYTE    ;SET TRUE IF SINGLE BYTE ALLOCATION MAP
159B            RESEL:  DS      BYTE    ;RESELECTION FLAG
159C            OLDDSK: DS      BYTE    ;DISK ON ENTRY TO BDOS
159D            FCBDSK: DS      BYTE    ;DISK NAMED IN FCB
159E            RCOUNT: DS      BYTE    ;RECORD COUNT IN CURRENT FCB
159F            EXTVAL: DS      BYTE    ;EXTENT NUMBER AND EXTMSK
15A0            VRECORD:DS      WORD    ;CURRENT VIRTUAL RECORD
15A2            ARECORD:DS      WORD    ;CURRENT ACTUAL RECORD
                ;
                ;       LOCAL VARIABLES FOR DIRECTORY ACCESS
15A4            DPTR:   DS      BYTE    ;DIRECTORY POINTER 0,1,2,3
15A5            DCNT:   DS      WORD    ;DIRECTORY COUNTER 0,1,...,DIRMAX
15A7            DREC:   DS      WORD    ;DIRECTORY RECORD 0,1,...,DIRMAX/4
                ;
1600 =          BIOS    EQU     ($ AND 0FF00H)+100H     ;NEXT MODULE
15A9                    END
```

```
0D74 ADDH       0008 ADDLIST    157C ALLOCA     0C96 ALLOCATED  1288 ALLOCWD
0000 ARECH      0001 ARECL      15A2 ARECORD    0B4B ARET       0C9C ATRAN
0CA2 ATRAN0     0AE1 BACKSP     09EE BACKUP     0A96 BACKX      1364 BADSEEK
0006 BDOSA      0811 BDOSE      1600 BIOS       15E1 BLKMSK     1580 BLKSHF
126C BLOCKOK    1576 BUFFA      10FD BUFFNZERO  0001 BYTE       1570 CDRMAXA
157A CHECKA     0D67 CHECKRODIR 0D5D CHECKROFILE                0DBD CHECKSUM
0D6D CHECKWRITE 1589 CHKSIZ     10C4 CLOSE      0DEB CLRMODNUM  0B12 COLUMN
0D9E COMPCDR    0B10 COMPCOL    0F27 COMPEXT    09A7 COMPOUT    0D11 COMPUTECS
0D17 COMPUTECS0 1385 COMPUTERR  0987 CONB0      09EA CONB1      0968 CONBRK
094B CONECH     0940 CONIN      098D CONOUT     101C COPYDIR    0F20 COPYDIRLOC
1031 COPYFCB    000D CR         0002 CRECH      0003 CRECL      0A11 CRLF
09F9 CRLFP      0A01 CRLFP0     0003 CTLC       0005 CTLE       0008 CTLH
005E CTL        09C4 CTLOUT     0010 CTLP       0012 CTLR       0013 CTLS
0015 CTLU       0018 CTLX       001A CTLZ       0004 CTRKH      0005 CTRKL
0B48 CURDSK     1574 CURRECA    1425 CURSELECT  1572 CURTRKA     15A5 DCNT
0FC0 DELETE     0FC8 DELETE0    1587 DIRBLK     08AF DIRINP     1591 DIRLOC
1585 DIRMAX     0004 DIRREC     0E0E DIRTOUSER  121A DISKEOF     000C DISKF
11DF DISKREAD   123F DISKWR0    1292 DISKWR1    12B2 DISKWR2    12DF DISKWR3
1222 DISKWRITE  1290 DISKWRU    156C DLOG       156E DMAAD      1594 DMINX
0C57 DMPOS0     0C65 DMPOS1     0C6E DMPOS2     0C50 DMPOSITION 1119 DMSET
1578 DPBADDR    000F DPBLIST    15A4 DPTR       15A7 DREC       0921 DSKERR
0010 DSKMAP     0915 DSKMSG     0003 DSKMSK     0002 DSKSHF     0020 DVERS
0959 ECHOC      1569 EFCB       00E5 EMPTY      FFFF ENDDIR     1133 ENDMERGE
0E1A ENDOFDIR   0FBE ENDSEARCH  0B15 ENTSP      08FC ERRFLG     15E2 EXTMSK
000C EXTNUM     159F EXTVAL     0000 FALSE      158F FCBCOPIED  159D FCBDSK
0020 FCBLEN     10F7 FCBNZERO   0005 FCBSHF     088E FUNC1      08D0 FUNC10
08D4 FUNC11     145E FUNC12     1464 FUNC13     147D FUNC14     1480 FUNC15
1489 FUNC16     148F FUNC17     14A8 FUNC18     14B7 FUNC19     14C0 FUNC20
14C7 FUNC21     14CE FUNC22     14D7 FUNC23     14E0 FUNC24     14E7 FUNC25
14EE FUNC26     14F7 FUNC27     14FE FUNC28     0894 FUNC2      1501 FUNC29
1508 FUNC30     1511 FUNC31     1518 FUNC32     152D FUNC33     1533 FUNC34
1539 FUNC35     153F FUNC36     0898 FUNC3      089E FUNC4      08A2 FUNC5
08A6 FUNC6      08BC FUNC7      08C2 FUNC8      08C7 FUNC9      0844 FUNCTAB
0080 FWFMSK     0E62 GETALLOCBIT                0FE2 GETBLOCK   0C70 GETDM
0C83 GETDMD     0D79 GETDPTRA   0CB5 GETEXTA    0CBD GETFCBA    0CCA GETFCB
13B2 GETFILESIZE                0DB2 GETMODNUM  13E6 GETNEXTSIZE
13C4 GETSIZE    1542 GOBACK     0B4D GOERR      0D1E HLROTL     0D1F HLROTL0
0CF3 HLROTR     0CF4 HLROTR0    0BBC HOME       0CE9 INDEX      1062 INDIC0
105D INDICATORS 0B49 INFO       0ED0 INITIAL0   0EF1 INITIAL2   0DE3 INITIALCS
0EC2 INITIALIZE 000A INVIS      0003 IOLOC      0B14 KBCHAR     0FE4 LEFTTST
000A LF         0AB8 LINELEN    1593 LINFO      0B13 LISTCP     0B4E LRET
0B47 LSTACK     001F LSTFCB     007F LSTREC     1141 MAKE       1163 MAKE0
1583 MAXALL     001F MAXEXT     000F MAXMOD     10E9 MERGE0     1104 MERGED
113C MERGERR    10B6 MERGEZERO  000E MODNUM     0B6A MOVE       0B6B MOVE0
000F NAMLEN     0DBB NEWCHECKSUM                0025 NFUNCS     125E NOPBLOCK
1455 NOSELECT   12DD NOSPACE    09BE NOTBACKSP  0B05 NOTC       0A7F NOTE
0A5E NOTH       0A90 NOTP       0A6E NOTRUB     0AEE NOTR       0AB3 NOTU
0AA7 NOTX       12BF NOUPDATE   0D38 NOWRITE    0020 NXTREC     0000 OFF
158B OFFSET     159C OLDDSK     FFFF ON         1073 OPEN       107C OPENCOPY
11A0 OPENMOD    10AD OPENRCNT   1177 OPENREEL   11AB OPENREEL0  11C9 OPENREEL1
11CC OPENREEL2  11D4 OPENRERR   09E9 PCTLH      0F15 PDOLLAR    0E09 PERERR
0B52 PERERROR   0925 PERMSG     08DB PERSUB     0A1B PRINT      14A2 QSELECT
1326 RANCLOSE   1373 RANDISKREAD                137C RANDISKWRITE
0021 RANREC     159E RCOUNT     0BD6 RLBUFF     0E05 RDDIR      0AF1 RDECH1
0AEE RDECHO     0A29 READ       0E2A READDIR    0E3F READDIR0   0F46 READDIR1
0B09 READEN     0A39 READN0     0A37 READNX     0000 REBOOT     000F RECCNT
1204 RECORDOK   0080 RECSIZ     1038 RENAME     1049 RENAME0    0AC0 REP0
0AD2 REP1       1431 RESELECT   159B RESEL      1010 RETBLOCK   1018 RETBLOCK0
155F RETMON     0BB8 RETSELECT  0FF5 RIGHTTST   1590 RMF        080D RODERR
0B5E RODERROR   093C RODMSG     156A RODSK      08ED RODSUB     080F ROFERR
0B64 ROFERROR   0009 ROFILE     0937 ROFMSG     08F3 ROFSUB     0E63 ROTL
0E5B ROTR       12E2 RSEEK      007F RUBOUT     0E94 SCANDM     0E9E SCANDM0
0EB1 SCANDM1    0EB7 SCANDM2    1596 SEARCHA    0F7E SEARCHEXT  0FA8 SEARCH
0F9F SEARCHFIN  1595 SEARCHL    0F5E SEARCHLOOP 0F3E SEARCHN    0F55 SEARCHNEXT
0F87 SEARCHOK   157E SECTPT     0EE6 SEEK       0EF9 SEEK0      0C0F SEEK1
0C21 SEEK2      102B SEEKCOPY   0D00 SEEKDIR    136B SEEKERR    135E SEEKOK
0B74 SELECTDISK 1401 SELECT     080B SELERR     0B58 SELERROR   0930 SELMSG
08E7 SELSUB     11DA SEQDISKREAD                121D SEQDISKWRITE
1592 SEQIO      0E09 SETALLOCBIT                0D25 SETCDISK   0DAB SETCDR
0DEB SETDATA    0DF1 SETDIR     0DE5 SETDMA     0E23 SETENDDIR  0CE1 SETFCB
0D91 SETFWF     0D98 SETLRET1   13EE SETRANDOM  0D46 SETRO      13EC SETSIZE
1527 SETUSRCODE 159A SINGLE     0018 SSIZE      08D7 STARET     0B11 STRTCOL
0DB4 SUBDH      0009 TAB        09DB TAB0       09D5 TABOUT     0000 TBUFF
0004 TCRECH     0005 TCRECL     0000 TEST       005C TFCB       1598 TINFO
158D TRANV      00FF TRUE       000D UBYTES     0B47 USRCODE    15A0 VRECORD
08F6 WAITERR    0002 WORD       0BDE WRBUFF     0DF7 WRDIR
```

CP/M 2.0
COPYRIGHT © 1978
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA. 93950
SER. # _____

```
                ;       MDS-800 I/O DRIVERS FOR CP/M 2.0
                ;       (FOUR DRIVE SINGLE DENSITY VERSION)
                ;
                ;       VERSION 2.0 AUGUST, 1979
                ;
0014 =          VERS    EQU     20      ;VERSION 2.0
                ;
                ;       COPYRIGHT (C) 1979
                ;       DIGITAL RESEARCH
                ;       BOX 579, PACIFIC GROVE
                ;       CALIFORNIA, 93950
                ;
                ;
FFFF =          TRUE    EQU     0FFFFH  ;VALUE OF "TRUE"
0000 =          FALSE   EQU     NOT TRUE        ;"FALSE"
0000 =          TEST    EQU     FALSE   ;TRUE IF TEST BIOS
                ;
                        IF      TEST
                BIAS    EQU     03400H  ;BASE OF CCP IN TEST SYSTEM
                        ENDIF
                        IF      NOT TEST
0000 =          BIAS    EQU     0000H   ;GENERATE RELOCATABLE CP/M SYSTEM
                        ENDIF
                ;
1600 =          PATCH   EQU     1600H
                ;
1600                    ORG     PATCH
0000 =          CPMB    EQU     $-PATCH ;BASE OF CPM CONSOLE PROCESSOR
0806 =          BDOS    EQU     806H+CPMB       ;BASIC DOS (RESIDENT PORTION)
1600 =          CPML    EQU     $-CPMB  ;LENGTH (IN BYTES) OF CPM SYSTEM
002C =          NSECTS  EQU     CPML/128        ;NUMBER OF SECTORS TO LOAD
0002 =          OFFSET  EQU     2       ;NUMBER OF DISK TRACKS USED BY CP/M
0004 =          CDISK   EQU     0004H   ;ADDRESS OF LAST LOGGED DISK ON WARM STA
0080 =          BUFF    EQU     0080H   ;DEFAULT BUFFER ADDRESS
000A =          RETRY   EQU     10      ;MAX RETRIES ON DISK I/O BEFORE ERROR
                ;
                ;       PERFORM FOLLOWING FUNCTIONS
                ;       BOOT    COLD START
                ;       WBOOT   WARM START (SAVE I/O BYTE)
                ;       (BOOT AND WBOOT ARE THE SAME FOR MDS)
                ;       CONST   CONSOLE STATUS
                ;               REG-A = 00 IF NO CHARACTER READY
                ;               REG-A = FF IF CHARACTER READY
                ;       CONIN   CONSOLE CHARACTER IN (RESULT IN REG-A)
                ;       CONOUT  CONSOLE CHARACTER OUT (CHAR IN REG-C)
                ;       LIST    LIST OUT (CHAR IN REG-C)
                ;       PUNCH   PUNCH OUT (CHAR IN REG-C)
                ;       READER  PAPER TAPE READER IN (RESULT TO REG-A)
                ;       HOME    MOVE TO TRACK 00
                ;
                ;       (THE FOLLOWING CALLS SET-UP THE IO PARAMETER BLOCK FOR T
                ;       MDS, WHICH IS USED TO PERFORM SUBSEQUENT READS AND WRITE
                ;       SELDSK  SELECT DISK GIVEN BY REG-C (0,1,2...)
                ;       SETTRK  SET TRACK ADDRESS (0,...76) FOR SUBSEQUENT READ/
                ;       SETSEC  SET SECTOR ADDRESS (1,...,26) FOR SUBSEQUENT REA
                ;       SETDMA  SET SUBSEQUENT DMA ADDRESS (INITIALLY 80H)
                ;
                ;       (READ AND WRITE ASSUME PREVIOUS CALLS TO SET UP THE IO P
                ;       READ    READ TRACK/SECTOR TO PRESET DMA ADDRESS
                ;       WRITE   WRITE TRACK/SECTOR FROM PRESET DMA ADDRESS
                ;
                ;       JUMP VECTOR FOR INDIVIUAL ROUTINES
1600 C3B316             JMP     BOOT
1603 C3C316     WBOOTE: JMP     WBOOT
1606 C36117             JMP     CONST
1609 C36417             JMP     CONIN
160C C36A17             JMP     CONOUT
160F C36D17             JMP     LIST
1612 C37217             JMP     PUNCH
1615 C37517             JMP     READER
1618 C37817             JMP     HOME
161B C37D17             JMP     SELDSK
161E C3A717             JMP     SETTRK
1621 C3AC17             JMP     SETSEC
1624 C3BB17             JMP     SETDMA
1627 C3C117             JMP     READ
162A C3CA17             JMP     WRITE
162D C37017             JMP     LISTST  ;LIST STATUS
1630 C3B117             JMP     SECTRAN
                ;
                        MACLIB  DISKDEF ;LOAD THE DISK DEFINITION LIBRARY
                        DISKS   4       ;FOUR DISKS
1633+=          DPBASE  EQU     $       ;BASE OF DISK PARAMETER BLOCKS
1633+82160000   DPE0:   DW      XLT0,0000H      ;TRANSLATE TABLE
1637+00000000           DW      0000H,0000H     ;SCRATCH AREA
163B+6E187316           DW      DIRBUF,DPB0     ;DIR BUFF,PARM BLOCK
163F+0D19EF18           DW      CSV0,ALV0       ;CHECK, ALLOC VECTORS
1643+82160000   DPE1:   DW      XLT1,0000H      ;TRANSLATE TABLE
1647+00000000           DW      0000H,0000H     ;SCRATCH AREA
164B+6E187316           DW      DIRBUF,DPB1     ;DIR BUFF,PARM BLOCK
164F+3C191D19           DW      CSV1,ALV1       ;CHECK, ALLOC VECTORS
1653+82160000   DPE2:   DW      XLT2,0000H      ;TRANSLATE TABLE
1657+00000000           DW      0000H,0000H     ;SCRATCH AREA
165B+6E187316           DW      DIRBUF,DPB2     ;DIR BUFF,PARM BLOCK
165F+6B194C19           DW      CSV2,ALV2       ;CHECK, ALLOC VECTORS
1663+82160000   DPE3:   DW      XLT3,0000H      ;TRANSLATE TABLE
1667+00000000           DW      0000H,0000H     ;SCRATCH AREA
166B+6E187316           DW      DIRBUF,DPB3     ;DIR BUFF,PARM BLOCK
166F+9A197B19           DW      CSV3,ALV3       ;CHECK, ALLOC VECTORS
                        DISKDEF 0,1,26,6,1024,243,64,64,OFFSET
1673+=          DPB0    EQU     $               ;DISK PARM BLOCK
1673+1AC0               DW      26              ;SEC PER TRACK
1675+03                 DB      3               ;BLOCK SHIFT
1676+07                 DB      7               ;BLOCK MASK
1677+00                 DB      0               ;EXTNT MASK
1678+F200               DW      242             ;DISK SIZE-1
167A+3F00               DW      63              ;DIRECTORY MAX
167C+C0                 DB      192             ;ALLOC0
167D+00                 DB      0               ;ALLOC1
167E+1000               DW      16              ;CHECK SIZE
1680+0200               DW      2               ;OFFSET
1682+=          XLT0    EQU     $               ;TRANSLATE TABLE
```

```
1682+01                 DB      1
1683+07                 DB      7
1684+0D                 DB      13
1685+13                 DB      19
1686+19                 DB      25
1687+05                 DB      5
1688+0B                 DB      11
1689+11                 DB      17
168A+17                 DB      23
168B+03                 DB      3
168C+09                 DB      9
168D+0F                 DB      15
168E+15                 DB      21
168F+02                 DB      2
1690+08                 DB      8
1691+0E                 DB      14
1692+14                 DB      20
1693+1A                 DB      26
1694+06                 DB      6
1695+0C                 DB      12
1696+12                 DB      18
1697+18                 DB      24
1698+04                 DB      4
1699+0A                 DB      10
169A+10                 DB      16
169B+16                 DB      22
                        DISKDEF 1,0
1673+=          DPB1    EQU     DPB0    ;EQUIVALENT PARAMETERS
001F+=          ALS1    EQU     ALS0    ;SAME ALLOCATION VECTOR SIZE
0010+=          CSS1    EQU     CSS0    ;SAME CHECKSUM VECTOR SIZE
1682+=          XLT1    EQU     XLT0    ;SAME TRANSLATE TABLE
                        DISKDEF 2,0
1673+=          DPB2    EQU     DPB0    ;EQUIVALENT PARAMETERS
001F+=          ALS2    EQU     ALS0    ;SAME ALLOCATION VECTOR SIZE
0010+=          CSS2    EQU     CSS0    ;SAME CHECKSUM VECTOR SIZE
1682+=          XLT2    EQU     XLT0    ;SAME TRANSLATE TABLE
                        DISKDEF 3,0
1673+=          DPB3    EQU     DPB0    ;EQUIVALENT PARAMETERS
001F+=          ALS3    EQU     ALS0    ;SAME ALLOCATION VECTOR SIZE
0010+=          CSS3    EQU     CSS0    ;SAME CHECKSUM VECTOR SIZE
1682+=          XLT3    EQU     XLT0    ;SAME TRANSLATE TABLE
                ;       ENDEF OCCURS AT END OF ASSEMBLY
                ;
                ;       END OF CONTROLLER - INDEPENDENT CODE.  THE REMAINING SUBR
                ;       ARE TAILORED TO THE PARTICULAR OPERATING ENVIRONMENT, AN
                ;       BE ALTERED FOR ANY SYSTEM WHICH DIFFERS FROM THE INTEL M
                ;
                ;       THE FOLLOWING CODE ASSUMES THE MDS MONITOR EXISTS AT 0F8
                ;       AND USES THE I/O SUBROUTINES WITHIN THE MONITOR
                ;
                ;       WE ALSO ASSUME THE MDS SYSTEM HAS FOUR DISK DRIVES
00FD =          REVRT   EQU     0FDH    ;INTERRUPT REVERT PORT
00FC =          INTC    EQU     0FCH    ;INTERRUPT MASK PORT
00F3 =          ICON    EQU     0F3H    ;INTERRUPT CONTROL PORT
007E =          INTE    EQU     0111$1110B      ;ENABLE RST 0(WARM BOOT), RST 7
                ;
```

CP/M 2.0
COPYRIGHT © 1978
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA. 93950
SER. # _____

```
                ;       MDS MONITOR EQUATES
F800 =          MON80   EQU     0F800H  ;MDS MONITOR
FF0F =          RMON80  EQU     0FF0FH  ;RESTART MON80 (BOOT ERROR)
F803 =          CI      EQU     0F803H  ;CONSOLE CHARACTER TO REG-A
F806 =          RI      EQU     0F806H  ;READER IN TO REG-A
F809 =          CO      EQU     0F809H  ;CONSOLE CHAR FROM C TO CONSOLE OUT
F80C =          PO      EQU     0F80CH  ;PUNCH CHAR FROM C TO PUNCH DEVICE
FF0F =          LO      EQU     0F80FH  ;LIST FROM C TO LIST DEVICE
F812 =          CSTS    EQU     0F812H  ;CONSOLE STATUS 00/FF TO REGISTER A
                ;
                ;       DISK PORTS AND COMMANDS
0078 =          BASE    EQU     78H     ;BASE OF DISK COMMAND IO PORTS
0078 =          DSTAT   EQU     BASE    ;DISK STATUS (INPUT)
0079 =          RTYPE   EQU     BASE+1  ;RESULT TYPE (INPUT)
007B =          RBYTE   EQU     BASE+3  ;RESULT BYTE (INPUT)
                ;
0079 =          ILOW    EQU     BASE+1  ;IOPB LOW ADDRESS (OUTPUT)
007A =          IHIGH   EQU     BASE+2  ;IOPB HIGH ADDRESS (OUTPUT)
                ;
0004 =          READF   EQU     4H      ;READ FUNCTION
0006 =          WRITF   EQU     6H      ;WRITE FUNCTION
0003 =          RECAL   EQU     3H      ;RECALIBRATE DRIVE
0004 =          IORDY   EQU     4H      ;I/O FINISHED MASK
000D =          CR      EQU     0DH     ;CARRIAGE RETURN
000A =          LF      EQU     0AH     ;LINE FEED
                ;
                SIGNON: ;SIGNON MESSAGE: XXK CP/M VERS Y.Y
169C 0D0A0A             DB      CR,LF,LF
                        IF      TEST
                        DB      '32'    ;32K EXAMPLE BIOS
                        ENDIF
                        IF      NOT TEST
169F 3030               DB      '00'    ;MEMORY SIZE FILLED BY RELOCATOR
                        ENDIF
16A1 6B20435020         DB      'k CP/M vers '
16AD 322E30             DB      VERS/10+'0','.',VERS MOD 10+'0'
16B0 0D0A00             DB      CR,LF,0
                ;
                BOOT:   ;PRINT SIGNON MESSAGE AND GO TO CCP
                ;       (NOTE: MDS BOOT INITIALIZED IOBYTE AT 0003H)
16B3 310001             LXI     SP,BUFF+80H
16B6 219C16             LXI     H,SIGNON
16B9 CDD317             CALL    PRMSG   ;PRINT MESSAGE
16BC AF                 XRA     A       ;CLEAR ACCUMULATOR
16BD 320400             STA     CDISK   ;SET INITIALLY TO DISK A
16C0 C30F17             JMP     GOCPM   ;GO TO CP/M
                ;
                ;
                WBOOT:; LOADER ON TRACK 0, SECTOR 1, WHICH WILL BE SKIPPED FOR W
                ;       READ CP/M FROM DISK - ASSUMING THERE IS A 128 BYTE COLD
                ;       START.
                ;
16C3 318000             LXI     SP,BUFF ;USING DMA - THUS 80 THRU FF AVAILABLE F
                ;
16C6 0E0A               MVI     C,RETRY ;MAX RETRIES
16C8 C5                 PUSH    B
```

```
                WBOOT0: ;ENTER HERE ON ERROR RETRIES
16C9 010000             LXI     B,CPMB  ;SET DMA ADDRESS TO START OF DISK SYSTEM
16CC CDBB17             CALL    SETDMA
16CF 0E00               MVI     C,0     ;BOOT FROM DRIVE 0
16D1 CD7D17             CALL    SELDSK
16D4 0E00               MVI     C,0
16D6 CDA717             CALL    SETTRK  ;START WITH TRACK 0
16D9 0E02               MVI     C,2     ;START READING SECTOR 2
16DB CDAC17             CALL    SETSEC
                ;
                ;       READ SECTORS, COUNT NSECTS TO ZERO
16DE C1                 POP     B       ;10-ERROR COUNT
16DF 062C               MVI     B,NSECTS
                RDSEC:  ;READ NEXT SECTOR
16E1 C5                 PUSH    B       ;SAVE SECTOR COUNT
16E2 CDC117             CALL    READ
16E5 C24917             JNZ     BOOTERR ;RETRY IF ERRORS OCCUR
16E8 2A6C18             LHLD    IOD     ;INCREMENT DMA ADDRESS
16EB 118000             LXI     D,128   ;SECTOR SIZE
16EE 19                 DAD     D       ;INCREMENTED DMA ADDRESS IN HL
16EF 44                 MOV     B,H
16F0 4D                 MOV     C,L     ;READY FOR CALL TO SET DMA
16F1 CDBB17             CALL    SETDMA
16F4 3A6B18             LDA     IOS     ;SECTOR NUMBER JUST READ
16F7 FE1A               CPI     26      ;READ LAST SECTOR?
16F9 DA0517             JC      RD1
                ;       MUST BE SECTOR 26, ZERO AND GO TO NEXT TRACK
16FC 3A6A18             LDA     IOT     ;GET TRACK TO REGISTER A
16FF 3C                 INR     A
1700 4F                 MOV     C,A     ;READY FOR CALL
1701 CDA717             CALL    SETTRK
1704 AF                 XRA     A       ;CLEAR SECTOR NUMBER
1705 3C         RD1:    INR     A       ;TO NEXT SECTOR
1706 4F                 MOV     C,A     ;READY FOR CALL
1707 CDAC17             CALL    SETSEC
170A C1                 POP     b       ;RECALL SECTOR COUNT
170B 05                 DCR     B       ;DONE?
170C C2E116             JNZ     RDSEC
                ;
                ;       DONE WITH THE LOAD, RESET DEFAULT BUFFER ADDRESS
                GOCPM:  ;(ENTER HERE FROM COLD START BOOT)
                ;       ENABLE RST0 AND RST7
170F F3                 DI
1710 3E12               MVI     A,12H   ;INITIALIZE COMMAND
1712 D3FD               OUT     REVRT
1714 AF                 XRA     A
1715 D3FC               OUT     INTC    ;CLEARED
1717 3E7E               MVI     A,INTE  ;RST0 AND RST7 BITS ON
1719 D3FC               OUT     INTC
171B AF                 XRA     A
171C D3F3               OUT     ICON    ;INTERRUPT CONTROL
                ;
                ;       SET DEFAULT BUFFER ADDRESS TO 80H
171E 018000             LXI     B,BUFF
1721 CDBB17             CALL    SETDMA
                ;
                ;       RESET MONITOR ENTRY POINTS
1724 3EC3               MVI     A,JMP
1726 320000             STA     0
1729 210316             LXI     H,WBOOTE
172C 220100             SHLD    1       ;JMP WBOOT AT LOCATION 00
172F 320500             STA     5
1732 21060E             LXI     H,BDOS
1735 220600             SHLD    6       ;JMP BDOS AT LOCATION 5
                        IF      NOT TEST
1738 323800             STA     7*8     ;JMP TO MON80 (MAY HAVE BEEN CHANGED BY
173B 2100F8             LXI     H,MONE0
173E 223900             SHLD    7*8+1
                        ENDIF
                ;       LEAVE IOBYTE SET
                ;       PREVIOUSLY SELECTED DISK WAS B, SEND PARAMETER TO CPM
1741 3A0400             LDA     CDISK   ;LAST LOGGED DISK NUMBER
1744 4F                 MOV     C,A     ;SEND TO CCP TO LOG IT IN
1745 FB                 EI
1746 C30000             JMP     CPMB
                ;
                ;       ERROR CONDITION OCCURRED, PRINT MESSAGE AND RETRY
                BOOTERR:
1749 C1                 POP     B       ;RECALL COUNTS
174A 0D                 DCR     C
174B CA5217             JZ      BOOTER0
                ;       TRY AGAIN
174E C5                 PUSH    B
174F C3C916             JMP     WBOOT0
                ;
                BOOTER0:
                ;       OTHERWISE TOO MANY RETRIES
1752 215B17             LXI     H,BOOTMSG
1755 CDD317             CALL    PRMSG
1758 C30FF8             JMP     RMON80  ;MDS HARDWARE MONITOR
                ;
                BOOTMSG:
175B 3F626F6F74         DB      '?boot',0
                ;
                ;
                CONST:  ;CONSOLE STATUS TO REG-A
                ;       (EXACTLY THE SAME AS MDS CALL)
1761 C312F8             JMP     CSTS
                ;
                CONIN:  ;CONSOLE CHARACTER TO REG-A
1764 CD03F8             CALL    CI
1767 E67F               ANI     7FH     ;REMOVE PARITY BIT
1769 C9                 RET
                ;
                CONOUT: ;CONSOLE CHARACTER FROM C TO CONSOLE OUT
176A C309F8             JMP     CO
                ;
                LIST:   ;LIST DEVICE OUT
                ;       (EXACTLY THE SAME AS MDS CALL)
176D C30FF8             JMP     LO
                ;
                LISTST
```

```
                        ;RETURN LIST STATUS
1770 AF                 XRA     A
1771 C9                 RET             ;ALWAYS NOT READY
                        ;
                        PUNCH:  ;PUNCH DEVICE OUT
                        ;       (EXACTLY THE SAME AS MDS CALL)
1772 C30CF8             JMP     PO
                        ;
                        READER: ;READER CHARACTER IN TO REG-A
                        ;       (EXACTLY THE SAME AS MDS CALL)
1775 C306F8             JMP     RI
                        ;
                        HOME:   ;MOVE TO HOME POSITION
                        ;       TREAT AS TRACK 00 SEEK
1778 0E00               MVI     C,0
177A C3A717             JMP     SETTRK
                        ;
                        SELDSK: ;SELECT DISK GIVEN BY REGISTER C
177D 210000             LXI     H,0000H ;RETURN 0000 IF ERROR
1780 79                 MOV     A,C
1781 FE04               CPI     NDISKS  ;TOO LARGE?
1783 D0                 RNC             ;LEAVE HL = 0000
                        ;
1784 E602               ANI     10B     ;00 00 FOR DRIVE 0,1 AND 10 10 FOR DRIVE
1786 326618             STA     DBANK   ;TO SELECT DRIVE BANK
1789 79                 MOV     A,C     ;00, 01, 10, 11
178A E601               ANI     1B      ;MDS HAS 0,1 AT 78, 2,3 AT 88
178C B7                 ORA     A       ;RESULT 00?
178D CA9217             JZ      SETDRIVE
1790 3E30               MVI     A,00110000B     ;SELECTS DRIVE 1 IN BANK
                        SETDRIVE:
1792 47                 MOV     B,A     ;SAVE THE FUNCTION
1793 216818             LXI     H,IOF   ;IO FUNCTION
1796 7E                 MOV     A,M
1797 E6CF               ANI     11001111B       ;MASK OUT DISK NUMBER
1799 B0                 ORA     B       ;MASK IN NEW DISK NUMBER
179A 77                 MOV     M,A     ;SAVE IT IN IOPB
179B 69                 MOV     L,C
179C 2600               MVI     H,0     ;HL=DISK NUMBER
179E 29                 DAD     H       ;*2
179F 29                 DAD     H       ;*4
17A0 29                 DAD     H       ;*8
17A1 29                 DAD     H       ;*16
17A2 113316             LXI     D,DPBASE
17A5 19                 DAD     D       ;HL=DISK HEADER TABLE ADDRESS
17A6 C9                 RET
                        ;
                        ;
                        SETTRK: ;SET TRACK ADDRESS GIVEN BY C
17A7 216A18             LXI     H,IOT
17AA 71                 MOV     M,C
17AB C9                 RET
                        ;
                        SETSEC: ;SET SECTOR NUMBER GIVEN BY C
17AC 216B18             LXI     H,IOS
17AF 71                 MOV     M,C
```

```
17B0 C9                 RET
                        SECTRAN:
                                ;TRANSLATE SECTOR BC USING TABLE AT DE
17B1 0600               MVI     B,0     ;DOUBLE PRECISION SECTOR NUMBER IN BC
17B3 EB                 XCHG            ;TRANSLATE TABLE ADDRESS TO HL
17B4 09                 DAD     B       ;TRANSLATE(SECTOR) ADDRESS
17B5 7E                 MOV     A,M     ;TRANSLATED SECTOR NUMBER TO A
17B6 326B18             STA     IOS
17B9 6F                 MOV     L,A     ;RETURN SECTOR NUMBER IN L
17BA C9                 RET
                        ;
                        SETDMA: ;SET DMA ADDRESS GIVEN BY REGS B,C
17BB 69                 MOV     L,C
17BC 60                 MOV     H,B
17BD 226C18             SHLD    IOD
17C0 C9                 RET
                        ;
                        READ:   ;READ NEXT DISK RECORD (ASSUMING DISK/TRK/SEC/DMA SET)
17C1 0E04               MVI     C,READF ;SET TO READ FUNCTION
17C3 CDE017             CALL    SETFUNC
17C6 CDF017             CALL    WAITIO  ;PERFORM READ FUNCTION
17C9 C9                 RET             ;MAY HAVE ERROR SET IN REG-A
                        ;
                        ;
                        WRITE:  ;DISK WRITE FUNCTION
17CA 0E06               MVI     C,WRITF
17CC CDE017             CALL    SETFUNC ;SET TO WRITE FUNCTION
17CF CDF017             CALL    WAITIO
17D2 C9                 RET             ;MAY HAVE ERROR SET
                        ;
                        ;
                        ;       UTILITY SUBROUTINES
                        PRMSG:  ;PRINT MESSAGE AT H,L TO 0
17D3 7E                 MOV     A,M
17D4 B7                 ORA     A       ;ZERO?
17D5 C8                 RZ
                        ;       MORE TO PRINT
17D6 E5                 PUSH    H
17D7 4F                 MOV     C,A
17D8 CD6A17             CALL    CONOUT
17DB E1                 POP     H
17DC 23                 INX     H
17DD C3D317             JMP     PRMSG
                        ;
                        SETFUNC:
                        ;       SET FUNCTION FOR NEXT I/O (COMMAND IN REG-C)
17E0 216818             LXI     H,IOF   ;IO FUNCTION ADDRESS
17E3 7E                 MOV     A,M     ;GET IT TO ACCUMULATOR FOR MASKING
17E4 E6F8               ANI     11111000B       ;REMOVE PREVIOUS COMMAND
17E6 B1                 ORA     C       ;SET TO NEW COMMAND
17E7 77                 MOV     M,A     ;REPLACED IN IOPB
                        ;       THE MDS-800 CONTROLLER REQUIRES DISK BANK BIT IN SECTOR
                        ;       MASK THE BIT FROM THE CURRENT I/O FUNCTION
17E8 E620               ANI     00100000B       ;MASK THE DISK SELECT BIT
17EA 216B18             LXI     H,IOS           ;ADDRESS THE SECTOR SELECT BYTE
17ED B6                 ORA     M               ;SELECT PROPER DISK BANK
```

```
17EE 77                 MOV     M,A             ;SET DISK SELECT BIT ON/OFF
17EF C9                 RET
                ;
                WAITIO:
17F0 0E0A               MVI     C,RETRY ;MAX RETRIES BEFORE PERM ERROR
                REWAIT:
                ;       START THE I/O FUNCTION AND WAIT FOR COMPLETION
17F2 CD3F18             CALL    INTYPE  ;IN RTYPE
17F5 CD4C18             CALL    INBYTE  ;CLEARS THE CONTROLLER
                ;
17F8 3A6618             LDA     DBANK           ;SET BANK FLAGS
17FB B7                 ORA     A               ;ZERO IF DRIVE 0,1 AND NZ IF 2,3
17FC 3E67               MVI     A,IOPB AND 0FFH ;LOW ADDRESS FOR IOPB
17FE 0618               MVI     B,IOPB SHR 8    ;HIGH ADDRESS FOR IOPB
1800 C20B18             JNZ     IODR1   ;DRIVE BANK 1?
1803 D379               OUT     ILOW            ;LOW ADDRESS TO CONTROLLER
1805 78                 MOV     A,B
1806 D37A               OUT     IHIGH   ;HIGH ADDRESS
1808 C31018             JMP     WAIT0           ;TO WAIT FOR COMPLETE
                ;
                IODR1:  ;DRIVE BANK 1
180B D389               OUT     ILOW+10H        ;88 FOR DRIVE BANK 10
180D 78                 MOV     A,B
180E D38A               OUT     IHIGH+10H
                ;
1810 CD5918     WAIT0:  CALL    INSTAT          ;WAIT FOR COMPLETION
1813 E604               ANI     IORDY           ;READY?
1815 CA1018             JZ      WAIT0
                ;
                ;       CHECK IO COMPLETION OK
1818 CD3F18             CALL    INTYPE          ;MUST BE IO COMPLETE (00) UNLINK
                ;       00 UNLINKED I/O COMPLETE,    01 LINKED I/O COMPLETE (NOT
                ;       10 DISK STATUS CHANGED       11 (NOT USED)
181B FE02               CPI     10B             ;READY STATUS CHANGE?
181D CA3218             JZ      WREADY
                ;
                ;       MUST BE 00 IN THE ACCUMULATOR
1820 B7                 ORA     A
1821 C23818             JNZ     WERROR          ;SOME OTHER CONDITION, RETRY
                ;
                ;       CHECK I/O ERROR BITS
1824 CD4C18             CALL    INBYTE
1827 17                 RAL
1828 DA3218             JC      WREADY          ;UNIT NOT READY
182B 1F                 RAR
182C E6FE               ANI     11111110B       ;ANY OTHER ERRORS?  (DELETED DAT
182E C23818             JNZ     WERROR
                ;
                ;       READ OR WRITE IS OK, ACCUMULATOR CONTAINS ZERO
1831 C9                 RET
                ;
                WREADY: ;NOT READY, TREAT AS ERROR FOR NOW
1832 CD4C18             CALL    INBYTE          ;CLEAR RESULT BYTE
1835 C33818             JMP     TRYCOUNT
                ;
                WERROR: ;RETURN HARDWARE MALFUNCTION (CRC, TRACK, SEEK, ETC.)
                ;       THE MDS CONTHOLLER HAS RETURNED A BIT IN EACH POSITION
                ;       OF THE ACCUMULATOR, CORRESPONDING TO THE CONDITIONS:
                ;       0       - DELETED DATA (ACCEPTED AS OK ABOVE)
                ;       1       - CRC ERROR
                ;       2       - SEEK ERROR
                ;       3       - ADDRESS ERROR (HARDWARE MALFUNCTION)
                ;       4       - DATA OVER/UNDER FLOW (HARDWARE MALFUNCTION)
                ;       5       - WRITE PROTECT (TREATED AS NOT READY)
                ;       6       - WRITE ERROR (HARDWARE MALFUNCTION)
                ;       7       - NOT READY
                ;       (ACCUMULATOR BITS ARE NUMBERED 7 6 5 4 3 2 1 0)
                ;
                ;       IT MAY BE USEFUL TO FILTER OUT THE VARIOUS CONDITIONS,
                ;       BUT WE WILL GET A PERMANENT ERROR MESSAGE IF IT IS NOT
                ;       RECOVERABLE.  IN ANY CASE, THE NOT READY CONDITION IS
                ;       TREATED AS A SEPARATE CONDITION FOR LATER IMPROVEMENT
                TRYCOUNT:
                ;       REGISTER C CONTAINS RETRY COUNT, DECREMENT 'TIL ZERO
183E 0D                 DCR     C
1839 C2F217             JNZ     REWAIT  ;FOR ANOTHER TRY
                ;
                ;       CANNOT RECOVER FROM ERROR
183C 3E01               MVI     A,1     ;ERROR CODE
183E C9                 RET
                ;
                ;       INTYPE, INBYTE, INSTAT READ DRIVE BANK 00 OR 10
183F 3A6618     INTYPE: LDA     DBANK
1842 B7                 ORA     A
1843 C24918             JNZ     INTYP1  ;SKIP TO BANK 10
1846 DB79               IN      RTYPE
1848 C9                 RET
1849 DB89       INTYP1: IN      RTYPE+10H       ;78 FOR 0,1  88 FOR 2,3
184B C9                 RET
                ;
184C 3A6618     INBYTE: LDA     DBANK
184F B7                 ORA     A
1850 C25618             JNZ     INBYT1
1853 DB7B               IN      RBYTE
1855 C9                 RET
1856 DB8B       INBYT1: IN      RBYTE+10H
1858 C9                 RET
                ;
1859 3A6618     INSTAT: LDA     DBANK
185C B7                 ORA     A
185D C26318             JNZ     INSTA1
1860 DB78               IN      DSTAT
1862 C9                 RET
1863 DB88       INSTA1: IN      DSTAT+10H
1865 C9                 RET
                ;
                ;
                ;
                ;       DATA AREAS (MUST BE IN RAM)
1866 00         DBANK:  DB      0               ;DISK BANK 00 IF DRIVE 0,1
                                                ;          10 IF DRIVE 2,3
                IOPB    ;IO PARAMETER BLOCK
```

```
1867 80                 DB      80H     ;NORMAL I/O OPERATION
186E 04         IOF:    DB      READF   ;IO FUNCTION, INITIAL READ
1869 01         ION     DB      1       ;NUMBER OF SECTORS TO READ
186A 02         IOT:    DB      OFFSET  ;TRACK NUMBER
186B 01         IOS:    DB      1       ;SECTOR NUMBER
186C 0000       IOD:    DW      BUFF    ;IO ADDRESS
                ;
                ;
                ;       DEFINE RAM AREAS FOR BDOS OPERATION
                        ENDEF
186E+=          BEGDAT  EQU     $
186E+           DIRBUF: DS      128     ;DIRECTORY ACCESS BUFFER
18EE+           ALV0:   DS      31
190D+           CSV0:   DS      16
191D+           ALV1:   DS      31
193C+           CSV1:   DS      16
194C+           ALV2:   DS      31
196B+           CSV2:   DS      16
197B+           ALV3:   DS      31
199A+           CSV3:   DS      16
19AA+=          ENDDAT  EQU     $
013C+=          DATSIZ  EQU     $-BEGDAT
19AA            END
```

```
001F ALS1       001F ALS2       001F ALS3       18EE ALV0       191D ALV1
194C ALV2       197B ALV3       0078 BASE       0806 BDOS       186E BEGDAT
0000 BIAS       16B3 BOOT       1752 BOOTER0    1749 BOOTERR    175B BOOTMSG
0080 BUFF       0004 CDISK      F803 CI         1764 CONIN      176A CONOUT
1761 CONST      F809 CO         0000 CPMB       1600 CPML       000D CR
0010 CSS1       0010 CSS2       0010 CSS3       F812 CSTS       190D CSV0
193C CSV1       196B CSV2       199A CSV3       013C DATSIZ     1866 DBANK
186E DIRBUF     1673 DPB0       1673 DPB1       1673 DPB2       1673 DPB3
1633 DPBASE     1633 DPE0       1643 DPE1       1653 DPE2       1663 DPE3
0078 DSTAT      19AA ENDDAT     0000 FALSE      170F GOCPM      1778 HOME
00F3 ICON       007A IHIGH      0079 ILOW       1856 INBYT1     184C INBYTE
1863 INSTA1     1859 INSTAT     00FC INTC       007E INTE       1849 INTYP1
183F INTYPE     186C IOD        180B IODR1      1868 IOF        1869 ION
1867 IOPB       0004 IORDY      186B IOS        186A IOT        000A LF
176D LIST       1770 LISTST     F80F LO         F800 MON80      002C NSECTS
0002 OFFSET     1600 PATCH      F80C PO         17D3 PRMSG      1772 PUNCH
007B RBYTE      1705 RD1        16E1 RDSEC      1775 READER     17C1 READ
0004 READF      0003 RECAL      000A RETRY      00FD REVRT      17F2 REWAIT
F806 RI         FF0F RMON80     0079 RTYPE      17B1 SECTRAN    177D SELDSK
17BB SETDMA     1792 SETDRIVE   17E0 SETFUNC    17AC SETSEC     17A7 SETTRK
169C SIGNON     0000 TEST       FFFF TRUE       1838 TRYCOUNT   0014 VERS
1810 WAIT0      17F0 WAITIO     16C9 WBOOT0     1603 WBOOTE     16C3 WBOOT
1838 WERROR     1832 WREADY     17CA WRITE      0006 WRITF      1682 XL10
1682 XLT1       1682 XLT2       1682 XLT3
```

```
ASM80 TRINT.SRC DEBUG

ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE    1

  LOC  OBJ          SEQ          SOURCE STATEMENT

                      1 ;        PIP INTERFACE TO BDOS (CAN BE USED FOR OTHER TRA
                      2          PUBLIC  BOOT,IOBYTE,BDISK,BDOS,MON1,MON2,MON3
                      3          PUBLIC  MAXB,FCB,BUFF
  0000                4 BOOT     EQU     0000H   ;WARM START
  0003                5 IOBYTE   EQU     0003H   ;IO BYTE
  0004                6 BDISK    EQU     0004H   ;BOOT DISK #
  0005                7 BDOS     EQU     0005H   ;BDOS ENTRY
  0005                8 MON1     EQU     0005H   ;BDOS ENTRY
  0005                9 MON2     EQU     0005H   ;BDOS ENTRY
  0005               10 MON3     EQU     0005H   ;BDOS ENTRY
  0006               11 MAXB     EQU     0006H   ;MAX MEM BASE
  005C               12 FCB      EQU     005CH   ;DEFAULT FCB
  0080               13 BUFF     EQU     0080H   ;DEFAULT BUFFER
                     14          END

PUBLIC SYMBOLS
BDISK  A 0004    BDOS   A 0005    BOOT   A 0000    BUFF   A 0080    FCB    A 005
MON1   A 0005    MON2   A 0005    MON3   A 0005

EXTERNAL SYMBOLS

USER SYMBOLS
BDISK  A 0004    BDOS   A 0005    BOOT   A 0000    BUFF   A 0080    FCB    A 005
MON1   A 0005    MON2   A 0005    MON3   A 0005

ASSEMBLY COMPLETE,  NO ERRORS
```



```
0020 =          VERSION EQU     20H
                ;       XSUB RELOCATOR PROGRAM, INCLUDED WITH THE MODULE
                ;       TO PERFORM THE MOVE FROM 200H TO THE DESTINATION ADDRESS
                ;
                ;       COPYRIGHT (C) 1979
                ;       DIGITAL RESEARCH
                ;       BOX 579
                ;       PACIFIC GROVE, CA.
                ;       93950
                ;
0100                    ORG     100H
0100 01                 DB      (LXI OR (B SHL 3))      ;LXI B,MODULE SIZE
0103                    ORG     $+2                     ;SKIP ADDRESS FIELD
0103 C38901             JMP     START
0106 2045787465         DB      ' Extended Submit Vers '
011C 322E30             DB      VERSION/16+'0','.',VERSION MOD 16+'0'
011F 2C20436F70         DB      ', Copyright (c) 1979, Digital Research '
0146 457874656ENOGO:    DB      'Extended Submit Already Present$'
0166 5265717569BADVER:  DB      'Requires CP/M Version 2.0 or later$'
                ;
0005 =          BDOS    EQU     0005H   ;BDOS ENTRY POINT
0009 =          PRINT   EQU     9       ;BDOS PRINT FUNCTION
000C =          VERS    EQU     12      ;GET VERSION NUMBER
0800 =          CCPLEN  EQU     0800H   ;SIZE OF CCP
0200 =          MODULE  EQU     200H    ;MODULE ADDRESS
                ;
                START:
                ;       CCP'S STACK USED THROUGHOUT
0189 C5                 PUSH    B       ;SAVE THE MODULE'S LENGTH
018A 3A0600             LDA     BDOS+1  ;XSUB ALREADY PRESENT?
018D FE06               CPI     06H     ;LOW ADDRESS MUST BE 06H
018F CA9C01             JZ      CONTINUE
                ;
                ;       BDOS IS NOT LOWEST MODULE IN MEMORY, RETURN TO CCP
0192 0E09               MVI     C,PRINT
0194 114601             LXI     D,NOGO  ;ALREADY PRESENT MESSAGE
0197 CD0500             CALL    BDOS    ;TO PRINT THE MESSAGE
019A C1                 POP     B       ;RECALL LENGTH
019B C9                 RET             ;TO THE CCP
                ;
                CONTINUE:
019C 0E0C               MVI     C,VERS
019E CD0500             CALL    BDOS    ;VERSION NUMBER?
01A1 FE20               CPI     VERSION ;2.0 OR GREATER
01A3 D2B001             JNC     VERSOK
                ;
                ;       WRONG VERSION
01A6 0E09               MVI     C,PRINT
01A8 116601             LXI     D,BADVER
01AB CD0500             CALL    BDOS
01AE C1                 POP     B
01AF C9                 RET             ;TO CCP
                ;
                VERSOK:
01B0 210700             LXI     H,BDOS+2;ADDRESS FIELD OF JUMP TO BDOS (TOP MEMO
01B3 7E                 MOV     A,M     ;A HAS HIGH ORDER ADDRESS OF MEMORY TOP
```

```
01B4 3D                 DCR     A       ;PAGE DIRECTLY BELOW BDOS
01B5 D600               SUI     (CCPLEN SHR 8)  ;-CCP PAGES
01B7 C1                 POP     B       ;RECALL LENGTH OF MODULE
01B8 C5                 PUSH    B       ;AND SAVE IT AGAIN
01B9 90                 SUB     B       ;A HAS HIGH ORDER ADDRESS OF RELOC AREA
01BA 57                 MOV     D,A
01BB 1E00               MVI     E,0     ;D,E ADDRESSES BASE OF RELOC AREA
01BD D5                 PUSH    D       ;SAVE FOR RELOCATION BELOW
                ;
01BE 210002             LXI     H,MODULE;READY FOR THE MOVE
01C1 78         MOVE:   MOV     A,B     ;BC=0?
01C2 B1                 ORA     C
01C3 CACF01             JZ      RELOC
01C6 0B                 DCX     B       ;COUNT MODULE SIZE DOWN TO ZERO
01C7 7E                 MOV     A,M     ;GET NEXT ABSOLUTE LOCATION
01C8 12                 STAX    D       ;PLACE IT INTO THE RELOC AREA
01C9 13                 INX     D
01CA 23                 INX     H
01CB C3C101             JMP     MOVE
                ;
                RELOC:  ;STORAGE MOVED, READY FOR RELOCATION
                ;       HL ADDRESSES BEGINNING OF THE BIT MAP FOR RELOCATION
01CE D1                 POP     D       ;RECALL BASE OF RELOCATION AREA
01CF C1                 POP     B       ;RECALL MODULE LENGTH
01D0 E5                 PUSH    H       ;SAVE BIT MAP BASE IN STACK
01D1 62                 MOV     H,D     ;RELOCATION BIAS IS IN D
                ;
01D2 78         REL0:   MOV     A,B     ;BC=0?
01D3 B1                 ORA     C
01D4 CAF001             JZ      ENDREL
                ;
                ;       NOT END OF THE RELOCATION, MAY BE INTO NEXT BYTE OF BIT
01D7 0B                 DCX     B       ;COUNT LENGTH DOWN
01D8 7B                 MOV     A,E
01D9 E607               ANI     111B    ;0 CAUSES FETCH OF NEXT BYTE
01DB C2E301             JNZ     REL1
                ;       FETCH BIT MAP FROM STACKED ADDRESS
01DE E3                 XTHL
01DF 7E                 MOV     A,M     ;NEXT 8 BITS OF MAP
01E0 23                 INX     H
01E1 E3                 XTHL            ;BASE ADDRESS GOES BACK TO STACK
01E2 6F                 MOV     L,A     ;L HOLDS THE MAP AS WE PROCESS 8 LOCATIO
01E3 7D         REL1:   MOV     A,L
01E4 17                 RAL             ;CY SET TO 1 IF RELOCATION NECESSARY
01E5 6F                 MOV     L,A     ;BACK TO L FOR NEXT TIME AROUND
01E6 D2EC01             JNC     REL2    ;SKIP RELOCATION IF CY=0
                ;
                ;       CURRENT ADDRESS REQUIRES RELOCATION
01E9 1A                 LDAX    D
01EA 84                 ADD     H       ;APPLY BIAS IN H
01EB 12                 STAX    D
01EC 13         REL2:   INX     D       ;TO NEXT ADDRESS
01ED C3D201             JMP     REL0    ;FOR ANOTHER BYTE TO RELOCATE
                ;
                ENDREL: ;END OF RELOCATION
01F0 D1                 POP     D       ;CLEAR STACKED ADDRESS
                ;       H HAS THE HIGH ORDER 8-BITS OF RELOCATED MODULE ADDRESS
01F1 2E00               MVI     L,0
01F3 E9                 PCHL            ;GO TO RELOCATED PROGRAM
01F4                    END
```

```
0166 BADVER     0005 BDOS       0800 CCPLEN     019C CONTINUE   01F0 ENDREL
0200 MODULE     01C1 MOVE       0146 NOGO       0009 PRINT      01D2 REL0
01E3 REL1       01EC REL2       01CE RELOC      0109 START      0020 VERSION
000C VERS       01B0 VERSOK
```

```
                ;       XSUB LOADS BELOW CCP, AND FEEDS COMMAND LINES TO
                ;       PROGRAMS WHICH READ BUFFERED INPUT
                ;
0000 =          BIAS    EQU     0000H   ;BIAS FOR RELOCATION
FFFF =          BASE    EQU     0FFFFH  ;NO INTERCEPTS BELOW HERE
0000 =          WBOOT   EQU     0000H
0005 =          BDOS    EQU     0005H
0006 =          BDOSL   EQU     BDOS+1
0080 =          DBUFF   EQU     0080H
                ;
000D =          CR      EQU     0DH     ;CARRIAGE RETURN
000A =          LF      EQU     0AH     ;LINE FEED
000E =          MODNUM  EQU     14      ;MODULE NUMBER POSITION
0009 =          PBUFF   EQU     9       ;PRINT BUFFER
000A =          RBUFF   EQU     10      ;READ BUFFER
000F =          OPENF   EQU     15      ;OPEN FILE
0010 =          CLOSEF  EQU     16      ;CLOSE FILE
0013 =          DELF    EQU     19      ;DELETE FILE
0014 =          DREADF  EQU     20      ;DISK READ
001A =          DMAF    EQU     26      ;SET DMA FUNCTION
                ;
                ;
0000                    ORG     0000H+BIAS
                ;       INITIALIZE JMPS TO INCLUDE XSUB MODULE
0000 211A00             LXI     H,WSTART
0003 220100             SHLD    WBOOT+1
0006 2A0600             LHLD    BDOSL
0009 221800             SHLD    RBDOS+1 ;REAL BDOS ENTRY
000C 214500             LXI     H,TRAP  ;ADDRESS TO FILL
000F 220600             SHLD    BDOSL   ;JMP @0005 LEADS TO TRAP
0012 E1                 POP     H       ;CCP RETURN ADDRESS
0013 229B01             SHLD    CCPRET
0016 E9                 PCHL            ;BACK TO CCP
                ;
0017 C30000     RBDOS:  JMP     0000H   ;FILLED IN AT INITIALIZATION
                ;
                WSTART:
001A 31BD01             LXI     SP,STACK
001D 0E09               MVI     C,PBUFF ;PRINT MESSAGE
001F 113500             LXI     D,ACTMSG
0022 CD1700             CALL    RBDOS
0025 218000             LXI     H,DBUFF ;RESTORE DEFAULT BUFFER
0028 229901             SHLD    UDMA
002B 214500             LXI     H,TRAP
002E 220600             SHLD    BDOSL   ;FIXUP LOW JUMP ADDRESS
0031 2A9B01             LHLD    CCPRET  ;BACK TO CCP
0034 E9                 PCHL
0035 0D0A2A2A73ACTMSG:  DB      CR,LF,'(xsub active)$'
                ;
                TRAP:   ;ARRIVE HERE AT EACH BDOS CALL
0045 E1                 POP     H       ;RETURN ADDRESS
0046 E5                 PUSH    H       ;BACK TO STACK
0047 7C                 MOV     A,H     ;HIGH ADDRESS
0048 FEFF               CPI     BASE SHR 8
004A D21700             JNC     RBDOS   ;SKIP CALLS ON BDOS ABOVE HERE
004D 79                 MOV     A,C     ;FUNCTION NUMBER
004E FE0A               CPI     RBUFF
0050 CA8D00             JZ      RNBUFF  ;READ NEXT BUFFER
0053 FE1A               CPI     DMAF    ;SET DMA ADDRESS?
0055 C21700             JNZ     RBDOS   ;SKIP IF NOT
0058 EB                 XCHG            ;DMA TO HL
0059 229901             SHLD    UDMA    ;SAVE IT
005C EB                 XCHG
005D C31700             JMP     RBDOS
                ;
                SETDMA:
0060 0E1A               MVI     C,DMAF
0062 111601             LXI     D,COMBUF
0065 CD1700             CALL    RBDOS
0068 C9                 RET
                ;
                RSETDMA:
0069 0E1A               MVI     C,DMAF
006B 2A9901             LHLD    UDMA
006E EB                 XCHG
006F CD1700             CALL    RBDOS
0072 C9                 RET
                ;
                FBDOS:
0073 C5                 PUSH    B
0074 D5                 PUSH    D
0075 CD6000             CALL    SETDMA
0078 D1                 POP     D
0079 C1                 POP     B
007A CD1700             CALL    RBDOS
007D F5                 PUSH    PSW
007E CD6900             CALL    RSETDMA
0081 F1                 POP     PSW
0082 C9                 RET
                ;
                CKSUB:  ;CHECK FOR SUB FILE PRESENT
0083 0E0F               MVI     C,OPENF
0085 11F500             LXI     D,SUBFCB
0088 CD7300             CALL    FBDOS   ;SUBMIT FILE PRESENT?
008B 3C                 INR     A       ;00 IF NOT PRESENT
008C C9                 RET
                ;
                RNBUFF:
008D D5                 PUSH    D       ;COMMAND ADDRESS
008E CD8300             CALL    CKSUB   ;SUB FILE PRESENT?
0091 D1                 POP     D
0092 0E0A               MVI     C,RBUFF
0094 CA1700             JZ      RBDOS   ;NO SUB FILE NOW
                ;
0097 D5                 PUSH    D
0098 3A0401             LDA     SUBRC   ;LENGTH OF FILE
009B B7                 ORA     A       ;ZERO?
009C CA1700             JZ      RBDOS   ;SKIP IF SO
009F 3D                 DCR     A       ;LENGTH - 1
00A0 321501             STA     SUBCR   ;NEXT TO READ
00A3 0F14               MVI     C,DREADF
00A5 11F500             LXI     D,SUBFCB
```

```
00A8 CD7300             CALL    FBDOS   ;READ RECORD
                ;       NOW PRINT THE BUFFER WITH CR,IF
00AB 211601             LXI     H,COMBUF
00AE 5E                 MOV     E,M     ;LENGTH
00AF 1600               MVI     D,0     ;HIGH ORDER 00
00B1 19                 DAD     D       ;TO LAST CHARACTER POSITION
00B2 23                 INX     H
00B3 360D               MVI     M,CR
00B5 23                 INX     H
00B6 360A               MVI     M,LF
00B8 23                 INX     H
00B9 3624               MVI     M,'$'
00BB 0E09               MVI     C,PBUFF
00BD 111701             LXI     D,COMBUF+1
00C0 CD1700             CALL    RBDOS   ;TO PRINT IT
00C3 E1                 POP     H       ;.MAX LENGTH
00C4 111601             LXI     D,COMBUF
00C7 1A                 LDAX    D       ;HOW LONG?
00C8 BE                 CMP     M       ;CY IF OK
00C9 DACE00             JC      MOVLIN
00CC 7E                 MOV     A,M     ;MAX LENGTH
00CD 12                 STAX    D       ;TRUNCATE LENGTH
                MOVLIN:
00CE 4F                 MOV     C,A     ;LENGTH TO C
00CF 0C                 INR     C       ;+1
00D0 23                 INX     H       ;TO LENGTH OF LINE
                RDLOOP:
00D1 1A                 LDAX    D       ;NEXT CHAR
00D2 77                 MOV     M,A
00D3 23                 INX     H
00D4 13                 INX     D
00D5 0D                 DCR     C
00D6 C2D100             JNZ     RDLOOP  ;LOOP TIL COPIED
00D9 0E10               MVI     C,CLOSEF
00DB 11F500             LXI     D,SUBFCB
00DE 210F00             LXI     H,MODNUM
00E1 19                 DAD     D       ;HL=FCB(MODNUM)
00E2 3600               MVI     M,0     ;=0 SO ACTS AS IF WRITTEN
00E4 3A1501             LDA     SUBCR   ;LENGTH OF FILE
00E7 3D                 DCR     A       ;INCREMENTED BY READ OP
00E8 320401             STA     SUBRC   ;DECREASE FILE LENGTH
00EB B7                 ORA     A       ;AT ZERO?
00EC C2F100             JNZ     FILEOP
00EF 0E13               MVI     C,DELF  ;DELETE IF AT END
00F1 CD7300     FILEOP: CALL    FBDOS
00F4 C9                 RET
                ;
                SUBFCB:
00F5 01                 DB      1       ;A:
00F6 2424242020         DB      '$$$    '
00FB 535542             DB      'SUB'
0101 000000             DB      0,0,0
                SUBRC
0104                    DS      1
0105                    DS      16      ;MAP
0115            SUBCR:  DS      1
                ;
0116            COMBUF: DS      131
0199 8000       UDMA:   DW      DBUFF
019B            CCPRET: DS      2       ;CCP RETURN ADDRESS
019D                    DS      32      ;16 LEVEL STACK
                STACK:
01BD                    END
```

```
0035 ACTMSG     FFFF BASE       0005 BDOS       0006 BDOSL      0000 BIAS
019B CCPRET     0083 CKSUB      0010 CLOSEF     0116 COMBUF     000D CR
0080 DBUFF      0013 DELF       001A DMAF       0014 DREADF     0073 FBDOS
00F1 FILEOP     000A LF         000F MODNUM     00CE MOVLIN     000F OPENF
0009 PBUFF      0017 RBDOS      000A RBUFF      00D1 RDLOOP     008D RNBUFF
0069 RSETDMA    0060 SETDMA     01BD STACK      0115 SUBCR      00F5 SUBFCB
0104 SUBRC      0045 TRAP       0199 UDMA       0000 WBOOT      001A WSTART
```

```
ISIS-II PL/M-80 V2.1 COMPILATION OF MODULE PIPMOD
OBJECT MODULE PLACED IN PIP.OBJ
COMPILER INVOKED BY:  PLM80 PIP.PLM DEBUG


   1          PIPMOD:
              DO;
              /* P E R I P H E R A L   I N T E R C H A N G E   P R O G R A M

                      COPYRIGHT (C) 1976, 1977, 1978, 1979
                      DIGITAL RESEARCH
                      BOX 579
                      PACIFIC GROVE, CA
                      93950
                  */

   2   1      DECLARE
                  CPMVERSION LITERALLY '0020H'; /* REQUIRED FOR OPERATION */

   3   1      DECLARE
                  IOBYTE   BYTE EXTERNAL,      /* IOBYTE AT 0003H */
                  MAXB  ADDRESS EXTERNAL,      /* ADDR FIELD OF JMP BDOS */
                  FCB (33) BYTE EXTERNAL,      /* DEFAULT FILE CONTROL BLOCK *
                  BUFF(128)BYTE EXTERNAL;      /* DEFAULT BUFFER */

   4   1      DECLARE
                  ENDFILE LITERALLY '1AH',     /* END OF FILE MARK */
                  JMP   LITERALLY '0C3H',      /* 8080 JUMP INSTRUCTION */
                  RET   LITERALLY '0C9H';      /* 8080 RETURN */

              /* THE FIRST PORTION OF THE PIP PROGRAM 'FAKES' THE PAGE ONE
              (100H - 1FFH) SECTION OF PIP WHICH CONTAINS A JUMP TO PIPENTRY, AN
              SPACE FOR CUSTOM I/O DRIVERS (WHICH CAN BE 'PATCHED' USING DDT) IN
              REMAINING PAGE ONE AREA.  THE PIP PROGRAM ACTUALLY STARTS AT 200H

   5   1      DECLARE JUMP BYTE DATA(JMP); /* JMP INSTRUCTION TO */
              /* JMP .PIPENTRY-3 WHERE THE LXI SP,STACK ACTUALLY OCCURS */
   6   1      DECLARE JADR ADDRESS DATA(.PIPENTRY-3); /* START OF PIP */
   7   1      DECLARE INPSUB(3) BYTE DATA(RET,0,0);     /* INP: RET NOP NOP */
   8   1      DECLARE OUTSUB(3) BYTE DATA(RET,0,0);     /* OUT: RET NOP NOP */
   9   1      DECLARE INPDATA BYTE DATA(ENDFILE); /* RETURNED DATA */
                  /* NOTE:  PAGE 1 AT 100H CONTAINS THE FOLLOWING
                  100H:  JMP PIPENTRY     ;TO START THE PIP PROGRAM
                  103H:  RET              ;INP: DEFAULTS TO EMPTY INPUT (DATA 1AH
                  104H:  NOP
                  105H:  NOP
                  106H:  RET              ;OUT: DEFAULTS TO EMPTY OUTPUT
                  107H:  NOP
                  108H:  NOP
                  109H: 1AH=ENDFILE       ;DATA FROM INP: FUNCTION IS STORED HERE
                                          ;RETURN FROM THE INP: ENTRY POINT
                  10AH: - 1FFH            ;SPACE RESERVED FOR SPECIAL PURPOSE
                  ; DRIVERS - IF INCLUDED, THEN REPLACE 103H AND 106H BY JMP'S
                  ; TO THE PROPER LOCATIONS WITHIN THE RESERVED AREA.
                  ;  ALSO, RETURN DATA FROM INP: ENTRY POINT AT 109H.
                  ; THESE DRIVERS ARE MOST EASILY INSERTED WITH THE DDT PROGRAM
```



```
                  ; UNDER CP/M
                  */

  10   1      DECLARE /* 16 BYTE MESSAGE */
                  FREEMEMORY LITERALLY '''(INP:/OUT:SPACE)''',
                  /* 256 BYTE AREA FOR INP: OUT: PATCHING */
                  RESERVED(*) BYTE DATA(0,0,0,0,0,0,
                  FREEMEMORY, FREEMEMORY, FREEMEMORY,
                  FREEMEMORY, FREEMEMORY, FREEMEMORY, FREEMEMORY,
                  FREEMEMORY, FREEMEMORY, FREEMEMORY, FREEMEMORY,
                  FREEMEMORY, FREEMEMORY, FREEMEMORY, FREEMEMORY);



  11   1          DECLARE COPYRIGHT(*) BYTE DATA (
                  '   COPYRIGHT (C) 1979, DIGITAL RESEARCH,  PIP VERS 1.5');

  12   1          DECLARE INPLOC ADDRESS DATA (.INPSUB);  /* ADDRESS OF INP: DEV
  13   1          DECLARE OUTLOC ADDRESS DATA (.OUTSUB);  /* ADDRESS OF OUT: DEV

  14   1      OUT: PROCEDURE(B);
  15   2          DECLARE B BYTE;
                  /* SEND B TO OUT: DEVICE */
  16   2          CALL OUTLOC;
  17   2          END OUT;

  18   1      INP: PROCEDURE BYTE;
  19   2          CALL INPLOC;
  20   2          RETURN INPDATA;
  21   2          END INP;


  22   1      TIMEOUT: PROCEDURE;
                  /* WAIT FOR 50 MSEC */
  23   2          CALL TIME(250); CALL TIME(250);
  25   2          END TIMEOUT;

                /* LITERAL DECLARATIONS */
  26   1          DECLARE
                    LIT LITERALLY 'LITERALLY',
                    LPP LIT '60',       /* LINES PER PAGE */
                    TAB LIT '09H',      /* HORIZONTAL TAB */
                    FF  LIT '0CH',      /* FORM FEED */
                    LA  LIT '05FH',     /* LEFT ARROW */
                    LB  LIT '05BH',     /* LEFT BRACKET */
                    RB   LIT '05DH',    /* RIGHT BRACKET */
                    XOFF LIT '13H',  /* TRANSMIT BUFFER FUNCTION */

                    RDR LIT '5',
                    LST LIT '10',
                    PUNP LIT '15',                  /* POSITION OF 'PUN' + 1 */
                    CONP LIT '19',                  /* CONSOLE */
                    NULP LIT '19',                  /* NUL: BEFORE INCREMENT */
                    EOFP LIT '20',                  /* EOF: BEFORE INCREMENT */
                    HSRDR LIT 'RDR',                /* READER DEVICES */
                    PRNT  LIT '10',                 /* PRINTER */
```

```
                FSIZE LIT '33',
                FRSIZE LIT '36',  /* SIZE OF RANDOM FCb */
                NSIZE LIT '8',
                FNSIZE LIT '11',
                MDISK LIT '1',
                FNAM LIT '8',
                FEXT LIT '9',
                FEXTL LIT '3',
                ROFILE  LITERALLY '9',  /* READ ONLY FILE FIELD */
                SYSFILE LITERALLY '10', /* SYSTEM FILE FIELD */
                FREEL LIT '12',  /* REEL NUMBER FIELD OF FCB */

                HBUFS LIT '80',                 /* "HEX" BUFFER SIZE */

                ERR LIT '0',
                SPECL LIT '1',
                FILE LIT '2',
                PERIPH LIT '3',
                DISKNAME LIT '4';

 27   1       DECLARE
                COLUMN BYTE,      /* COLUMN COUNT FOR PRINTER TABS */
                LINENO BYTE,      /* LINE WITHIN PAGE */
                AMBIG BYTE,                     /* SET FOR AMBIGUOUS FILE REFS *
                PARSET BYTE,                    /* TRUE IF PARAMETERS PRESENT */
                FEEDBASE BYTE,                  /* USED TO FEED SEARCH CHARACTER
                FEEDLEN BYTE,                   /* LENGTH OF FEED STRING */
                MATCHLEN BYTE,                  /* USED IN MATCHING STRINGS */
                QUITLEN BYTE,                   /* USED TO TERMINATE QUIT COMMAN
                NBUF BYTE,                      /* NUM BUFFERS-1 IN SBUFF AND DB
                CDISK BYTE,                     /* CURRENT DISK */
                BUFFER LITERALLY 'BUFF',        /* DEFAULT BUFFER */
                SEARFCB LITERALLY 'FCB',        /* SEARCH FCB IN MULTI COPY */
                MEMSIZE LITERALLY 'MAXB',       /* MEMORY SIZE */
                SBLEN ADDRESS,                  /* SOURCE BUFFER LENGTH */
                DBLEN ADDRESS,                  /* DEST BUFFER LENGTH */
                SBASE ADDRESS,                  /* SOURCE BUFFER BASE */
                /* THE VECTORS DBUFF AND SBUFF ARE DECLARED WITH DIMENSION
                1024, BUT ACTUALLY VARY WITH THE FREE MEMORY SIZE */
                DBUFF(1024) BYTE AT (.MEMORY), /* DESTINATION BUFFER */
                SBUFF BASED SBASE (1024) BYTE, /* SOURCE BUFFER */
                SDISK BYTE,                     /* SOURCE DISK */
                (SCOM, DHEX) BYTE,              /* SOURCE IS 'COM' FILE IF TRUE
                                                /* DEST IS 'HEX' FILE IF TRUE
                SOURCE (FSIZE) BYTE,            /* SOURCE FCB */
                SFUB BYTE AT(.SOURCE(13)),      /* UNFILLED BYTES FIELD */
                DEST (FRSIZE) BYTE,             /* DESTINATION FCB */
                DESTR ADDRESS AT(.DEST(33)),    /* RANDOM RECORD POSITION */
                DESTO BYTE    AT(.DEST(35)),    /* RANDOM OVERFLOW BYTE */
                DFUB BYTE AT (.DEST(13)),       /* UNFILLED BYTES FIELD */
                DDISK BYTE,                     /* DESTINATION DISK */
                HBUFF(HBUFS) BYTE,              /* HEX FILE BUFFER */
                HSOURCE BYTE,                   /* NEXT HEX SOURCE CHARACTER */

                NSOURCE ADDRESS,                /* NEXT SOURCE CHARACTER *
P
```



```
                HARDEOF ADDRESS,                /* SET TO NSOURCE ON REAL EOF */
                NDEST ADDRESS;                  /* NEXT DESTINATION CHARACTER */

 28   1       DECLARE
                /* SUBMIT FILE CONTROL BLOCK FOR ERROR DELETE */
                SUBFCB (*) BYTE DATA (0,'$$$     SUB',0,0,0);

 29   1        DECLARE
                PDEST BYTE,                     /* DESTINATION DEVICE */
                PSOURCE BYTE;                   /* CURRENT SOURCE DEVICE */

 30   1        DECLARE
                MULTCOM BYTE,                   /* FALSE IF PROCESSING ONE LINE
                PUTNUM BYTE,                    /* SET WHEN READY FOR NEXT LINE
                CONCNT BYTE,                     /* COUNTER FOR CONSOLE READY CH
                CHAR BYTE,                      /* LAST CHARACTER SCANNED */
                TYPE BYTE,                      /* TYPE OF CHARACTER SCANNED */
                FLEN BYTE;                      /* FILE NAME LENGTH */

 31   1       MON1: PROCEDURE(F,A) EXTERNAL;
 32   2           DECLARE F BYTE,
                    A ADDRESS;
 33   2           END MON1;

 34   1       MON2: PROCEDURE(F,A) BYTE EXTERNAL;
 35   2           DECLARE F BYTE,
                    A ADDRESS;
 36   2           END MON2;

 37   1       MON3: PROCEDURE(F,A) ADDRESS EXTERNAL;
 38   2           DECLARE F BYTE,
                    A ADDRESS;
 39   2           END MON3;

 40   1       BOOT: PROCEDURE EXTERNAL;
                /* SYSTEM REBOOT */
 41   2           END BOOT;

 42   1       READRDR: PROCEDURE BYTE;
                /* READ CURRENT READER DEVICE */
 43   2           RETURN MON2(3,0);
 44   2           END READRDR;

 45   1       READCHAR: PROCEDURE BYTE;
                /* READ CONSOLE CHARACTER *
 46   2           RETURN MON2(1,0);
 47   2           END READCHAR;

 48   1       DECLARE
                TRUE LITERALLY '1',
                FALSE LITERALLY '0',
                FOREVER LITERALLY 'WHILE TRUE',
                CR LITERALLY '13',
                LF LITERALLY '10',
                WHAT LITERALLY '63';

 49   1       PRINTCHAR: PROCEDURE(CHAR);
P
```

```
 50   2          DECLARE CHAR BYTE;
 51   2          CALL MON1(2,CHAR AND 7FH);
 52   2          END PRINTCHAR;

 53   1      CRLF: PROCEDURE;
 54   2          CALL PRINTCHAR(CR);
 55   2          CALL PRINTCHAR(LF);
 56   2          END CRLF;

 57   1      PRINT: PROCEDURE(A);
 58   2          DECLARE A ADDRESS;
                 /* PRINT THE STRING STARTING AT ADDRESS A UNTIL THE
                 NEXT DOLLAR SIGN IS ENCOUNTERED */
 59   2          CALL CRLF;
 60   2          CALL MON1(9,A);
 61   2          END PRINT;

 62   1      DECLARE DCNT BYTE;

 63   1      VERSION: PROCEDURE ADDRESS;
 64   2          RETURN MON3(12,0); /* VERSION NUMBER */
 65   2          END VERSION;

 66   1      INITIALIZE: PROCEDURE;
 67   2          CALL MON1(13,0);
 68   2          END INITIALIZE;

 69   1      SELECT: PROCEDURE(D);
 70   2          DECLARE D BYTE;
 71   2          CALL MON1(14,D);
 72   2          END SELECT;

 73   1      OPEN: PROCEDURE(FCB);
 74   2          DECLARE FCB ADDRESS;
 75   2          DCNT = MON2(15,FCB);
 76   2          END OPEN;

 77   1      CLOSE: PROCEDURE(FCB);
 78   2          DECLARE FCB ADDRESS;
 79   2          DCNT = MON2(16,FCB);
 80   2          END CLOSE;

 81   1      SEARCH: PROCEDURE(FCB);
 82   2          DECLARE FCB ADDRESS;
 83   2          DCNT = MON2(17,FCB);
 84   2          END SEARCH;

 85   1      SEARCHN: PROCEDURE;
 86   2          DCNT = MON2(18,0);
 87   2          END SEARCHN;

 88   1      DELETE: PROCEDURE(FCB);
 89   2          DECLARE FCB ADDRESS;
 90   2          CALL MON1(19,FCB);
 91   2          END DELETE;

 92   1      DISKREAD: PROCEDURE(FCB) BYTE;
P
```



```
 93   2          DECLARE FCB ADDRESS;
 94   2          RETURN MON2(20,FCB);
 95   2          END DISKREAD;

 96   1      DISKWRITE: PROCEDURE(FCB) BYTE;
 97   2          DECLARE FCB ADDRESS;
 98   2          RETURN MON2(21,FCB);
 99   2          END DISKWRITE;

100   1      MAKE: PROCEDURE(FCB);
101   2          DECLARE FCB ADDRESS;
102   2          DCNT = MON2(22,FCB);
103   2          END MAKE;

104   1      RENAME: PROCEDURE(FCB);
105   2          DECLARE FCB ADDRESS;
106   2          CALL MON1(23,FCB);
107   2          END RENAME;

108   1      DECLARE
               CUSER BYTE, /* CURRENT USER NUMBER */
               SUSER BYTE; /* SOURCE USER NUMBER ('G' PARAMETER) */

109   1      SETIND: PROCEDURE(FCB);
110   2          DECLARE FCB ADDRESS;
111   2          CALL MON1(30,FCB);
112   2          END SETIND;

113   1      GETUSER: PROCEDURE BYTE;
114   2          RETURN MON2(32,0FFH);
115   2          END GETUSER;

116   1      SETUSER: PROCEDURE(USER);
117   2          DECLARE USER BYTE;
118   2          CALL MON1(32,USER);
119   2          END SETUSER;

120   1      SETCUSER: PROCEDURE;
121   2          CALL SETUSER(CUSER);
122   2          END SETCUSER;

123   1      SETSUSER: PROCEDURE;
124   2          CALL SETUSER(SUSER);
125   2          END SETSUSER;

126   1      READ$RANDOM: PROCEDURE(FCB) BYTE;
127   2          DECLARE FCB ADDRESS;
128   2          RETURN MON2(33,FCB);
129   2          END READ$RANDOM;

130   1      WRITE$RANDOM: PROCEDURE(FCB) BYTE;
131   2          DECLARE FCB ADDRESS;
132   2          RETURN MON2(34,FCB);
133   2          END WRITE$RANDOM;

134   1      SET$RANDOM: PROCEDURE(FCB);
135   2          DECLARE FCB ADDRESS;
```

```
                /* SET RANDOM RECORD POSITION */
136   2         CALL MON1(36,FCB);
137   2         END SET$RANDOM;

138   1       DECLARE CBUFF(130) BYTE,   /* COMMAND BUFFER */
                  MAXLEN BYTE AT (.CBUFF(0)),  /* MAX BUFFER LENGTH */
                  COMLEN BYTE AT (.CBUFF(1)),  /* CURRENT LENGTH */
                  COMBUFF (128) BYTE AT (.CBUFF(2)); /* COMMAND BUFFER CONTENTS
139   1       DECLARE (TCBP,CBP) BYTE;   /* TEMP CBP, COMMAND BUFFER POINTER */

140   1       READCOM: PROCEDURE;
                  /* READ INTO COMMAND BUFFER */
141   2         MAXLEN = 128;
142   2         CALL MON1,10,.MAXLEN);
143   2         END READCOM;

144   1       DECLARE MCBP BYTE;

145   1       CONBRK: PROCEDURE BYTE;
                  /* CHECK CONSOLE CHARACTER READY */
146   2         RETURN MON2(11,0);
147   2         END CONBRK;

148   1       DECLARE /* CONTROL TOGGLE VECTOR */
                  CONT(26) BYTE,   /* ONE FOR EACH ALPHABETIC */
                  /* 00 01 02 03 04 05 06 07 08 09 10 11 12 13
                       A  B  C  D  E  F  G  H  I  J  K  L  M  N
                     14 15 16 17 18 19 20 21 22 23 24 25
                       O  P  Q  R  S  T  U  V  W  X  Y  Z   */
                  BLOCK  BYTE AT(.CONT(1)),     /* BLOCK MODE TRANSFER */
                  DELET  BYTE AT(.CONT(3)),     /* DELETE CHARACTERS */
                  ECHO   BYTE AT(.CONT(4)),     /* ECHO CONSOLE CHARACTERS */
                  FORMF  BYTE AT(.CONT(5)),     /* FORM FILTER */
                  GETU   BYTE AT(.CONT(6)),     /* GET FILE, USER # */
                  HEXT   BYTE AT(.CONT(7)),     /* HEX FILE TRANSFER */
                  IGNOR  BYTE AT(.CONT(8)),     /* IGNORE :00 RECORD ON FILE */
                  LOWER  BYTE AT(.CONT(11)),    /* TRANSLATE TO LOWER CASE */
                  NUMB   BYTE AT(.CONT(13)),    /* NUMBER OUTPUT LINES */
                  OBJ    BYTE AT(.CONT(14)),    /* OBJECT FILE TRANSFER */
                  PAGCNT BYTE AT(.CONT(15)),    /* PAGE LENGTH */
                  QUITS  BYTE AT(.CONT(16)),    /* QUIT COPY */
                  RSYS   BYTE AT(.CONT(17)),    /* READ SYSTEM FILES */
                  STARTS BYTE AT(.CONT(18)),    /* START COPY */
                  TABS   BYTE AT(.CONT(19)),    /* TAB SET */
                  UPPER  BYTE AT(.CONT(20)),    /* UPPER CASE TRANSLATE */
                  VERIF  BYTE AT(.CONT(21)),    /* VERIFY EQUAL FILES ONLY */
                  WRROF  BYTE AT(.CONT(22)),    /* WRITE TO R/O FILE */
                  ZEROP  BYTE AT(.CONT(25));    /* ZERO PARITY ON INPUT */

149   1         SETDMA: PROCEDURE(A);
150   2           DECLARE A ADDRESS;
151   2           CALL MON1(26,A);
152   2           END SETDMA;

              /* INTELLEC 8 INTEL/ICOM READER INPUT */

153   1       INTIN: PROCEDURE BYTE;
P
```



```
                /* READ THE INTEL / ICOM READER */
154   2         DECLARE PTRI LITERALLY '3',   /* DATA */
                        PTRS LITERALLY '1',   /* STATUS */
                        PTRC LITERALLY '1',   /* COMMAND */
                        PTRG LITERALLY '0CH', /* GO */
                        PTRN LITERALLY '08H'; /* STOP */

                /* STROBE THE READER */
155   2         OUTPUT(PTRC) = PTRG;
156   2         OUTPUT(PTRC) = PTRN;
157   2           DO WHILE NOT ROL(INPUT(PTRS),3); /* NOT READY */
158   3           END;
                /* DATA READY */
159   2         RETURN INPUT(PTRI) AND 7FH;
160   2         END INTIN;


161   1       DECLARE ZEROSUP BYTE,  /* ZERO SUPPRESSION */
                  (C3,C2,C1) BYTE;     /* LINE COUNT ON PRINTER  */

162   1         ERROR: PROCEDURE(A);
163   2           DECLARE A ADDRESS, I BYTE;
164   2           CALL SETCUSER;
165   2           CALL PRINT(A); CALL PRINTCHAR(':'); CALL PRINTCHAR(' ');
168   2             DO I = TCBP TO CBP;
169   3             IF I < COMLEN THEN CALL PRINTCHAR(COMBUFF(I));
171   3             END;
                  /* ZERO THE COMLEN IN CASE THIS IS A SINGLE COMMAND */
172   2           COMLEN = 0;
                  /* DELETE ANY $$$.SUB FILES IN CASE BATCH PROCESSING */
                  /* DELETE SUB FILE ONLY IF PRESENT (MAY BE R/O DISK) */
173   2           CALL SEARCH(.SUBFCB);
174   2           IF DCNT <> 255 THEN CALL DELETE(.SUBFCB);
176   2           CALL CRLF;
177   2           GO TO RETRY;
178   2           END ERROR;

179   1         MOVE: PROCEDURE(S,D,N);
180   2           DECLARE (S,D) ADDRESS, N BYTE;
181   2           DECLARE A BASED S BYTE, B BASED D BYTE;
182   2             DO WHILE (N:=N-1) <> 255;
183   3             B = A; S = S+1; D = D+1;
186   3             END;
187   2           END MOVE;


188   1         FILLSOURCE: PROCEDURE;
                  /* FILL THE SOURCE BUFFERS */
189   2           DECLARE (I,J) BYTE;
190   2           NSOURCE = 0;
191   2           CALL SELECT(SDISK);
192   2           CALL SETSUSER; /* SOURCE USER NUMBER SET */
193   2             DO I = 0 TO NBUF;
                    /* SET DMA ADDRESS TO NEXT BUFFER POSITION */
194   3             CALL SETDMA(.SBUFF(NSOURCE));
195   3             IF (J := DISKREAD(.SOURCE)) <> 0 THEN
196   3               DO; IF J <> 1 THEN
```

```
198  4                      CALL ERROR(.('DISK READ ERROR$'));
                          * END - OF - FILE */
199  4                  HARDEOF = NSOURCE; /* SET HARD END-OF-FILE */
200  4                  SBUFF(NSOURCE) = ENDFILE; I = NBUF;
202  4                  END; ELSE
203  3              NSOURCE = NSOURCE + 128;
204  3              END;
205  2          NSOURCE = 0;
206  2          CALL SETCUSER; /* BACK TO CURRENT USER NUMBER */
207  2          END FILLSOURCE;


208  1      WRITEDEST: PROCEDURE;
             /* WRITE OUTPUT BUFFERS UP TO BUT NOT INCLUDING POSITION
             NDEST = THE LOW ORDER 7 BITS OF NDEST ARE ZERO */
209  2          DECLARE (I, J, N) BYTE;
210  2          DECLARE DMA ADDRESS;
211  2          DECLARE DATAOK BYTE;
212  2          IF (N := LOW(SHR(NDEST,7)) - 1) = 255 THEN RETURN ;
214  2          NDEST = 0;
215  2          CALL SELECT(DDISK);
216  2          CALL SETRANDOM(.DEST); /* SET BASE RECORD FOR VERIFY */
217  2              DO I = 0 TO N;
                    /* SET DMA ADDRESS TO NEXT BUFFER */
218  3              DMA = .DBUFF(NDEST);
219  3              CALL SETDMA(DMA);
220  3              IF DISKWRITE(.DEST) <> 0 THEN
221  3                  CALL ERROR(.('DISK WRITE ERROR$'));
222  3              NDEST = NDEST + 128;
223  3              END;
224  2          IF VERIF THEN /* VERIFY DATA WRITTEN OK */
225  2              DO;
226  3              NDEST = 0;
227  3              CALL SETDMA(.BUFF); /* FOR COMPARE */
228  3                  DO I = 0 TO N;
229  4                  DATAOK = READRANDOM(.DEST) = 0;
230  4                  DESTR = DESTR + 1; /* NEXT RANDOM READ */
231  4                  J = 0;
                            /* PERFORM COMPARISON */
232  4                      DO WHILE DATAOK AND J < 80H;
233  5                      DATAOK = BUFFER(J) = DBUFF(NDEST+J);
234  5                      J = J + 1;
235  5                      END;
236  4                  NDEST = NDEST + 128;
237  4                  IF NOT DATAOK THEN
238  4                      CALL ERROR(.('VERIFY ERROR$'));
239  4                  END;
240  3              DATAOK = DISKWRITE(.DEST);
                    /* NOW READY TO CONTINUE THE WRITE OPERATION */
241  3              END;
242  2          NDEST = 0;
243  2          END WRITEDEST;

244  1      PUTDCHAR: PROCEDURE(B);
245  2          DECLARE (B,IOB) BYTE;
             /* WRITE BYTE B TO THE DESTINATION DEVICE GIVEN BY PDEST */
246  2          IF B >= ' ' THEN
```



```
247  2              DO; COLUMN = COLUMN + 1;
249  3              IF DELET > 0 THEN /* MAY BE PAST RIGHT SIDE */
250  3                  DO; IF COLUMN > DELET THEN RETURN;
253  4                  END;
254  3              END;
255  2          IOB = IOBYTE; /* IN CASE IT IS ALTERED */
256  2              DO CASE PDEST;
                    /* CASE 0 IS THE DESTINATION FILE */
257  3                  DO;
258  4                  IF NDEST >= DBLEN THEN CALL WRITEDEST;
260  4                  DBUFF(NDEST) = B;
261  4                  NDEST = NDEST+1;
262  4                  END;
                    /* CASE 1 IS ARD (ADDMASTER) */
263  3                  GO TO NOTDEST;
                    /* CASE 2 IS IRD (INTEL/ICOM) */
264  3                  GO TO NOTDEST;
                    /* CASE 3 IS PTR */
265  3                  GO TO NOTDEST;
                    /* CASE 4 IS UR1 */
266  3                  GO TO NOTDEST;
                    /* CASE 5 IS UR2 */
267  3                  GO TO NOTDEST;
                    /* CASE 6 IS RLR */
268  3                  NOTDEST:
                            CALL ERROR(.('NOT A CHARACTER SINK$'));
                    /* CASE 7 IS OUT */
269  3                  CALL OUT(B);
                    /* CASE 8 IS LPT */
270  3                      DO; IOBYTE = 1000$0000B; GO TO LSTL;
273  4                      END;
                    /* CASE 9 IS UL1 */
274  3                      DO; IOBYTE = 1100$0000B; GO TO LSTL;
277  4                      END;
                    /* CASE 10 IS PRN (TABS EXPANDED, LINES LISTED, CHANGED TO
278  3                      DO; IOBYTE = 1000$0000B; GO TO LSTL;
281  4                      END;
                    /* CASE 11 IS LST */
282  3                  LSTL:
                            CALL MON1(5,B);
                    /* CASE 12 IS PTP */
283  3                  DO; IOBYTE = 0001$0000B; GO TO PUNL;
286  4                  END;
                    /* CASE 13 IS UP1 */
287  3                  DO; IOBYTE = 0010$0000B; GO TO PUNL;
290  4                  END;
                    /* CASE 14 IS UP2 */
291  3                  DO; IOBYTE = 0011$0000B; GO TO PUNL;
294  4                  END;
                    /* CASE 15 IS PUN */
295  3                  PUNL:
                            CALL MON1(4,B);
                        /* CASE 16 IS TTY */
296  3                      DO; IOBYTE = 0; GO TO CONL;
299  4                      END;
                        /* CASE 17 IS CRT */
300  3                      DO; IOBYTE = 1; GO TO CONL;
```

```
303   4                   END;
                        /* CASE 18 IS UC1 */
304   3                   DO; IOBYTE = 11B; GO TO CONL;
307   4                   END;
                        /* CASE 19 IS CON */
308   3                   CONL:
                            CALL MON1(2,B);
309   3                END;
310   2           IOBYTE = IOB;
311   2           END PUTDCHAR;

312   1       PUTDESTC: PROCEDURE(B);
313   2           DECLARE (B,I) BYTE;
                    /* WRITE DESTINATION CHARACTER, TAB EXPANSION */
314   2           IF B <> TAB THEN CALL PUTDCHAR(B); ELSE
316   2           IF TABS = 0 THEN CALL PUTDCHAR(B); ELSE
                        /* B IS TAB CHAR, TABS > 0 */
318   2               DO; I = COLUMN;
320   3                   DO WHILE I >= TABS;
321   4                   I = I - TABS;
322   4                   END;
323   3               I = TABS - I;
324   3                   DO WHILE I > 0;
325   4                   I = I - 1;
326   4                   CALL PUTDCHAR(' ');
327   4                   END;
328   3               END;
329   2           IF B = CR THEN COLUMN = 0;
331   2           END PUTDESTC;

332   1       PRINT1: PROCEDURE(B);
333   2           DECLARE B BYTE;
334   2           IF (ZEROSUP := ZEROSUP AND B = 0) THEN CALL PUTDESTC(' '); ELS
336   2               CALL PUTDESTC('0'+B);
337   2           END PRINT1;

338   1       PRINTDIG: PROCEDURE(D);
339   2           DECLARE D BYTE;
340   2           CALL PRINT1(SHR(D,4)); CALL PRINT1(D AND 1111B);
342   2           END PRINTDIG;

343   1       NEWLINE: PROCEDURE;
344   2           DECLARE ONE BYTE;
345   2           ONE = 1;
346   2           ZEROSUP = NUMB = 1;
347   2           C1 = DEC(C1+ONE); C2 = DEC(C2 PLUS 0); C3 = DEC(C3 PLUS 0);
350   2           CALL PRINTDIG(C3); CALL PRINTDIG(C2); CALL PRINTDIG(C1);
353   2           IF NUMB = 1 THEN /* USUALLY PRINTER OUTPUT */
354   2               DO; CALL PUTDESTC(':'); CALL PUTDESTC(' ');
357   3               END; ELSE
358   2               CALL PUTDESTC(TAB);
359   2           END NEWLINE;

360   1       CLEARBUFF: PROCEDURE;
                    /* CLEAR OUTPUT BUFFER IN BLOCK MODE TRANSMISION */
361   2           DECLARE NA ADDRESS;
362   2           DECLARE I BYTE;
```



```
363   2           I = LOW(NDEST) AND 7FH;  /* REMAINING PARTIAL BUFFER LENGTH */
364   2           NA = NDEST AND 0FF80H;   /* START OF SEGMENT NOT WRITTEN */
365   2           CALL WRITEDEST;          /* CLEARS BUFFERS */
366   2           CALL MOVE(.DBUFF(NA),.DBUFF,I);
                  /* DATA MOVED TO BEGINNING OF BUFFER */
367   2           NDEST = I;
368   2           END CLEARBUFF;

369   1       PUTDEST: PROCEDURE(B);
370   2           DECLARE (I,B) BYTE;
                    /* WRITE DESTINATION CHARACTER, CHECK TABS AND LINES */
371   2           IF FORMF THEN /* SKIP FORM FEEDS */
372   2               DO; IF B = FF THEN RETURN;
375   3               END;
376   2           IF PUTNUM THEN /* END OF LINE OR START OF FILE */
377   2               DO;
378   3               IF B <> FF THEN /* NOT FORM FEED */
379   3                   DO;
380   4                   IF (I:=PAGCNT) <> 0 THEN /* PAGE EJECT */
381   4                       DO; IF I=1 THEN I:=LPP;
384   5                       IF (LINENO := LINENO + 1) >= I THEN
385   5                           DO; LINENO = 0; /* NEW PAGE */
387   6                           CALL PUTDESTC(FF);
388   6                           END;
389   5                       END;
390   4                   IF NUMB <> 0 THEN
391   4                       CALL NEWLINE;
392   4                   PUTNUM = FALSE;
393   4                   END;
394   3               END;
395   2           IF BLOCK THEN /* BLOCK MODE TRANSFER */
396   2               DO;
397   3               IF B = XOFF AND PDEST = 0 THEN
398   3                   DO; CALL CLEARBUFF; /* BUFFERS WRITTEN */
400   4                   RETURN; /* DON'T PASS THE X-OFF */
401   4                   END;
402   3               END;
403   2           IF B = FF THEN LINENO = 0;
405   2           CALL PUTDESTC(B);
406   2           IF B = LF THEN PUTNUM = TRUE;
408   2           END PUTDEST;


409   1       UTRAN: PROCEDURE(B) BYTE;
410   2           DECLARE B BYTE;
                    /* TRANSLATE ALPHA TO UPPER CASE */
411   2           IF B >= 110$0001B AND B <= 111$1010B THEN /* LOWER CASE */
412   2               B = B AND 101$1111B; /* TO UPPER CASE */
413   2           RETURN B;
414   2           END UTRAN;

415   1       LTRAN: PROCEDURE(B) BYTE;
416   2           DECLARE B BYTE;
                    /* TRANSLATE TO LOWER CASE ALPHA */
417   2           IF B >= 'A' AND B <= 'Z' THEN B = B OR 10$0000B; /* TO LOWER *
419   2           RETURN B;
420   2           END LTRAN;
```

```
421   1       GETSOURCEC: PROCEDURE BYTE;
                  /* READ NEXT SOURCE CHARACTER */
422   2           DECLARE (IOB,B,CONCHK) BYTE;

423   2           IF PSOURCE - 1 <= RDR THEN /* 1 ... RDR+1 */
424   2               DO; IF (BLOCK OR NEXT) AND CONBRK THEN
426   3                   DO;
427   4                   IF READCHAR = ENDFILE THEN RETURN ENDFILE;
429   4                   CALL PRINT(.('READER STOPPING',CR,LF,'$'));
430   4                   RETURN XOFF;
431   4                   END;
432   3               END;
433   2           CONCHK = TRUE; /* CONSOLE STATUS CHECK BELOW */
434   2           IOB = IOBYTE; /* SAVE IT IN CASE IT IS ALTERED */
435   2               DO CASE PSOURCE;
                    /* CASE 0 IS SOURCE FILE */
436   3                   DO; IF NSOURCE >= SBLEN THEN CALL FILLSOURCE;
439   4                   B = SBUFF(NSOURCE);
440   4                   NSOURCE = NSOURCE + 1;
441   4                   END;
                    /* CASE 1 IS INP */
442   3                   B = INP;
                    /* CASE 2 IS IRD (INTEL/ICOM) */
443   3                   B = INTIN;
                    /* CASE 3 IS PTR */
444   3                       DO; IOBYTE = 0000$0100B; GO TO RDRL;
447   4                       END;
                    /* CASE 4 IS UR1 */
448   3                       DO; IOBYTE = 0000$1000B; GO TO RDRL;
451   4                       END;
                    /* CASE 5 IS UR2 */
452   3                       DO; IOBYTE = 0000$1100B; GO TO RDRL;
455   4                       END;
                    /* CASE 6 IS RDR */
456   3                   RDRL:
                            B = MON2(3,0) AND 7FH;
                    /* CASE 7 IS OUT */
457   3                   GO TO NOTSOURCE;
                    /* CASE 8 IS LPT */
458   3                   GO TO NOTSOURCE;
                    /* CASE 9 IS UL1 */
459   3                   GO TO NOTSOURCE;
                    /* CASE 10 IS PRN */
460   3                   GO TO NOTSOURCE;
                    /* CASE 11 IS LST */
461   3                   GO TO NOTSOURCE;
                    /* CASE 12 IS PTP */
462   3                   GO TO NOTSOURCE;
                    /* CASE 13 IS UP1 */
463   3                   GO TO NOTSOURCE;
                    /* CASE 14 IS UP2 */
464   3                   GO TO NOTSOURCE;
                    /* CASE 15 IS PUN */
465   3                   NOTSOURCE:
                                DO; CALL ERROR(.('NOT A CHARACTER SOURCE$'));
467   4                           END;
```



```
                    /* CASE 16 IS TTY */
468   3                       DO; IOBYTE = 0; GO TO CONL;
471   4                       END;
                    /* CASE 17 IS CRT */
472   3                       DO; IOBYTE = 01B; GO TO CONL;
475   4                       END;
                    /* CASE 18 IS UC1 */
476   3                       DO; IOBYTE = 11B; GO TO CONL;
479   4                       END;
                    /* CASE 19 IS CON */
480   3                     CONL:
                            DO; CONCHK = FALSE; /* DON'T CHECK CONSOLE STATUS
482   4                       B = MON2(1,0);
483   4                       END;
484   3               END; /* OF CASES */
485   2           IOBYTE = IOB; /* RESTORE IOBYTE */
486   2           IF ECHO THEN /* COPY TO CONSOLE DEVICE */
487   2               DO; IOB = PDEST; PDEST = CONP; CALL PUTDEST(B);
491   3               PDEST = IOB;
492   3               END;
493   2           IF CONCHK THEN /* TEST FOR CONSOLE CHAR READY */
494   2               DO;
495   3               IF SCOM THEN /* SOURCE IS A COM FILE */
496   3                   CONCHK = (CONCNT := CONCNT + 1) = 0; ELSE /* ASCII */
497   3                   CONCHK = B = LF;
498   3               IF CONCHK THEN
499   3                   DO; IF CONBRK THEN
501   4                       DO;
502   5                       IF READCHAR = ENDFILE THEN RETURN ENDFILE;
504   5                       CALL ERROR(.('ABORTED$'));
505   5                       END;
506   4                   END;
507   3               END;
508   2           IF ZEROP THEN B = B AND 7FH;
510   2           IF UPPER THEN RETURN UTRAN(B);
512   2           IF LOWER THEN RETURN LTRAN(B);
514   2           RETURN B;
515   2           END GETSOURCEC;

516   1       GETSOURCE: PROCEDURE BYTE;
                  /* GET NEXT SOURCE CHARACTER */
517   2           DECLARE CHAR BYTE;
518   2           MATCH: PROCEDURE(B) BYTE;
                      /* MATCH START AND QUIT STRINGS */
519   3               DECLARE (B,C) BYTE;
520   3               IF (C:=COMBUFF(B:=(B+MATCHLEN))) = ENDFILE THEN /* END MAT
521   3                   DO; COMBUFF(B) = CHAR; /* SAVE CURRENT CHARACTER */
523   4                   RETURN TRUE;
524   4                   END;
525   3               IF C = CHAR THEN MATCHLEN = MATCHLEN + 1; ELSE
527   3                   MATCHLEN = 0; /* NO MATCH */
528   3               RETURN FALSE;
529   3               END MATCH;
530   2           IF QUITLEN > 0 THEN
531   2               DO; IF (QUITLEN := QUITLEN - 1) = 1 THEN RETURN LF;
534   3               RETURN ENDFILE; /* TERMINATED WITH CR,LF,ENDFILE */
535   3               END;
```

```
591   3             FCB(FLEN:=FLEN+1) = CHAR;
592   3             IF CHAR = WHAT THEN AMBIG = TRUE; /* CONTAINS AMBIGUOUS RE
594   3             END PUTCHAR;

595   2         FILLQ: PROCEDURE(LEN);
                    /* FILL CURRENT NAME OR TYPE WITH QUESTION MARKS */
596   3             DECLARE LEN BYTE;
597   3             CHAR = WHAT; /* QUESTION MARK */
598   3                 DO WHILE FLEN < LEN;
599   4                 CALL PUTCHAR;
600   4                 END;
601   3             END FILLQ;

602   2         GETFCB: PROCEDURE(I) BYTE;
603   3             DECLARE I BYTE;
604   3             RETURN FCB(I);
605   3             END GETFCB;

606   2         SCANPAR: PROCEDURE;
607   3             DECLARE (I,J) BYTE;
                    /* SCAN OPTIONAL PARAMETERS */
608   3             PARSET = TRUE;
609   3             SUSER = CUSER; /* SOURCE USER := CURRENT USER */
610   3             CHAR = GNC; /* SCAN PAST BRACKET */
611   3                 DO WHILE NOT(CHAR = CR OR CHAR = RB);
612   4                 IF (I := CHAR - 'A') > 25 THEN /* NOT ALPHA */
613   4                     DO; IF CHAR = ' ' THEN CHAR = GNC; ELSE
616   5                     CALL ERROR(.('BAD PARAMETER$'));
617   5                     END; ELSE
618   4                     DO; /* SCAN PARAMETER VALUE */
619   5                     IF CHAR = 'S' OR CHAR = 'Q' THEN
620   5                         DO; /* START OR QUIT COMMAND */
621   6                         J = CBP + 1; /* START OF STRING */
622   6                             DO WHILE NOT ((CHAR := GNC) = ENDFILE OR C
623 + 7                             END;
624   6                          CHAR=GNC;
625   6                         END; ELSE
626   5                     IF (J := (CHAR := GNC) - '0') > 9 THEN J = 1;
                            ELSE
628   5                         DO WHILE (K := (CHAR := GNC) - '0') <= 9;
629   6                         J = J * 10 + K;
630   6                         END;
631   5                     CONT(I) = J;
632   5                     IF I = 6 THEN /* SET SOURCE USER */
633   5                         DO;
634   6                         IF J > 31 THEN
635   6                             CALL ERROR(.('INVALID USER NUMBER$'));
636   6                         SUSER = J;
637   6                         END;
638   5                     END;
639   4                 END;
640   3             CHAR = GNC;
641   3             END SCANPAR;

642   2         CHKSET: PROCEDURE;
643   3             IF CHAR = LA THEN CHAR = '=';
645   3             END CHKSET;
```



```
536   2             DO FOREVER; /* LOOKING FOR START */
537   3             IF FEEDLEN = 0 THEN /* GET SEARCH CHARACTERS */
538   3                 DO; FEEDLEN = FEEDLEN - 1;
540   4                 CHAR = COMBUFF(FEEDBASE);
541   4                 FEEDBASE = FEEDBASE + 1;
542   4                 RETURN CHAR;
543   4                 END;
544   3             IF (CHAR := GETSOURCEC) = ENDFILE THEN RETURN ENDFILE;
546   3             IF STARTS > 0 THEN /* LOOKING FOR START STRING */
547   3                 DO; IF MATCH(STARTS) THEN
549   4                     DO; FEEDBASE = STARTS; STARTS = 0;
552   5                     FEEDLEN = MATCHLEN + 1;
553   5                     END; /* OTHERWISE NO MATCH, SKIP CHARACTER */
554   4                 END; ELSE
555   3             IF QUITS > 0 THEN /* PASS CHARACTERS TIL MATCH */
556   3                 DO; IF MATCH(QUITS) THEN
558   4                     DO; QUITS = 0; QUITLEN = 2;
                            /* SUBSEQUENTLY RETURN CR, LF, ENDFILE */
561   5                     RETURN CR;
562   5                     END;
563   4                 RETURN CHAR;
564   4                 END; ELSE
565   3             RETURN CHAR;
566   3             END; /* OF DO FOREVER */
567   2             END GETSOURCE;

568   1         DECLARE DISK BYTE;              /* SELECTED DISK */

569   1           GNC: PROCEDURE BYTE;
570   2             IF (CBP := CBP + 1) >= COMLEN THEN RETURN CR;
572   2             RETURN UTRAN(COMBUFF(CBP));
573   2             END GNC;

574   1           DEBLANK: PROCEDURE;
575   2                 DO WHILE (CHAR := GNC) = ' ';
576   3                 END;
577   2             END DEBLANK;

578   1           SCAN: PROCEDURE(FCBA);
579   2             DECLARE FCBA ADDRESS,           /* ADDRESS OF FCB TO FILL *
                      FCB BASED FCBA (FSIZE) BYTE;  /* FCB TEMPLATE */
580   2             DECLARE (I,J,K) BYTE; /* TEMP COUNTERS */

                    /* SCAN LOOKS FOR THE NEXT DELIMITER, DEVICE NAME, OR FILE NAME
                       THE VALUE OF CBP MUST BE 255 UPON ENTRY THE FIRST TIME */

581   2             DELIMITER: PROCEDURE(C) BYTE;
582   3                 DECLARE (I,C) BYTE;
583   3                 DECLARE DEL(*) BYTE DATA
                        (' .:,<>',CR,LA,LB,RB);
584   3                     DO I = 0 TO LAST(DEL);
585   4                     IF C = DEL(I) THEN RETURN TRUE;
587   4                     END;
588   3                 RETURN FALSE;
589   3                 END DELIMITER;

590   2             PUTCHAR: PROCEDURE;
P
```

```
                /* INITIALIZE FILE CONTROL BLOCK TO EMPTY */
646   2         AMBIG = FALSE; TYPE = ERR; CHAR = ' '; FLEN = 0;
650   2             DO WHILE FLEN < FSIZE-1;
651   3             IF FLEN = FNSIZE THEN CHAR = 0;
653   3             CALL PUTCHAR;
654   3             END;

                /* DEBLANK COMMAND BUFFER */
655   2             CALL DEBLANK;

                /* SAVE STARTING POSITION OF SCAN FOR DIAGNOSTICS */
656   2         TCBP = CBP;

                /* MAY BE A SEPARATOR */
657   2         IF DELIMITER(CHAR) THEN
658   2             DO; CALL CHKSET;
660   3             TYPE = SPECL; RETURN;
662   3             END;

                /* CHECK PERIPHERALS AND DISK FILES */
663   2         DISK = 0;
                /* CLEAR PARAMETERS */
664   2             DO I = 0 TO 25; CONT(I) = 0;
666   3             END;
667   2         PARSET = FALSE;
668   2         FEEDLEN,MATCHLEN,QUITLEN = 0;
                /* SCAN NEXT NAME */
669   2             DO FOREVER;
670   3             FLEN = 0;
671   3                 DO WHILE NOT DELIMITER(CHAR);
672   4                 IF FLEN >= NSIZE THEN /* ERROR, FILE NAME TOO LONG */
673   4                     RETURN;
674   4                 IF CHAR = '*' THEN CALL FILLQ(NSIZE); ELSE CALL PUTCHA
677   4                 CHAR = GNC;
678   4                 END;

                    /* CHECK FOR DISK NAME OR DEVICE NAME */
679   3                 IF CHAR = ':' THEN
680   3                     DO; IF DISK <> 0 THEN RETURN; /* ALREADY SET */
683   4                     IF FLEN = 1 THEN
                              /* MAY BE DISK NAME A ... Z */
684   4                         DO;
685   5                         IF (DISK := GETFCB(1) - 'A' + 1) > 26 THEN
686   5                         /* ERROR, INVALID DISK NAME */ RETURN;
687   5                         CALL DEBLANK; /* MAY BE DISK NAME ONLY */
688   5                         IF DELIMITER(CHAR) THEN
689   5                             DO; IF CHAR = LB THEN
691   6                                 CALL SCANPAR;
692   6                             CBP = CBP - 1;
693   6                             TYPE = DISKNAME;
694   6                             RETURN;
695   6                             END;
696   5                         END; ELSE

                              /* MAY BE A THREE CHARACTER DEVICE NAME */
697   4                     IF FLEN <> 3 THEN /* ERROR, CANNOT BE DEVICE NAME */
```



```
698   4                 RETURN; ELSE

                        /* LOOK FOR DEVICE NAME */
699   4                 DO; DECLARE (I,J,K) BYTE, M LITERALLY '20',
                        IO(*) BYTE DATA
                        ('INPIRDPTRUR1UR2RDROUTLPTUL1PRNLST',
                         'PTPUP1UP2PUNTTYCRTUC1CONNUL$OI',0);
                        /* NOTE THAT ALL READER-LIKE DEVICES MUST BE
                        PLACED BEFORE 'RDR', AND ALL LISTING-LIKE DEVICES
                        MUST APPEAR BELOW LST, BUT ABOVE RDR.  THE LITERAL
                        DECLARATIONS FOR RDR, LST, AND PUNP MUST INDICATE
                        THE POSITIONS OF THESE DEVICES IN THE LIST */
701   5                 J = 255;
702   5                     DO K = 0 TO M;
703   6                     I = 0;
704   6                         DO WHILE (I:=I+1) <= 3) AND
                                IO(J+I) = GETFCB(I);
705   7                         END;
706   6                     IF I = 4 THEN /* COMPLETE MATCH */
707   6                         DO; TYPE = PERIPH;
                                /* SCAN PARAMETERS */
709   7                         IF GNC = LB THEN CALL SCANPAR;
711   7                         CBP = CBP - 1; CHAR = K;
713   7                         RETURN;
714   7                         END;
715   6                     /* OTHERWISE TRY NEXT DEVICE */ J = J + 3;
716   6                     END;

717   5                 /* ERROR, NO DEVICE NAME MATCH */ RETURN;
718   5                 END;
719   4             IF CHAR = LB THEN /* PARAMETERS FOLLOW */
720   4                 CALL SCANPAR;
721   4             END; ELSE

                  /* CHAR IS NOT ':', SO FILE NAME IS SET. SCAN REMAINDER */
722   3             DO; IF FLEN = 0 THEN /* ERROR, NO PRIMARY NAME */
724   4                 RETURN;
725   4             FLEN = FNAM;
726   4             IF CHAR = '.' THEN /* SCAN FILE TYPE */
727   4                 DO WHILE NOT DELIMITER(CHAR := GNC);
728   5                 IF FLEN >= FNSIZE THEN
729   5                 /* ERROR, TYPE FIELD TOO LONG */ RETURN;
730   5                 IF CHAR = '*' THEN CALL FILLQ(FNSIZE);
732   5                     ELSE CALL PUTCHAR;
733   5                 END;

734   4             IF CHAR = LB THEN
735   4                 CALL SCANPAR;
                      /* RESCAN DELIMITER NEXT TIME AROUND */
736   4             CBP = CBP - 1;
737   4             TYPE = FILE;
                      /* DISK IS THE SELECTED DISK (1 2 3 ... ) */
738   4             IF DISK = 0 THEN DISK = CDISK + 1; /* DEFAULT */
740   4             FCB(0),FCB(32) = 0;
741   4             RETURN;
742   4             END;
743   3         END;
```

```
744   2           END SCAN;

745   1         NULLS: PROCEDURE;
                    /* SEND 40 NULLS TO OUTPUT DEVICE */
746   2           DECLARE I BYTE;
747   2               DO I = 0 TO 39; CALL PUTDEST(0);
749   3               END;
750   2           END NULLS;


751   1         DECLARE FEXTH(FEXTL) BYTE,      /* HOLDS DESTINATION FILE TYPE *
                    COPYING BYTE;               /* TRUE WHILE COPYING TO DEST FI

752   1         MOVEXT: PROCEDURE(A);
753   2           DECLARE A ADDRESS;
                    /* MOVE THREE CHARACTER EXTENT INTO DEST FCB */
754   2           CALL MOVE(A,.DEST(FEXT),FEXTL);
755   2           END MOVEXT;

756   1       EQUAL: PROCEDURE(A,B) BYTE;
                    /* COMPARE THE STRINGS AT A AND B UNTIL EITHER A MISMATCH OR
                    A '$' IS ENCOUNTERED IN STRING B */
757   2           DECLARE (A,B) ADDRESS,
                    (SA BASED A, SB BASED B) BYTE;
758   2               DO WHILE SB <> '$';
759   3               IF (SB AND 7FH) <> (SA AND 7FH) THEN RETURN FALSE;
761   3               A = A + 1; B = B + 1;
763   3               END;
764   2           RETURN TRUE;
765   2           END EQUAL;

766   1       READ$EOF: PROCEDURE BYTE;
                    /* RETURN TRUE IF END OF FILE */
767   2           CHAR = GETSOURCE;
768   2           IF SCOM THEN RETURN HARDEOF <= NSOURCE;
770   2           RETURN CHAR = ENDFILE;
771   2           END READ$EOF;


772   1       HEXRECORD: PROCEDURE BYTE;
                    /* READ ONE RECORD INTO SBUFF AND CHECK FOR PROPER FORM
                        RETURNS 0 IF RECORD OK
                        RETURNS 1 IF END OF TAPE (:00000)
                        RETURNS 2 IF ERROR IN RECORD        */


773   2           DECLARE XOFFSET BYTE; /* TRUE IF XOFF RECVD */
774   2           DECLARE NOERRS BYTE; /* TRUE IF NO ERRORS IN THIS RECORD *

775   2           PRINTERR: PROCEDURE(A);
                        /* PRINT ERROR MESSAGE IF NOERRS TRUE */
776   3               DECLARE A ADDRESS;
777   3               IF NOERRS THEN
778   3                   DO; NOERRS = FALSE;
780   4                   CALL PRINT(A);
781   4                   END;
782   3               END PRINTERR;
```



```
783   2           CHECKXOFF: PROCEDURE;
784   3               IF XOFFSET THEN
785   3                   DO; XOFFSET = FALSE;
787   4                   CALL CLEARBUFF;
788   4                   END;
789   3               END CHECKXOFF;

790   2           SAVECHAR: PROCEDURE BYTE;
                        /* READ CHARACTER AND SAVE IN BUFFER */
791   3               DECLARE I BYTE;
792   3               IF NOERRS THEN
793   3                   DO;
794   4                       DO WHILE (I := GETSOURCE) = XOFF; XOFFSET = TRUE;
796   5                       END;
797   4                   HBUFF(HSOURCE) = I;
798   4                   IF (HSOURCE := HSOURCE + 1) >= LAST(HBUFF) THEN
799   4                       CALL PRINTERR(.('RECORD TOO LONG$'));
800   4                   RETURN I;
801   4                   END;
802   3               RETURN ENDFILE; /* ON ERROR FLAG */
803   3               END SAVECHAR;

804   2           DECLARE (M, RL, CS, RT) BYTE,
                        LDA ADDRESS;  /* LOAD ADDRESS WHICH FOLLOWS : */

805   2           READHEX: PROCEDURE BYTE;
806   3               DECLARE H BYTE;
807   3               IF (H := SAVECHAR) - '0' <= 9 THEN RETURN H-'0';
809   3               IF H - 'A' > 5 THEN
810   3                   CALL PRINTERR(.('INVALID DIGIT$'));
811   3               RETURN H - 'A' + 10;
812   3               END READHEX;

813   2           READBYTE: PROCEDURE BYTE;
                        /* READ TWO HEX DIGITS */
814   3               RETURN SHL(READHEX,4) OR READHEX;
815   3               END READBYTE;

816   2           READCS: PROCEDURE BYTE;
                        /* READ BYTE WITH CHECKSUM */
817   3               RETURN CS := CS + READBYTE;
818   3               END READCS;

819   2           READADDR: PROCEDURE ADDRESS;
                        /* READ DOUBLE BYTE WITH CHECKSUM */
820   3               RETURN SHL(DOUBLE(READCS),8) OR READCS;
821   3               END READADDR;

822   2           NOERRS = TRUE; /* NO ERRORS DETECTED IN THIS RECORD */

                    /* READ NEXT RECORD */
                        /* SCAN FOR THE ':' */
823   2               HSOURCE = 0;
824   2                   DO WHILE (CS := SAVECHAR) <> ':';
825   3                   HSOURCE = 0;
826   3                   IF CS = ENDFILE THEN
```

```
827   3                   DO; CALL PRINT'.('END OF FILE, CTL-Z',WHAT,'$'));
829   4                   IF READCHAR = ENDFILE THEN RETURN 1;
831   4                      ELSE HSOURCE   0;
832   4                   END;
833   3               CALL CHECKXOFF;
834   3               END;

                /* ':' FOUND */
835   2           CS = 0;
836   2           IF (RL := READCS) = 0 THEN /* END OF TAPE */
837   2               DO; DO WHILE (RL := SAVECHAR) <> ENDFILE;
839   4                   CALL CHECKXOFF;
840   4                   END;
841   3               IF NOERRS THEN RETURN 1;
843   3               RETURN 2;
844   3               END;

            /* RECORD LENGTH IS NOT ZERO */
845   2           LDA = READADDR; /* LOAD ADDRESS */

            /* READ WORDS UNTIL RECORD LENGTH EXHAUSTED */
846   2           RT = READCS; /* RECORD TYPE */
847   2              DO WHILE RL <> 0 AND NOERRS; RL = RL - 1;
849   3              M   READCS;
                     /* INCREMENT LA HERE FOR EXACT ADDRESS */
850   3              END;

            /* CHECK SUM */
851   2           IF CS + READBYTE <> 0 THEN
852   2               CALL PRINTERR(.('CHECKSUM ERROR$'));

853   2           CALL CHECKXOFF;
854   2           IF NOERRS THEN RETURN 0;
856   2           RETURN 2;
857   2           END HEXRECORD;

858   1       READTAPE: PROCEDURE;
                /* READ HEX FILE FROM HIGH SPEED READER TO 'HEX' FILE,
                CHECK EACH RECORD FOR VALID DIGITS, AND PROPER CHECKSUM */
859   2           DECLARE (I,A) BYTE;
860   2               DO FOREVER;
861   3                   DO WHILE (I := HEXRECORD) <= 1;
862   4                   IF NOT (I = 1 AND IGNOR) THEN
863   4                       DO A = 1 TO HSOURCE;
864   5                       CALL PUTDEST(HBUFF(A-1));
865   5                       END;
866   4                   CALL PUTDEST(CR); CALL PUTDEST(LF);
868   4                   IF I = 1 THEN /* END OF TAPE ENCOUNTERED */
869   4                       RETURN;
870   4                   END;
871   3               CALL CRLF; HBUFF(HSOURCE) = '$';
873   3               CALL PRINT(.HBUFF);
874   3               CALL PRINT(.('CORRECT ERROR, TYPE RETURN OR CTL-Z$'));
875   3               CALL CRLF;
876   3               IF READCHAR = ENDFILE THEN RETURN;
878   3               END;
879   2           END READTAPE;
```



```
880   1       FORMERR: PROCEDURE;
881   2           CALL ERROR(.('INVALID FORMAT$'));
882   2           END FORMERR;

883   1       SETUPDEST: PROCEDURE;
884   2           CALL SELECT(DDISK);
885   2           IHEX = EQUAL(.DEST(FEXT),.('HEX$'));
886   2           CALL MOVE(.DEST(FEXT),.FEXTH,FEXTL); /* SAVE TYPE */
887   2           DEST(ROFILE) = DEST(ROFILE) AND 7FH;
888   2           DEST(SYSFILE)= DEST(SYSFILE)AND 7FH;
889   2           CALL MOVEXT(.('$$$'));
890   2           CALL DELETE(.DEST); /* REMOVE OLD $$$ FILE */
891   2           CALL MAKE(.DEST); /* CREATE A NEW ONE */
892   2           IF DCNT = 255 THEN CALL ERROR(.('NO DIRECTORY SPACE$'));
894   2           DEST(32),NDEST = 0;
895   2           END SETUPDEST;

896   1       SETUPSOURCE: PROCEDURE;
897   2           HARDEOF = 0FFFFH;
898   2           CALL SETSUSER; /* SOURCE USER */
899   2           CALL SELECT(SDISK);
900   2           CALL OPEN(.SOURCE);
901   2           CALL SETCUSER; /* BACK TO CURRENT USER */
902   2           IF (NOT RSYS) AND ROL(SOURCE(SYSFILE),1) THEN
903   2               DCNT = 255;
904   2           IF DCNT = 255 THEN CALL ERROR(.('NO FILE$'));
906   2           SOURCE(32) = 0;
                 /* CAUSE IMMEDIATE READ */
907   2           SCOM = EQUAL(.SOURCE(FEXT),.('COM$'));
908   2           NSOURCE   SBLEN;
909   2           END SETUPSOURCE;

910   1       CHECK$STRINGS: PROCEDURE;
911   2           IF STARTS > 0 THEN
912   2               CALL ERROR(.('START NOT FOUND$'));
913   2           IF QUITS  > 0 THEN
914   2               CALL ERROR(.('QUIT NOT FOUND$'));
915   2           END CHECK$STRINGS;

916   1       CLOSEDEST: PROCEDURE(DIRECT);
917   2           DECLARE DIRECT BYTE;
                 /* DIRECT IS TRUE IF SECTOR BY-SECTOR COPY */
918   2           IF DIRECT THEN
                     /* GET UNFILLED BYTES FROM SOURCE BUFFER */
919   2               DFUB = SFUB; ELSE DFUB = 0;
921   2               DO WHILE (LOW(NDEST) AND 7FH) <> 0;
922   3               DFUB = DFUB + 1;
923   3               CALL PUTDEST(ENDFILE);
924   3               END;
925   2           CALL CHECK$STRINGS;
926   2           CALL WRITEDEST;
927   2           CALL SELECT(DDISK);
928   2           CALL CLOSE(.DEST);
929   2           IF DCNT = 255 THEN
930   2               CALL ERROR(.('CANNOT CLOSE DESTINATION FILE$'));
931   2           CALL MOVEXT(.FEXTH); /* RECALL ORIGINAL TYPTE */
```

```
 932   2          CALL OPEN(.DEST);
 933   2          IF DCNT <> 255 THEN /* FILE EXISTS */
 934   2              DO;
 935   3              IF ROL(DEST(ROFILE),1) THEN /* READ ONLY */
 936   3                  DO;
 937   4                  IF NOT WRROF THEN
 938   4                      DO;
 939   5                      CALL PRINT (.('DESTINATION IS R/O. DELETE (Y/N)?$'
 940   5                      IF UTRAN(READCHAR) <> 'Y' THEN
 941   5                          DO; CALL PRINT(.('**NOT DELETED**$'));
 943   6                          CALL CRLF;
 944   6                          CALL MOVEXT'.('$$$'));
 945   6                          CALL DELETE(.DEST);
 946   6                          RETURN;
 947   6                          END;
 948   5                      CALL CRLF;
 949   5                      END;
 950   4                  DEST(ROFILE) = DEST(ROFILE) AND 7FH;
 951   4                  CALL SETIND(.DEST);
 952   4                  END;
 953   3              CALL DELETE(.DEST);
 954   3              END;
 955   2          CALL MOVE(.DEST,.DEST(16),16); /* READY FOR RENAME */
 956   2          CALL MOVEXT(.('$$$'));
 957   2          CALL RENAME(.DEST);
 958   2          END CLOSEDEST;

 959   1      SIZE$NBUF: PROCEDURE;
              /* COMPUTE NUMBER OF BUFFERS - 1 FROM DBLEN */
 960   2          NBUF = (SHR(DBLEN,7) AND 0FFH) - 1;
              /* COMPUTED AS DBLEN/128-1, WHERE DBLEN <= 32K (AND THUS
              NBUF RESULTS IN A VALUE <= 2**15/2**7-1 = 2**8-1 = 255) */
 961   2          END SIZE$NBUF;

 962   1      SET$DBLEN: PROCEDURE;
              /* ABSORB THE SOURCE BUFFER INTO THE DEST BUFFER */
 963   2          SBASE = .MEMORY;
 964   2          IF DBLEN >= 4000H THEN DBLEN = 7F80H; ELSE
 966   2              DBLEN = DBLEN + SBLEN;
 967   2          CALL SIZE$NBUF;
 968   2          END SET$DBLEN;

 969   1      SIZE$MEMORY: PROCEDURE;
              /* SET UP SOURCE AND DESTINATION BUFFERS */
 970   2          SBASE = .MEMORY + SHR(MEMSIZE - .MEMORY,1);
 971   2          SBLEN, DBLEN = SHR((MEMSIZE - .MEMORY) AND 0FF00H,1);
 972   2          CALL SIZE$NBUF;
 973   2          END SIZE$MEMORY;

 974   1      COPYCHAR: PROCEDURE;
              /* PERFORM THE ACTUAL COPY FUNCTION */
 975   2          DECLARE RESIZED BYTE; /* TRUE IF SBUFF AND DBUFF COMBINED */
 976   2          IF (RESIZED := (BLOCK AND PSOURCE <> 0)) THEN /* BLOCK MODE */
 977   2              CALL SET$DBLEN; /* ABSORB SOURCE BUFFER */
 978   2          IF HEXT OR IGNOR THEN /* HEX FILE */
 979   2              CALL READTAPE; ELSE
 980   2              DO WHILE NOT READ$EOF;
```



```
 981   3              CALL PUTDEST(CHAR);
 982   3              END;
 983   2          IF RESIZED THEN
 984   2              DO; CALL CLEARBUFF;
 986   3              CALL SIZE$MEMORY;
 987   3              END;
 988   2          END COPYCHAR;

 989   1      SIMPLECOPY: PROCEDURE;
 990   2          DECLARE (FASTCOPY,I) BYTE;
 991   2          REAL$EOF: PROCEDURE BYTE;
 992   3              RETURN HARDEOF <> 0FFFFH;
 993   3              END REALEOF;
 994   2          CALL SIZE$MEMORY;
 995   2          TCBP = MCBP; /* FOR ERROR TRACING */
 996   2          CALL SETUPDEST;
 997   2          CALL SETUPSOURCE;
              /* FILES READY FOR DIRECT COPY */
 998   2          FASTCOPY = TRUE;
                  /* LOOK FOR PARAMETERS */
 999   2              DO I = 0 TO 25;
1000   3              IF CONT(I) <> 0 THEN
1001   3                  DO;
1002   4                  IF NOT(I = 14 OR I = 21) THEN
                          /* NOT OBJ OR VERIFY */
1003   4                  FASTCOPY = FALSE;
1004   4                  END;
1005   3              END;
1006   2          IF FASTCOPY THEN /* COPY DIRECTLY TO DBUFF */
1007   2              DO; CALL SET$DBLEN; /* EXTEND DBUFF */
1009   3                  DO WHILE NOT REAL$EOF;
1010   4                  CALL FILLSOURCE;
1011   4                  IF REAL$EOF THEN
1012   4                      NDEST = HARDEOF; ELSE NDEST = DBLEN;
1014   4                  CALL WRITEDEST;
1015   4                  END;
1016   3              CALL SIZE$MEMORY; /* RESET TO TWO BUFFERS */
1017   3              END; ELSE
1018   2          CALL COPYCHAR;
1019   2          CALL CLOSEDEST(FASTCOPY);
1020   2          END SIMPLECOPY;

1021   1      MULTCOPY: PROCEDURE;
1022   2          DECLARE (NEXTDIR, NDONT, NCOPIED) BYTE;
1023   2          PRNAME: PROCEDURE;
                  /* PRINT CURRENT FILE NAME */
1024   3              DECLARE (I,C) BYTE;
1025   3              CALL CRLF;
1026   3                  DO I = 1 TO FNSIZE;
1027   4                  IF (C := DEST(I)) <> ' ' THEN
1028   4                      DO; IF I = FEXT THEN CALL PRINTCHAR('.');
1031   5                      CALL PRINTCHAR(C);
1032   5                      END;
1033   4                  END;
1034   3              END PRNAME;

1035   2          NEXTDIR,NCOPIED = 0;
P
```

```
1036  2             DO FOREVER;
                    /* FIND A MATCHING ENTRY */
1037  3             CALL SETSUSER; /* SOURCE USER */
1038  3             CALL SELECT(SDISK);
1039  3             CALL SETDMA(.BUFFER);
1040  3             CALL SEARCH(.SEARFCB);
1041  3             NDCNT = 0;
1042  3                 DO WHILE (DCNT <> 255) AND NDCNT < NEXTDIR;
1043  4                 NDCNT = NDCNT + 1;
1044  4                 CALL SEARCHN;
1045  4                 END;
1046  3             CALL SETCUSER;
                    /* FILE CONTROL BLOCK IN BUFFER */
1047  3             IF DCNT = 255 THEN
1048  3                 DO; IF NCOPIED = 0 THEN
1050  4                 CALL ERROR(.('NOT FOUND$'));  CALL CRLF;
1052  4                 RETURN;
1053  4                 END;
1054  3             NEXTDIR = NDCNT + 1;
                    /* GET THE FILE CONTROL BLOCK NAME TO DEST */
1055  3             CALL MOVE(.BUFFER+SHL(DCNT AND 11B,5),.DEST,16);
1056  3             CALL MOVE(.DEST,.SOURCE,16); /* FILL BOTH FCB'S */
1057  3             DEST(0) = 0;
1058  3             IF RSYS OR NOT ROL(DEST(SYSFILE),1) THEN /* OK TO READ */
1059  3                 DO;
1060  4                 IF (NCOPIED := NCOPIED + 1) = 1 THEN
1061  4                 CALL PRINT(.('COPYING -$'));
1062  4                 CALL PRNAME;
1063  4                 CALL SIMPLECOPY;
1064  4                 END;
1065  3             END;
1066  2         END MULTCOPY;

1067  1     SET$SDISK: PROCEDURE;
1068  2         IF DISK > 0 THEN SDISK = DISK - 1; ELSE SDISK = CDISK;
1071  2         END SET$SDISK;

1072  1     SET$DDISK: PROCEDURE;
1073  2         IF PARSET THEN /* PARAMETERS PRESENT */ CALL FORMERR;
1075  2         IF DISK > 0 THEN DDISK = DISK - 1; ELSE DDISK = CDISK;
1078  2         END SET$DDISK;

1079  1     CHECK$DISK: PROCEDURE;
1080  2         IF SUSER <> CUSER THEN /* DIFFERENT DISKS */
1081  2             RETURN;
1082  2         IF DDISK = SDISK THEN CALL FORMERR;
1084  2         END CHECK$DISK;

1085  1     CHECK$EOL: PROCEDURE;
1086  2         CALL DEBLANK;
1087  2         IF CHAR <> CR THEN CALL FORMERR;
1089  2         END CHECK$EOL;

1090  1     SCANDEST: PROCEDURE(COPYFCB);
1091  2         DECLARE COPYFCB ADDRESS;
1092  2         CALL SET$SDISK;
1093  2         CALL CHECK$EOL;
```



```
1094  2         CALL MOVE(.SOURCE,COPYFCB,33);
1095  2         CALL CHECK$DISK;
1096  2         END SCANDEST;

1097  1     SCANEQL: PROCEDURE;
1098  2         CALL SCAN(.SOURCE);
1099  2         IF NOT (TYPE = SPECL AND CHAR = '=') THEN CALL FORMERR;
1101  2         MCBP = CBP; /* FOR ERROR PRINTING */
1102  2         END SCANEQL;


1103  1     PIPENTRY:
            /* BUFFER AT 80H CONTAINS REMAINDER OF LINE TYPED
            FOLLOWING THE COMMAND 'PIP' - IF ZERO THEN PROMPT TIL CR */
            CALL MOVE(.BUFF,.COMLEN,80H);
1104  1     MULTCOM = COMLEN = 0;

            /* GET CURRENT CP/M VERSION */
1105  1     IF VERSION < CPMVERSION THEN
1106  1         DO;
1107  2         CALL PRINT(.('REQUIRES CP/M 2.0 OR NEWER FOR OPERATION.$'));
1108  2         CALL BOOT;
1109  2         END;
            /* GET CURRENT USER */
1110  1     CUSER = GETUSER;
            /* GET CURRENT DISK */
1111  1     CDISK = MON2(25,0);

1112  1     RETRY:
            /* ENTER HERE ON ERROR EXIT FROM THE PROCEDURE 'ERROR' */
              SUSER = CUSER; /* SOURCE AND CURRENT USER SAME */
1113  1       CALL SIZE$MEMORY;
              /* MAIN PROCESSING LOOP.  PROCESS UNTIL CR ONLY */
1114  1       DO FOREVER;
1115  2       C1, C2, C3 = 0; /* LINE COUNT = 000000 */
1116  2       PUTNUM = TRUE; /* ACTS LIKE LF OCCURRED ON ASCII FILE */
1117  2       CONCNT,COLUMN = 0; /* PRINTER TABS */
1118  2       LINENO = 254; /* INCREMENTED TO 255 > PAGCNT */
              /* READ FROM CONSOLE IF NOT A ONELINER */
1119  2       IF MULTCOM THEN
1120  2           DO; CALL PRINTCHAR('*'); CALL READCOM;
1123  3           CALL CRLF;
1124  3           END;
1125  2       CBP = 255;
1126  2       IF COMLEN = 0 THEN /* SINGLE CARRIAGE RETURN */
1127  2           DO; CALL SELECT(CDISK);
1129  3           CALL BOOT;
1130  3           END;

              /* LOOK FOR SPECIAL CASES FIRST */
1131  2       DDISK,SDISK,PSOURCE,PDEST = 0;
1132  2         CALL SCAN(.DEST);
1133  2         IF TYPE = PERIPH THEN GO TO SIMPLECOM;
1135  2         IF TYPE = DISKNAME THEN
1136  2             DO; DDISK = DISK - 1;
1138  3             CALL SCANEQL;
1139  3             CALL SCAN(.SOURCE);
```

```
                /* MAY BE MULTI COPY */
1140  3         IF TYPE <> FILE THEN CALL FORMERR;
1142  3         IF AMBIG THEN
1143  3             DO; CALL SCANDEST(.SEARCB);
1145  4             CALL MULTCOPY;
1146  4             END; ELSE
1147  3             DO; CALL SCANDEST(.DEST);
                      /* FORM IS A:=B:UFN */
1149  4             CALL SIMPLECOPY;
1150  4             END;
1151  3         GO TO ENDCOM;
1152  3         END;


1153  2       IF TYPE <> FILE OR AMBIG THEN CALL FORMERR;
1155  2         CALL SET$DDISK;
1156  2         CALL SCANEQL;
1157  2         CALL SCAN(.SOURCE);
1158  2         IF TYPE = DISKNAME THEN
1159  2             DO;
1160  3             CALL SET$SDISK; CALL CHECK$DISK;
1162  3             CALL MOVE(.DEST,.SOURCE,33);
1163  3             CALL CHECK$EOL;
1164  3             CALL SIMPLECOPY;
1165  3             GO TO ENDCOM;
1166  3             END;
                  /* MAY BE POSSIBLE TO DO A FAST DISK COPY */
1167  2         IF TYPE = FILE THEN /* FILE TO FILE */
1168  2             DO; CALL DEBLANK; IF CHAR <> CR THEN GO TO SIMPLECOM;
                      /* FILE TO FILE */
1172  3             CALL SET$SDISK;
1173  3             CALL SIMPLECOPY;
1174  3             GO TO ENDCOM;
1175  3             END;

1176  2     SIMPLECOM:
              CBP = 255; /* READY FOR RESCAN */

              /* OTHERWISE PROCESS SIMPLE REQUEST */
1177  2         CALL SCAN(.DEST);
1178  2         IF (TYPE < FILE) OR AMBIG THEN /* DELIMITER OR ERROR */
1179  2             CALL ERROR(.('UNRECOGNIZED DESTINATION$'));

1180  2         DHEX = FALSE;
1181  2         IF TYPE = FILE THEN
1182  2             DO; /* DESTINATION IS A FILE, SAVE EXTENT NAME */
1183  3             CALL SET$DDISK;
1184  3             CALL SETUPDEST;
1185  3             CHAR = 255;
1186  3             END; ELSE
                  /* PERIPHERAL NAME */
1187  2         IF CHAR >= NULP OR CHAR <= RDR THEN CALL ERROR(.('CANNOT WRITE

              IF (PDEST := CHAR + 1) = PUNP THEN CALL NULLS;

              /* NOW SCAN THE DELIMITER */
1191  2         CALL SCAN(.SOURCE);
```



```
1192  2         IF TYPE <> SPECL OR CHAR <> '=' THEN
1193  2             CALL ERROR(.('INVALID PIP FORMAT$'));

              /* OTHERWISE SCAN AND COPY UNTIL CR */
1194  2         COPYING = TRUE;
1195  2         DO WHILE COPYING;
1196  3         SUSER = CUSER;
1197  3         CALL SCAN(.SOURCE);
                  /* SUSER MAY HAVE BEEN RESET */
1198  3         SCOM = FALSE;
1199  3         IF TYPE = FILE AND NOT AMBIG THEN /* A SOURCE FILE */
1200  3             DO;
1201  4             CALL SET$SDISK;
1202  4             CALL SETUPSOURCE;
1203  4             CHAR = 255;
1204  4             END; ELSE

1205  3             IF TYPE <> PERIPH OR (CHAR <= LST AND CHAR > RDR) THEN
1206  3                 CALL ERROR(.('CANNOT READ$'));


                SCOM = SCOM OR OBJ; /* MAY BE ABSOLUTE COPY */
1208  3         PSOURCE = CHAR + 1;
1209  3         IF CHAR = NULP THEN CALL NULLS; ELSE
1211  3         IF CHAR = EOFP THEN CALL PUTDEST(ENDFILE); ELSE
1213  3             DO; /* DISK COPY */
1214  4             IF (CHAR < HSRDR AND DHEX) THEN HEXT = 1;
                      /* HEX FILE SET IF SOURCE IS RDR AND DEST IS HEX FILE */
1216  4             IF PDEST = PRNT THEN
1217  4                 DO; NUMB = 1;
1219  5                 IF TABS = 0 THEN TABS = 8;
1221  5                 IF PAGCNT = 0 THEN PAGCNT = 1;
1223  5                 END;
1224  4             CALL COPYCHAR;
1225  4             END;

1226  3             CALL CHECK$STRINGS;
                      /* READ ENDFILE, GO TO NEXT SOURCE */
1227  3             CALL SCAN(.SOURCE);
1228  3             IF TYPE <> SPECL OR (CHAR <> ',' AND CHAR <> CR) THEN
1229  3                 CALL ERROR(.('INVALID SEPARATOR$'));

1230  3             COPYING = CHAR <> CR;
1231  3             END;

              /* IF NECESSARY, CLOSE FILE OR PUNCH TRAILER */
1232  2         IF PDEST = PUNP THEN
1233  2             DO; CALL PUTDEST(ENDFILE); CALL NULLS;
1236  3             END;
1237  2         IF PDEST = 0 THEN /* FILE HAS TO BE CLOSED AND RENAMED */
1238  2             CALL CLOSEDEST(FALSE);

              /* COMLEN SET TO 0 IF NOT PROCESSING MULTIPLE COMMANDS */
1239  2     ENDCOM:
              COMLEN = MULTCOM;

1240  2         END; /* DO FOREVER */

1241  1     END;
```

```
0000 PIP#
0000 PIPMOD#
07E6 14     07FA 16     07F2 17     07F3 18     07F3 19
07FB 20     07FF 21     07FF 22     07FF 23     0804 24
0809 25     080A 42     080A 43     0813 44     0813 45
0813 46     081C 47     081C 49     0820 51     082D 52
082E 53     082E 54     0833 55     0838 56     0839 57
083F 59     0842 60     084B 61     084C 63     084C 64
0855 65     0855 66     0855 67     085D 68     085E 69
0862 71     086D 72     086E 73     0874 75     0880 76
0881 77     0887 79     0893 80     0894 81     089A 83
08A6 84     08A7 85     08A7 86     08B2 87     08B3 88
08B9 90     08C2 91     08C3 92     08C9 94     08D3 95
08D3 96     08D9 98     08E3 99     08E3 100    08E9 102
08F5 103    08F6 104    08FC 106    0905 107    0906 109
090C 111    0915 112    0916 113    0916 114    091F 115
091F 116    0923 118    092E 119    092F 120    092F 121
0936 122    0937 123    0937 124    093E 125    093F 126
0945 128    094F 129    094F 130    0955 132    095F 133
095F 134    0965 136    096F 137    096F 140    096F 141
0974 142    097C 143    097D 145    097D 146    0986 147
0986 149    098C 151    0995 152    0996 153    0996 155
099A 156    099E 157    09A7 158    09AA 159    09AF 160
09AF 162    09B5 164    09B8 165    09C0 166    09C5 167
09CA 168    09DA 169    09E4 170    09F1 171    09F8 172
09FD 173    0A03 174    0A0F 175    0A11 176    0A14 177
0A17 178    0A18 179    0A27 182    0A33 183    0A3D 184
0A44 185    0A4B 186    0A4E 187    0A4F 188    0A4F 190
0A55 191    0A5C 192    0A5F 193    0A6E 194    0A7B 195
0A89 197    0A91 198    0A97 199    0A9D 200    0AA4 201
0AAA 202    0AAD 203    0AB7 204    0ABE 205    0AC4 206
0AC7 207    0AC8 208    0ACB 212    0ADA 213    0ADB 214
0AE1 215    0AE8 216    0AEE 217    0AFD 218    0B07 219
0B0F 220    0B1A 221    0B20 222    0B2A 223    0B31 224
0B38 226    0B3E 227    0B44 228    0B53 229    0B61 230
0B68 231    0B6D 232    0B7B 233    0B9B 234    0B9F 235
0BA2 236    0BAC 237    0BB3 238    0BB9 239    0BC0 240
0BC9 241    0BC9 242    0BCF 243    0BD0 244    0BD4 246
0BDC 248    0BE0 249    0BE9 251    0BF3 252    0BF4 253
0BF4 254    0BF4 255    0BFA 256    0C0A 257    0C0A 258
0C16 259    0C19 260    0C24 261    0C2B 262    0C2E 263
0C31 264    0C34 265    0C37 266    0C3A 267    0C3D 268
0C46 269    0C50 270    0C50 271    0C55 272    0C58 273
0C5B 274    0C5B 275    0C60 276    0C63 277    0C66 278
0C66 279    0C6B 280    0C6E 281    0C71 282    0C7F 283
0C7F 284    0C84 285    0C87 286    0C8A 287    0C8A 288
0C8F 289    0C92 290    0C95 291    0C95 292    0C9A 293
0C9D 294    0CA0 295    0CAE 296    0CAE 297    0CB3 298
0CB6 299    0CB9 300    0CB9 301    0CBE 302    0CC1 303
0CC4 304    0CC4 305    0CC9 306    0CCC 307    0CCF 308
0CDD 309    0D05 310    0D0B 311    0D0C 312    0D10 314
0D18 315    0D22 316    0D2A 317    0D34 319    0D3A 320
0D44 321    0D4E 322    0D51 323    0D59 324    0D62 325
0D66 326    0D6B 327    0D6E 328    0D6E 329    0D76 330
0D7B 331    0D7C 332    0D80 334    0D91 335    0D99 336
0DA2 337    0DA3 338    0DA7 340    0DB4 341    0DBD 342
0DBE 343    0DBE 345    0DC3 346    0DCE 347    0DD6 348
0DDF 349    0DE8 350    0DEF 351    0DF6 352    0DFD 353
0E05 355    0E0A 356    0E0F 357    0E12 358    0E17 359
0E18 360    0F18 363    0E21 364    0E2A 365    0F2D 366
0E3C 367    0F44 368    0E45 369    0E49 371    0E50 373
0E58 374    0E59 375    0E59 376    0E60 378    0E68 380
0E73 382    0E7B 383    0EB0 384    0EBE 386    0E93 387
0E98 388    0E98 389    0E98 390    0EA1 391    0EA4 392
0EA9 393    0EA9 394    0EA9 395    0EB0 397    0EC8 399
0ECB 400    0ECC 401    0ECC 402    0ECC 403    0ED4 404
0ED9 405    0EE0 406    0EEB 407    0EED 408    0EEE 409
0EF2 411    0F09 412    0F11 413    0F15 414    0F15 415
0F19 417    0F30 418    0F3B 419    0F3C 420    0F3C 421
0F3C 423    0F47 425    0F59 427    0F61 428    0F64 429
0F6A 430    0F6D 431    0F6D 432    0F6D 433    0F72 434
0F7B 435    0F88 436    0F88 437    0F94 438    0F97 439
0FA3 440    0FAA 441    0FAD 442    0FB6 443    0FBF 444
0FBF 445    0FC4 446    0FC7 447    0FCA 448    0FCA 449
0FCF 450    0FD2 451    0FD5 452    0FD5 453    0FDA 454
0FDD 455    0FE0 456    0FF0 457    0FF3 458    0FF6 459
0FF9 460    0FFC 461    0FFF 462    1002 463    1005 464
1008 465    1008 466    100E 467    1011 468    1011 469
1016 470    1019 471    101C 472    101C 473    1021 474
1024 475    1027 476    1027 477    102C 478    102F 479
1032 480    1032 481    1037 482    1042 483    1045 484
106D 485    1073 486    107A 488    1080 489    1085 490
108C 491    1092 492    1092 493    1099 495    10A0 496
10B2 497    10BD 498    10C4 500    10CB 502    10D3 503
10D6 504    10DC 505    10DC 506    10DC 507    10DC 508
10E3 509    10EB 510    10F2 511    10FA 512    1101 513
1109 514    110D 515    110D 516    11AD 518    11B1 520
11C9 522    11D6 523    11D9 524    11D9 525    11E3 526
11EA 527    11FF 528    11F2 529    110D 530    1116 532
1122 533    1125 534    1128 535    1128 536    1128 537
1131 539    1135 540    1141 541    1145 542    1146 543
1146 544    1151 545    1154 546    115D 548    1168 550
116E 551    1173 552    117A 553    117A 554    117D 555
1186 557    1191 559    1196 560    119B 561    119E 562
119E 563    11A2 564    11A5 565    11A9 566    11AC 567
11F2 569    11F2 570    1200 571    1203 572    1211 573
1211 574    1211 575    121C 576    121F 577    1220 578
1438 581    143C 584    144A 585    145A 586    145D 587
1464 586    1467 589    1467 590    1467 591    1479 592
1481 593    1486 594    1487 595    148B 597    1490 598
149A 599    149D 600    14A0 601    14A1 602    14A5 604
14B1 605    14B1 606    14B1 608    14B6 609    14BC 610
14C2 611    14DA 612    14E9 614    14F1 615    14FA 616
1500 617    1503 619    151B 621    1522 622    153D 623
1540 624    1546 625    1549 626    155B 627    1563 628
1575 629    158A 630    158D 631    159A 632    15A2 634
15AB 635    15B1 636    15B7 637    15B7 638    15B7 639
15BA 640    15C0 641    15C1 642    15C1 643    15C9 644
15CE 645    1226 646    122B 647    1230 648    1233 649
1238 650    1240 651    1248 652    124D 653    1250 654
1253 655    1256 656    125C 657    1267 659    126A 660
126F 661    1270 662    1270 663    1275 664    1283 665
128E 666    1295 667    129A 668    12A5 669    12A5 670
12AA 671    12B5 672    12BD 673    12BE 674    12C6 675
12CE 676    12D1 677    12D7 678    12DA 679    12E2 681
12EA 682    12EE 683    12F3 685    1305 686    1306 687
1309 688    1314 690    131C 691    131F 692    1323 693
1328 694    1329 695    1329 696    132C 697    1334 698
1335 701    133A 702    1346 703    134B 704    137B 705
137E 706    1386 708    138B 709    1393 710    1396 711
```

```
139A 712    13A0 713    13A1 714    13A1 715    13A9 716
13B0 717    13B1 718    13B1 719    1389 720    13BC 721
13BF 723    13C7 724    13C8 725    13CD 726    13D5 727
13E3 728    13EB 729    13EC 730    13F4 731    13FC 732
13FF 733    1402 734    140A 735    140D 736    1411 737
1415 738    141E 739    1425 740    1433 741    1434 742
1434 743    1437 744    15CF 745    15CF 747    15DD 748
15E2 749    15E9 750    15EA 752    15F0 754    15FC 755
15FD 756    1607 758    1610 759    1623 760    1626 761
162D 762    1634 763    1637 764    163A 765    163A 766
163A 767    1640 768    1647 769    1653 770    165C 771
165C 772    1713 775    1719 777    1720 779    1725 780
172D 781    172D 782    172E 783    172E 784    1735 786
173A 787    173D 788    173D 789    173E 790    173E 792
1745 794    1750 795    1755 796    1758 797    1765 798
1771 799    1777 800    177B 801    177B 802    177E 803
177E 805    177E 807    178D 808    1793 809    179F 810
17A5 811    17AD 812    17AD 813    17AD 814    17BC 815
17BC 816    17BC 817    17C5 818    17C5 819    17C5 820
17DB 821    165C 822    1661 823    1666 824    1671 825
1676 826    167E 828    1684 829    168C 830    168F 831
1694 832    1694 833    1697 834    169A 835    169F 836
16AA 838    16B5 839    16B8 840    16BB 841    16C2 842
16C5 843    16C8 844    16C8 845    16CF 846    16D4 847
16E4 848    16E8 849    16EE 850    16F1 851    16FD 852
1703 853    1706 854    170D 855    1710 856    1713 857
17DB 858    17DB 860    17DB 861    17E8 862    17F8 863
1807 864    1816 865    181D 866    1822 867    1827 868
182F 869    1830 870    1833 871    1836 872    1841 873
1847 874    184D 875    1850 876    1858 877    1859 878
185C 879    185D 880    185D 881    1863 882    1864 883
1864 884    186B 885    1877 886    1883 887    188B 888
1893 889    1899 890    189F 891    18A5 892    18AD 893
18B3 894    18BE 895    18BF 896    18BF 897    18C5 898
18C8 899    18CF 900    18D5 901    18D8 902    18E8 903
18ED 904    18F5 905    18FB 906    1900 907    190C 908
1912 909    1913 910    1913 911    191C 912    1922 913
192B 914    1931 915    1932 916    1936 918    193D 919
1946 920    194B 921    1956 922    195A 923    195F 924
1962 925    1965 926    1968 927    196F 928    1975 929
197D 930    1983 931    1989 932    198F 933    1997 935
199F 937    19A6 939    19AC 940    19B8 942    19BE 943
19C1 944    19C7 945    19CD 946    19CE 947    19CE 948
19D1 949    19D1 950    19D9 951    19DF 952    19DF 953
19E5 954    19E5 955    19F1 956    19F7 957    19FD 958
19FE 959    19FE 960    1A11 961    1A12 962    1A12 963
1A18 964    1A24 965    1A2D 966    1A38 967    1A3B 968
1A3C 969    1A3C 970    1A52 971    1A64 972    1A67 973
1A68 974    1A68 976    1A7B 977    1A7E 978    1A89 979
1A8F 980    1A96 981    1A9D 982    1AA0 983    1AA7 985
1AAA 986    1AAD 987    1AAD 988    1AAE 989    1B42 991
1B42 992    1B50 993    1AAE 994    1AB1 995    1AB7 996
1ABA 997    1ABD 998    1AC2 999    1ACE 1000   1ADD 1002
1AF5 1003   1AFA 1004   1AFA 1005   1B01 1006   1B08 1008
1B0B 1009   1B12 1010   1B15 1011   1B1C 1012   1B25 1013
1B2B 1014   1B2E 1015   1B31 1016   1B34 1017   1B37 1018
1B3A 1019   1B41 1020   1B50 1021   1C0F 1023   1C0F 1025
1C12 1026   1C20 1027   1C32 1029   1C3A 1030   1C3F 1031
1C46 1032   1C46 1033   1C4D 1034   1B50 1035   1B5A 1036
1B5A 1037   1B5D 1038   1B64 1039   1B6A 1040   1B70 1041

1B75 1042   1B8D 1043   1B91 1044   1B94 1045   1B97 1046
1B9A 1047   1BA2 1049   1BAA 1050   1BB0 1051   1BB3 1052
1BB4 1053   1BB4 1054   1BBB 1055   1BD5 1056   1BE1 1057
1BE6 1058   1BF3 1060   1BFF 1061   1C05 1062   1C08 1063
1C0B 1064   1C0B 1065   1C0E 1066   1C4F 1067   1C4F 1068
1C57 1069   1C61 1070   1C67 1071   1C68 1072   1C68 1073
1C6F 1074   1C72 1075   1C7B 1076   1C85 1077   1C8B 1078
1C8C 1079   1C8C 1080   1C96 1081   1C97 1082   1CA1 1083
1CA4 1084   1CA5 1085   1CA5 1086   1CA8 1087   1CB0 1088
1CB3 1089   1CB4 1090   1CBA 1092   1CBD 1093   1CC0 1094
1CCE 1095   1CD1 1096   1CD2 1097   1CD2 1098   1CD8 1099
1CF0 1100   1CF3 1101   1CF9 1102   04CE 1103   04DD 1104
04E8 1105   04F4 1107   04FA 1108   04FD 1109   04FD 1110
0503 1111   050E 1112   0517 1113   051A 1114   051A 1115
0525 1116   052A 1117   0532 1118   0535 1119   053C 1121
0541 1122   0544 1123   0547 1124   0547 1125   054C 1126
0554 1128   055B 1129   055E 1130   055F 1131   0570 1132
0576 1133   057E 1134   0581 1135   0589 1137   0590 1138
0593 1139   0599 1140   05A1 1141   05A4 1142   05AB 1144
05B1 1145   05B4 1146   05B7 1148   05BD 1149   05C0 1150
05C0 1151   05C3 1152   05C3 1153   05D3 1154   05D6 1155
05D9 1156   05DC 1157   05E2 1158   05EA 1160   05ED 1161
05F0 1162   05FC 1163   05FF 1164   0602 1165   0605 1166
0605 1167   060D 1169   0610 1170   0618 1171   061B 1172
061E 1173   0621 1174   0624 1175   0624 1176   0629 1177
062F 1178   063D 1179   0643 1180   0648 1181   0650 1183
0653 1184   0656 1185   065B 1186   065E 1187   0675 1188
067B 1189   0687 1190   068A 1191   0690 1192   06A8 1193
06AE 1194   06B3 1195   06BA 1196   06C0 1197   06C6 1198
06CB 1199   06DF 1201   06E2 1202   06E5 1203   06EA 1204
06ED 1205   070D 1206   0713 1207   071B 1208   0722 1209
072A 1210   0730 1211   0738 1212   0740 1214   074E 1215
0753 1216   075B 1218   0760 1219   0768 1220   076D 1221
0775 1222   077A 1223   077A 1224   077D 1225   077D 1226
0780 1227   0786 1228   07AA 1229   07B0 1230   07BB 1231
07BE 1232   07C6 1234   07CB 1235   07CE 1236   07CE 1237
07D6 1238   07DB 1239   07E1 1240   07E4 1241
0000 MODULE#
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE SUB
OBJECT MODULE PLACED IN SUBMIT.OBJ
COMPILER INVOKED BY:  PLM80 SUBMIT.PLM DEBUG


   1          sub:
              do;
              /* modified 7/26/79 to work with cpm 2.0, module number not zero *
   2    1     declare
                  wboot literally '0000h',   /* warm start entry point */
                  bdos  literally '0005h',   /* jmp bdos */
                  dfcba literally '005ch',   /* default fcb address */
                  dbuff literally '0080h';   /* default buffer address */

   3    1     declare jump byte data(0c3h); /* c3 = jmp */
   4    1     declare jadr address data(.submit);
              /* jmp to submit is placed at the beginning of the module */

   5    1     boot: procedure external;
                  /* system reboot */
   6    2           end boot;

   7    1     mon1: procedure(f,a) external;
   8    2           declare f byte, a address;
                    /* bdos interface, no returned value */
   9    2           end mon1;

  10    1     mon2: procedure(f,a) byte external;
  11    2           declare f byte, a address;
                    /* bdos interface, return byte value */
  12    2           end mon2;


  13    1     declare
                  copyright(*) byte data
                      ( ' copyright(c) 1977, digital research ');

  14    1     declare
                  ln(5) byte initial('001 $'),
                  ln1 byte at(.ln(0)),
                  ln2 byte at(.ln(1)),
                  ln3 byte at(.ln(2)),
                  dfcb(33) byte initial(0,'$$$     SUB',0,0,0),
                  drec byte at(.dfcb(32)),  /* current record */
                  buff(128) byte at(dbuff),   /* default buffer */
                  sfcb(33)  byte at(dfcba);   /* default fcb */

  15    1     submit: procedure;

                  /*  t h e    c p / m   's u b m i t'   f u n c t i o n

                      copyright (c) 1976, 1977, 1978
                      digital research
                      box 579
                      pacific grove, ca.
                      93950
```



```
                      */
  16    2     declare lit literally 'literally',
                  dcl lit 'declare',
                  proc lit 'procedure',
                  addr lit 'address',
                  ctll lit '0ch',
                  lca  lit '110$0001b',  /* lower case a */
                  lcz  lit '111$1010b',  /* lower case z */
                  endfile lit '1ah';     /* cp/m end of file */

  17    2     declare
                  true literally '1',
                  false literally '0',
                  forever literally 'while true',
                  cr literally '13',
                  lf literally '10',
                  what literally '63';

  18    2     print: procedure(a);
  19    3         declare a address;
                  /* print the string starting at address a until the
                  next dollar sign is encountered */
  20    3         call mon1(9,a);
  21    3         end print;


  22    2     declare dcnt byte;

  23    2     open: procedure(fcb);
  24    3         declare fcb address;
  25    3         dcnt = mon2(15,fcb);
  26    3         end open;

  27    2     close: procedure(fcb);
  28    3         declare fcb address;
  29    3         dcnt = mon2(16,fcb);
  30    3         end close;

  31    2     delete: procedure(fcb);
  32    3         declare fcb address;
  33    3         call mon1(19,fcb);
  34    3         end delete;

  35    2     diskread: procedure(fcb) byte;
  36    3         declare fcb address;
  37    3         return mon2(20,fcb);
  38    3         end diskread;

  39    2     diskwrite: procedure(fcb) byte;
  40    3         declare fcb address;
  41    3         return mon2(21,fcb);
  42    3         end diskwrite;

  43    2     make: procedure(fcb);
  44    3         declare fcb address;
  45    3         dcnt = mon2(22,fcb);
  46    3         end make;
```

```
 47   2      move: procedure(s,d,n);
 48   3          declare (s,d) address, n byte;
 49   3          declare a based s byte, b based d byte;
 50   3              do while (n := n - 1) <> 255;
 51   4              b = a; s = s + 1; d = d + 1;
 54   4              end;
 55   3          end move;

 56   2      declare oldsp address;   /* calling program's stack pointer */

 57   2      error: procedure(a);
 58   3          declare a address;
 59   3          call print(.(cr,lf,'$'));
 60   3          call print(.('Error On Line $'));
 61   3          call print(.ln1);
 62   3          call print(a);
 63   3          stackptr   oldsp;
              /* return to ccp */
 64   3          end error;

 65   2      declare sstring(128) byte, /* substitute string */
              sbp byte;                  /* source buffer pointer (0-128) */


 66   2      setup: procedure;
              /* move buffer to substitute string */
 67   3          call move(.buff(1),.sstring(0),127);
 68   3          sstring(buff(0))=0; /* mark end of string */
 69   3          call move(.('SUB'),.sfcb(9),3); /* set file type to sub */
 70   3          call open(.sfcb(0));
 71   3          if dcnt = 255 then
 72   3              call error(.('No ''SUB'' File Present$'));
              /* otherwise file is open - read subsequent data */
 73   3          sbp = 128; /* causes read below */

 74   3          end setup;


 75   2      getsource: procedure byte;
              /* read the next source character */
 76   3          declare b byte;
 77   3          if sbp > 127 then
 78   3              do; if diskread(.sfcb(0)) <> 0 then
 80   4                  return endfile;
 81   4              sbp = 0;
 82   4              end;
 83   3          if (b := buff((sbp:=sbp+1)-1)) = cr then
 84   3              do; /* increment line */
 85   4              if (ln3 := ln3 + 1) > '9' then
 86   4                  do; ln3 = '0';
 88   5                  if (ln2 := ln2 + 1) > '9' then
 89   5                      do; ln2 = '0';
 91   6                      ln1 = ln1 + 1;
 92   6                      end;
 93   5                  end;
 94   4              end;
```



```
              /* translate to upper case */
 95   3          if (b-61h) < 26 then /* lower case alpha */
 96   3              b = b and 5fh; /* change to upper case */
 97   3          return b;
 98   3          end getsource;

 99   2      writebuff: procedure;
              /* write the contents of the buffer to disk */
100   3          if diskwrite(.dfcb) <> 0 then /* error */
101   3              call error(.('Disk Write Error$'));
102   3          end writebuff;

103   2      declare rbuff(2048) byte, /* jcl buffer */
              rbp address,       /* jcl buffer pointer */
              rlen byte;      /* length of current command */

104   2      fillrbuff: procedure;
105   3          declare (s,ssbp) byte;  /* sub string buffer pointer */

106   3          notend: procedure byte;
                  /* look at next character in sstring, return
                  true if not at the end of the string - char passed
                  back in 's' */
107   4              if not ((s := sstring(ssbp)) = ' ' or s = 0) then
108   4                  do;
109   5                  ssbp = ssbp + 1;
110   5                  return true;
111   5                  end;
112   4              return false;
113   4              end notend;

114   3          deblankparm: procedure;
                  /* clear to next non blank substitute string */
115   4                  do while sstring(ssbp) = ' ';
116   5                  ssbp = ssbp + 1;
117   5                  end;
118   4              end deblankparm;

119   3          putrbuff: procedure(b);
120   4             declare b byte;
121   4              if (rbp := rbp + 1) > last(rbuff) then
122   4                  call error(.('Command Buffer Overflow$'));
123   4              rbuff(rbp) = b;
                  /* len: cl ... c125 :00:$ = 128 chars */
124   4              if (rlen := rlen + 1) > 125 then
125   4                  call error(.('Command Too Long$'));
126   4              end putrbuff;

127   3          declare (reading,b) byte;
              /* fill the jcl buffer */
128   3          rbuff(0),rbp = 0;
129   3          reading   true;
130   3              do while reading;
131   4              rlen   0; /* reset command length */
132   4                  do while (b:=getsource) <> endfile and b <> cr;
133   5                  if b <> lf then
134   5                      do; if b = '$' then /* copy substitute string */
```

```
136   6                       do; if (b:=getsource) = '$' then
                                  /* $$ replaced by $ */
138   7                           call putrbuff(b); else
139   7                       if (b := b - '0') > 9 then
140   7                           call error(.('Parameter Error$')); else
141   7                           do; /* find string 'b' in sstring */
142   8                               ssbp = 0; call deblankparm; /* ready to sca
144   8                               do while b <> 0; b = b - 1;
                                      /* clear next parameter */
146   9                                   do while notend;
147  10                                   end;
148   9                               call deblankparm;
149   9                               end;
                                  /* ready to copy substitute string from pos
150   8                               do while notend;
151   9                               call putrbuff(s);
152   9                               end;
153   8                           end;
154   7                       end; else /* not a '$' */
155   6                   if b = '^' then /* control_character */
156   6                       do; /* must be 'a ... 'z */
157   7                       if (b:=getsource - 'a') > 25 then
158   7                           call error(.('Invalid Control Character$')
                                else
159   7                           call putrbuff(b+1);
160   7                       end; else /* not $ or ^ */
161   6                   call putrbuff(b);
162   6                   end;
163   5               end; /* of line or input file - compute length */
164   4           reading = b = cr;
165   4           call putrbuff(rlen); /* store length */
166   4           end;
            /* entire file has been read and processed */
167   3       end fillrbuff;

168   2   makefile: procedure;
            /* write resulting command file */
169   3       declare i byte;
170   3       getrbuff: procedure byte;
171   4           return rbuff(rbp := rbp - 1);
172   4           end getrbuff;

173   3       call delete(.dfcb);
174   3       drec = 0; /* zero the next record to write */
175   3       call make(.dfcb);
176   3       if dcnt = 255 then call error(.('Directory Full$'));
178   3           do while (i := getrbuff) <> 0;
                    /* copy i characters to buffer */
                    /* 00 $ at end of line gives 1.3 & 1.4 compatibility */
179   4               buff(0) = i; buff(i+1) = 00; buff(i+2) = '$';
182   4                   do while i > 0;
183   5                   buff(i) = getrbuff; i=i-1;
185   5                   end;
                    /* buffer filled to $ */
186   4               call writebuff;
187   4               end;
188   3           call close(.dfcb);
```



```
189   3           if dcnt = 255 then call error(.('Cannot Close, Read/Only?$'));
191   3           end makefile;

                /* enter here from the ccp with the fcb set */
192   2       declare stack(10) address;  /* working stack */
193   2           oldsp = stackptr;
194   2           stackptr = .stack(length(stack));

195   2           call setup;
196   2           call fillrbuff;
197   2           call makefile;
198   2           call boot; /* reboot causes commands to be executed */
199   2           end submit;
200   1       end;
```

```
0000 SUBMIT#
0000 SUB#
01DF 15     01F7 18     01FD 20     0206 21     0207 23
020D 25     0219 26     021A 27     0220 29     022C 30
022D 31     0233 33     023C 34     023D 35     0243 37
024D 38     024D 39     0253 41     025D 42     025D 43
0263 45     026F 46     0270 47     027F 50     028B 51
0295 52     029C 53     02A3 54     02A6 55     02A7 57
02AD 59     02B3 60     02B9 61     02BF 62     02C7 63
02CB 64     02CC 66     02CC 67     02D8 68     02E3 69
02EF 70     02F5 71     02FD 72     0303 73     0308 74
0309 75     0309 77     0312 79     031D E0     0320 81
0325 82     0325 83     033D 85     034B 87     0350 88
035B 90     0360 91     0362 92     0362 93     0362 94
0362 95     036C 96     0374 97     0378 98     0378 99
0378 100    0383 101    0389 102    039A 104    04B1 106
0481 107    04A3 109    04A7 110    04AA 111    04AA 112
04AD 113    04AD 114    04AD 115    04BC 116    04C0 117
04C3 118    04C4 119    04C8 121    04D8 122    04DE 123
04E9 124    04F7 125    04FD 126    030A 128    0395 129
039A 130    03A1 131    03A6 132    03C1 133    03C9 135
03D1 137    03DC 138    03E6 139    03F5 140    03FE 142
0403 143    0406 144    040E 145    0412 146    0419 147
041C 148    041F 149    0422 150    0429 151    0430 152
0433 153    0433 154    0436 155    043E 157    044D 158
0456 159    045E 160    0461 161    0468 162    0468 163
046B 164    0476 165    047D 166    0480 167    04FE 168
057A 170    057A 171    0587 172    04FE 173    0504 174
0509 175    050F 176    0517 177    051D 178    0528 179
052E 180    0537 181    0542 182    054B 183    0558 184
055C 185    055F 186    0562 187    0565 188    056B 189
0573 190    0579 191    01DF 193    01E6 194    01EA 195
01ED 196    01F0 197    01F3 198    01F6 199
0000 MODULE#
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE STAT
OBJECT MODULE PLACED IN STAT.OBJ
COMPILER INVOKED BY:  PLM80 STAT.PLM DEBUG


   1          stat:
              do;
   2    1     declare
                 cpmversion literally '20h'; /* requires 2.0 cp/m */
                 /* c p / m   s t a t u s    c o m m a n d  (s t a t) */

              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */
              /* status    status    status    status    status    status */

              /*
                          copyright(c) 1975, 1976, 1977, 1978, 1979
                          digital research
                          box 579
                          pacific grove, ca
                          93950
                */

              /* modified 10/30/78 to fix the space computation */
              /* modified 01/28/79 to remove despool dependencies */
              /* modified 07/26/79 to operate under cp/m 2.0 */

   3    1     declare jump byte data(0c3h),
                    jadr address data (.status);
                    /* jump to status */

              /* function call 32 returns the address of the disk parameter
              block for the currently selected disk, which consists of:
                      scptrk      (2 by) number of sectors per track
                      blkshf      (1 by) log2 of blocksize  2**blkshf=blksize
                      blkmsk      (1 by) 2**blkshf-1
```



```
                      extmsk      (1 by) logical/physical extents
                      maxall      (2 by) max alloc number
                      dirmax      (2 by) size of directory-1
                      dirblk      (2 by) reservation bits for directory
                      chksiz      (2 by) size of checksum vector
                      offset      (2 by) offset for operating system
              */

   4    1     declare
                 /* fixed locations for cp/m */
                 bdosa literally '0006h',     /* bdos base */
                 buffa literally '0080h',     /* default buffer */
                 fcba  literally '005ch',     /* default file control block */
                 dolla literally '006dh',     /* dollar sign position */
                 parma literally '006eh',     /* parameter, if sent */
                 rreca literally '007dh',     /* random record 7d,7e,7f */
                 rreco literally '007fh',     /* high byte of random overflow */
                 ioba  literally '0003h',     /* iobyte address */
                 sectorlen literally '128',   /* sector length */
                 memsize address at(bdosa),   /* end of memory */
                 rrec address at(rreca),      /* random record address */
                 rovf byte at(rreco),         /* overflow on getfile */
                 doll byte at(dolla),         /* dollar parameter */
                 parm byte at(parma),         /* parameter */
                 sizeset byte,                /* true if displaying size field */
                 dpba address,                /* disk parameter block address */
                 dpb based dpba structure
                 (spt address, bls byte, bms byte, exm byte, mxa address,
                  dmx address, dbl address, cks address, ofs address),
                 scptrk literally 'dpb.spt',
                 blkshf literally 'dpb.bls',
                 blkmsk literally 'dpb.bms',
                 extmsk literally 'dpb.exm',
                 maxall literally 'dpb.mxa',
                 dirmax literally 'dpb.dmx',
                 dirblk literally 'dpb.dbl',
                 chksiz literally 'dpb.cks',
                 offset literally 'dpb.ofs';

   5    1     boot: procedure external;
                 /* reboot */
   6    2     end boot;

   7    1     mon1: procedure(f,a) external;
   8    2        declare f byte, a address;
   9    2        end mon1;

  10    1     mon2: procedure(f,a) byte external;
  11    2        declare f byte, a address;
  12    2        end mon2;

  13    1     mon3: procedure(f,a) address external;
  14    2        declare f byte, a address;
  15    2        end mon3;
```

```
16   1      status: procedure;
17   2          declare copyright(*) byte data (
                '   Copyright (c) 1979, Digital Research');
               /* dummy outer procedure 'status' will start at 100h */
               /* determine status of currently selected disk */

18   2      declare alloca address,
               /* alloca is the address of the disk allocation vector */
               alloc based alloca (1024) byte;  /* allocation vector */

19   2      declare
               true literally '1',
               false literally '0',
               forever literally 'while true',
               cr literally '13',
               lf literally '10';

20   2      printchar: procedure(char);
21   3          declare char byte;
22   3          call mon1(2,char);
23   3          end printchar;

24   2      crlf: procedure;
25   3          call printchar(cr);
26   3          call printchar(lf);
27   3          end crlf;

28   2      printb: procedure;
               /* print blank character */
29   3          call printchar(' ');
30   3          end printb;

31   2      printx: procedure(a);
32   3          declare a address;
33   3          declare s based a byte;
34   3              do while s <> 0;
35   4              call printchar(s);
36   4              a = a + 1;
37   4              end;
38   3          end printx;

39   2      print: procedure(a);
40   3          declare a address;
               /* print the string starting at address a until the
               next 0 is encountered */
41   3          call crlf;
42   3          call printx(a);
43   3          end print;

44   2      break: procedure byte;
45   3          return mon2(11,0); /* console ready */
46   3          end break;

47   2      declare dcnt byte;

48   2      version: procedure byte;
               /* returns current cp/m version # */
```



```
49   3          return mon2(12,0);
50   3          end version;

51   2      select: procedure(d);
52   3          declare d byte;
53   3          call mon1(14,d);
54   3          end select;

55   2      open: procedure(fcb);
56   3          declare fcb address;
57   3          dcnt = mon2(15,fcb);
58   3          end open;

59   2      search: procedure(fcb);
60   3          declare fcb address;
61   3          dcnt = mon2(17,fcb);
62   3          end search;

63   2      searchn: procedure;
64   3          dcnt = mon2(18,0);
65   3          end searchn;

66   2      cselect: procedure byte;
               /* return current disk number */
67   3          return mon2(25,0);
68   3          end cselect;

69   2      setdma: procedure(dma);
70   3          declare dma address;
71   3          call mon1(26,dma);
72   3          end setdma;

73   2      getalloca: procedure address;
               /* get base address of alloc vector */
74   3          return mon3(27,0);
75   3          end getalloca;

76   2      getlogin: procedure address;
               /* get the login vector */
77   3          return mon3(24,0);
78   3          end getlogin;

79   2      writeprot: procedure;
               /* write protect the current disk */
80   3          call mon1(28,0);
81   3          end writeprot;

82   2      getrodisk: procedure address;
               /* get the read-only disk vector */
83   3          return mon3(29,0);
84   3          end getrodisk;

85   2      setind: procedure;
               /* set file indicators for current fcb */
86   3          call mon1(30,fcba);
87   3          end setind;
```

```
 88   2      set$dpb: procedure;
                 /* set disk parameter block values */
 89   3          dpba = mon3(31,0); /* base of dpb */
 90   3          end set$dpb;

 91   2      getuser: procedure byte;
                 /* return current user number */
 92   3          return mon2(32,0ffh);
 93   3          end getuser;

 94   2      setuser: procedure(user);
 95   3          declare user byte;
 96   3          call mon1(32,user);
 97   3          end setuser;

 98   2      getfilesize: procedure(fcb);
 99   3          declare fcb address;
100   3          call mon1(35,fcb);
101   3          end getfilesize;

102   2      declare oldsp address,    /* sp on entry */
                  stack(16) address;   /* this program's stack */

103   2      declare
                  fcbmax literally '512', /* max fcb count */
                  fcbs literally 'memory',/* remainder of memory */
                  fcb(33) byte at (fcba), /* default file control block */
                  buff(128) byte at (buffa),          /* default buffer */
                  ioval byte at (ioba);               /* io byte */

104   2      declare bpb address; /* bytes per block */

105   2      set$bpb: procedure;
106   3          call set$dpb; /* disk parameters set */
107   3          bpb = shl(double(1),blkshf) * sectorlen;
108   3          end set$bpb;

109   2      select$disk: procedure(d);
110   3          declare d byte;
                  /* select disk and set bpb */
111   3          call select(d);
112   3          call set$bpb; /* bytes per block */
113   3          end select$disk;

114   2      getalloc: procedure(i) byte;
                  /* return the ith bit of the alloc vector */
115   3          declare i address;
116   3          return
                  rol(alloc(shr(i,3)), (i and 111b) + 1);
117   3          end getalloc;

118   2      declare
                  accum(4) byte,     /* accumulator */
                  ibp byte;          /* input buffer pointer */

119   2      compare: procedure(a) byte;
                  /* compare accumulator with four bytes addressed by a */
```



```
120   3          declare a address;
121   3          declare (s based a) (4) byte;
122   3          declare i byte;
123   3              do i = 0 to 3;
124   4              if s(i) <> accum(i) then return false;
126   4              end;
127   3          return true;
128   3          end compare;

129   2      scan: procedure;
                  /* fill accum with next input value */
130   3          declare (i,b) byte;
131   3          setacc: procedure(b);
132   4              declare b byte;
133   4              accum(i) = b; i = i + 1;
135   4              end setacc;
                  /* deblank input */
136   3              do while buff(ibp) = ' '; ibp=ibp+1;
138   4              end;
                  /* initialize accum length */
139   3          i = 0;
140   3              do while i < 4;
141   4              if (b := buff(ibp)) > 1 then /* valid */
142   4                  call setacc(b); else /* blank fill */
143   4                  call setacc(' ');
144   4              if b <= 1 or b = ',' or b = ':' or
                       b = '*' or b = '.' or b = '>' or
                       b = '<' or b = '=' then buff(ibp) = 1;
                      else
146   4                     ibp = ibp + 1;
147   4                 end;
148   3          ibp = ibp + 1;
149   3          end scan;

150   2      pdecimal: procedure(v,prec);
                  /* print value v with precision prec (10,100,1000)
                  with leading zero suppression */
151   3          declare
                      v address,     /* value to print */
                      prec address, /* precision */
                      zerosup byte, /* zero suppression flag */
                      d byte;       /* current decimal digit */
152   3          zerosup = true;
153   3          do while prec <> 0;
154   4          d = v / prec ; /* get next digit */
155   4          v = v mod prec;/* get remainder back to v */
156   4          prec = prec / 10; /* ready for next digit */
157   4          if prec <> 0 and zerosup and d = 0 then call printb; else
159   4              do; zerosup = false; call printchar('0'+d);
162   5              end;
163   4          end;
164   3          end pdecimal;

165   2      add$block: procedure(ak,ab);
166   3          declare (ak, ab) address;
                  /* add one block to the kilobyte accumulator */
167   3          declare kaccum based ak address; /* kilobyte accum */
```

```
168   3          declare baccum based ab address; /* byte accum */
169   3          baccum = baccum + bpb;
170   3              do while baccum > 1024;
171   4              baccum = baccum - 1024;
172   4              kaccum = kaccum + 1;
173   4              end;
174   3          end add$block;

175   2      count: procedure(mode) address;
176   3          declare mode byte; /* true if counting 0's */
                  /* count kb remaining, kaccum set upon exit */
177   3          declare
                      ka  address,  /* kb accumulator */
                      ba  address,  /* byte accumulator */
                      i   address,  /* local index */
                      bit byte;     /* always 1 if mode = false */
178   3          ka, ba = 0;
179   3          bit = 0;
180   3              do i = 0 to maxall;
181   4              if mode then bit = getalloc(i);
182   4              if not bit then call add$block(.ka,.ba);
185   4              end;
186   3          return ka;
187   3          end count;

188   2      abortmsg: procedure;
189   3          call print(.('** Aborted **',0));
190   3          end abortmsg;

191   2      userstatus: procedure;
              /* display active user numbers */
192   3          declare i byte;
193   3          declare user(32) byte;
194   3          declare ufcb(*) byte data ('???????????',0,0,0);
195   3          call print(.('Active User :',0));
196   3          call pdecimal(getuser,10);
197   3          call print(.('Active Files:',0));
198   3              do i = 0 to last(user);
199   4              user(i) = false;
200   4              end;
201   3          call setdma(.fcbs);
202   3          call search(.ufcb);
203   3              do while dcnt <> 255;
204   4              if (i := fcbs(shl(dcnt and 11b,5))) <  0e5h then
205   4                  user(i and 1fh) = true;
206   4              call searchn;
207   4              end;
208   3              do i = 0 to last(user);
209   4              if user(i) then call pdecimal(i,10);
211   4              end;
212   3          end userstatus;

213   2      drivestatus: procedure;
214   3          declare
                      rpb address,
                      rpd address;
215   3          pv: procedure(v);
```



```
216   4              declare v address;
217   4              call crlf;
218   4              call pdecimal(v,10000);
219   4              call printchar(':');
220   4              call printb;
221   4              end pv;
              /* print the characteristics of the currently selected drive */
222   3          call print(.('      ',0));
223   3          call printchar(cselect+'A');
224   3          call printchar(':');
225   3          call printx(.(' Drive Characteristics',0));
226   3          rpb = shl(double(i),blkshf); /* records/block=2**blkshf */
227   3          if (rpd := (maxall+1) * rpb) = 0 and (rpb <> 0) then
228   3              call print(.('65536: ',0)); else
229   3              call pv(rpd);
230   3              call printx(.('128 Byte Record Capacity',0));
231   3          call pv(count(false));
232   3              call printx(.('Kilobyte Drive  Capacity',0));
233   3          call pv(dirmax+1);
234   3              call printx(.('32  Byte Directory Entries',0));
235   3          call pv(shl(chksiz,2));
236   3              call printx(.('Checked  Directory Entries',0));
237   3          call pv((extmsk+1) * 128);
238   3              call printx(.('Records/ Extent',0));
239   3          call pv(rpb);
240   3              call printx(.('Records/ Block',0));
241   3          call pv(scptrk);
242   3              call printx(.('Sectors/ Track',0));
243   3          call pv(offset);
244   3              call printx(.('Reserved Tracks',0));
245   3          call crlf;
246   3          end drivestatus;

247   2      diskstatus: procedure;
              /* display disk status */
248   3          declare login address, d byte;
249   3          login = getlogin; /* login vector set */
250   3          d = 0;
251   3              do while login <> 0;
252   4              if low(login) then
253   4                  do; call select$disk(d);
255   5                  call drivestatus;
256   5                  end;
257   4              login = shr(login,1);
258   4              d = d + 1;
259   4              end;
260   3          end diskstatus;

261   2      match: procedure(va,vl) byte;
              /* return index+1 to vector at va if match */
262   3          declare va address,
                      v based va (16) byte,
                      vl byte;
263   3          declare (i,j,match,sync) byte;
264   3          j,sync = 0;
265   3              do sync = 1 to vl;
266   4              match = true;
```

```
267   4                 do i = 0 to 3;
268   5                 if v(j) <> accum(i) then match=false;
270   5                 j = j + 1;
271   5                 end;
272   4               if match then return sync;
274   4               end;
275   3           return 0; /* no match */
276   3           end match;

277   2       declare devl(*) byte data
                    ('CON:RDR:PUN:LST:DEV:VAL:USR:DSK:');

278   2       devreq: procedure byte;
                    /* process device request, return true if found */
                    /* device tables */
279   3           declare
                        devr(*) byte data
                        (/* console */ 'TTY:CRT:BAT:UC1:',
                         /* reader  */ 'TTY:PTR:UR1:UR2:',
                         /* punch   */ 'TTY:PTP:UP1:UP2:',
                         /* listing */ 'TTY:CRT:LPT:UL1:');

280   3           declare
                        (i,j,iobyte,items) byte;

281   3           prname: procedure(a);
282   4               declare a address,
                            x based a byte;
                        /* print device name at a */
283   4               do while x <> ':';
284   5               call printchar(x); a=a+1;
286   5               end;
287   4               call printchar(':');
288   4               end prname;

289   3           items = 0;
290   3           do forever;
291   4           call scan;
292   4           if (i:=match(.devl,8)) = 0 then return items<>0;
294   4           items = items+1; /* found first/next item */
295   4           if i = 5 then /* device status request */
296   4               do;
297   5               iobyte = ioval; j = 0;
299   5                   do i = 0 to 3;
300   6                   call prname(.devl(shl(i,2)));
301   6                   call printx(.(' is ',0));
302   6                   call prname(.devr(shl(iobyte and 11b,2)+j));
303   6                   j = j + 16; iobyte = shr(iobyte,2);
305   6                   call crlf;
306   6                   end;
307   5               end; else /* not dev: */
308   4               if i = 6 then /* list possible assignment */
309   4                   do;
310   5                   call print(.('Temp R/O Disk: d:=R/O',0));
311   5                   call print(.('Set Indicator: d:filename.typ',
                                '$R/O $R/W $SYS $DIR',0));
```



```
312   5                   call print(.('Disk Status  : DSK: d:DSK:',0));
313   5                   call print(.('User Status  : USR:',0));
314   5                   call print(.('Iobyte Assign:',0));
315   5                       do i = 0 to 3;  * each line shows one device */
316   6                       call crlf;
317   6                       call prname(.devl(shl(i,2)));
318   6                       call printx(.(' =',0));
319   6                           do j = 0 to 12 by 4;
320   7                           call printchar(' ');
321   7                           call prname(.devr(shl(i,4)+j));
322   7                           end;
323   6                       end;
324   5                   end; else
325   4               if i = 7 then /* list user status values */
326   4                   do; call userstatus;
328   5                   return true;
329   5                   end; else
330   4           if i = 8 then /* show the disk device status */
331   4                   call diskstatus; else
                        /* scan item i-1 in device table */
332   4                   do; /* find base of destination */
333   5                   j = shl(i:=i-1,4);
334   5                   call scan;
335   5                   if accum(0) <> '=' then
336   5                       do; call print(.('Bad Delimiter',0));
338   6                       return true;
339   6                       end;
340   5                   call scan;
341   5                   if (j:=match(.devr(j),4)-1) = 255 then
342   5                       do; call print(.('Invalid Assignment',0));
344   6                       return true;
345   6                       end;
346   5                   iobyte = 1111$1100b; /* construct mask */
347   5                       do while (i:=i-1) <> 255;
348   6                       iobyte = rol(iobyte,2);
349   6                       j = shl(j,2);
350   6                       end;
351   5                   ioval = (ioval and iobyte) or j;
352   5                   end;
                        /* end of current item, look for more */
353   4               call scan;
354   4               if accum(0) = ' ' then return true;
356   4               if accum(0) <> ',' then
357   4                   do; call print(.('Bad Delimiter',0));
359   5                   return true;
360   5                   end;
361   4               end; /* of do forever */
362   3           end devreq;

363   2       pvalue: procedure(v);
364   3           declare (d,zero) byte,
                        (k,v) address;
365   3           k = 10000;
366   3           zero = false;
367   3               do while k <> 0;
368   4               d = low(v/k); v = v mod k;
370   4               k = k / 10;
```

```
371  4              if zero or k <> 0 or d < 0 then
372  4                   do; zero = true; call printchar('0'+d);
375  5                   end;
376  4              end;
377  3          call printchar('k');
378  3          call crlf;
379  3          end pvalue;

380  2      comp$alloc: procedure;
381  3          alloca  getalloca;
382  3          call printchar(cselect+'A');
383  3          call printx(.(': ',0));
384  3          end comp$alloc;

385  2      prcount: procedure;
           /* print the actual byte count */
386  3          call pvalue(count(true));
387  3          end prcount;

388  2      pralloc: procedure;
           /* print allocation for current disk */
389  3          call print (.('Bytes Remaining On ',0));
390  3          call comp$alloc;
391  3          call prcount;
392  3          end pralloc;

393  2      prstatus: procedure;
           /* print the status of the disk system */
394  3          declare (login, rodisk) address;
395  3          declare d byte;
396  3          login = getlogin; /* login vector set */
397  3          rodisk = getrodisk; /* read only disk vector set */
398  3          d = 0;
399  3              do while login <> 0;
400  4              if low(login) then
401  4                  do; call select$disk(d);
403  5                  call comp$alloc;
404  5                  call printx(.('R/',0));
405  5                  if low(rodisk) then
406  5                      call printchar('O'); else
407  5                      call printchar('W');
408  5                  call printx(.(', Space: ',0));
409  5                  call prcount;
410  5                  end;
411  4              login = shr(login,1); rodisk = shr(rodisk,1);
413  4              d = d + 1;
414  4              end;
415  3          call crlf;
416  3          end prstatus;

417  2      setdisk: procedure;
418  3          if fcb(0) <> 0 then call select$disk(fcb(0)-1);
420  3          end setdisk;

421  2      getfile: procedure;
           /* process file request */
```



```
422  3          declare
                    fnam literally '11',   fext literally '12',
                    fmod literally '14',
                    frc  literally '15',   fln  literally '15',
                    fdm  literally '16',   fdl  literally '31',
                    ftyp literally '9',
                    rofile literally '9', /* read/only file */
                    infile literally '10';  /* invisible file */
423  3          declare
                    fcbn address,    /* number of fcb's collected so far */
                    finx(fcbmax) address, /* index vector used during sort */
                    fcbe(fcbmax) address, /* extent counts */
                    fcbb(fcbmax) address, /* byte count (mod kb) */
                    fcbk(fcbmax) address, /* kilobyte count */
                    fcbr(fcbmax) address, /* record count */
                    bfcba address,     /* index into directory buffer */
                    fcbsa address,     /* index into fcbs */
                    bfcb based bfcba (32) byte,   /* template over directory *
                    fcbv based fcbsa (16) byte;   /* template over fcbs entry
424  3          declare
                    i address,      /* fcb counter during collection and displa
                    l address,      /* used during sort and display */
                    k address,      /*  "  */
                    m address,      /*  "  */
                    kb byte,        /* byte counter */
                    lb byte,        /* byte counter */
                    mb byte,        /* byte counter */
                    (b,f) byte,     /* counters */
                    matched byte;   /* used during fcbs search */

425  3          multi16: procedure;
                    /* utility to compute fcbs address from i */
426  4              fcbsa = shl(i,4) + .fcbs;
427  4              end multi16;

428  3          declare
                    scase byte; /* status case # */

429  3          declare
                    fstatlist(*) byte data('R/O',0,'R/W',0,'SYS',0,'DIR',0);

430  3          setfilestatus: procedure byte;
                        /* eventually, scase set r/o=0,r/w=1,dat=2,sys=3 */
431  4              declare
                        fstat(*) byte data('R/O R/W SYS DIR ');
432  4              if doll = ' ' then return false;
434  4              call move(4,.parm,.accum); /* $???? */
435  4              if accum(0) = 'S' and accum(1) = ' ' then
436  4                  return not (sizeset := true);
                    /* must be a parameter */
437  4              if (scase := match(.fstat,4)) = 0 then
438  4                  call print(.('Invalid File Indicator',0));
439  4              return true;
440  4              end setfilestatus;

441  3          printfn: procedure;
442  4              declare (k, lb) byte;
```

```
                /* print file name */
443   4             do k = 1 to fnam;
444   5             if (lb := fcbv(k) and 7fh) <> ' ' then
445   5                 do; if k = ftyp then call printchar('.');
448   6                 call printchar(lb);
449   6                 end;
450   5             end;
451   4         end printfn;

452   3     call set$bpb; /* in case default dis */
453   3     call setdisk;
454   3     sizeset = false;
455   3     scase = 255;
456   3     if setfilestatus then
457   3         do; if scase = 0 then return;
460   4         scase = scase - 1;
461   4         end; else
462   3     if fcb(1) = ' ' then /* no file name */
463   3         do; call pralloc;
465   4         return;
466   4         end;
              /* read the directory, collect all common file names */
              fcbn,fcb(0) = 0;
468   3     fcb(fext),fcb(fmod) = '?'; /* question mark matches all */
469   3     call search(fcba); /* fill directory buffer */
470   3     collect: /* label for debug */
              do while dcnt <> 255;
              /* another item found, compare it for common entry */
471   4         bfcba = shl(dcnt and 11b,5)+buffa; /* dcnt mod 4 * 32 */
472   4         matched = false; i = 0;
474   4             do while not matched and i < fcbn;
                  /* compare current entry */
475   5             call multi16;
476   5                 do kb = 1 to fnam;
477   6                 if bfcb(kb) <> fcbv(kb) then kb = fnam; else
                  /* complete match if at end */
479   6                     matched = kb = fnam;
480   6                 end;
481   5             i = i + 1;
482   5             end;
483   4     checkmatched: /* label for debug */
              if matched then i = i - 1; else
485   4             do; /* copy to new position in fcbs */
486   5             fcbn = (i := fcbn) + 1;
487   5             call multi16;
                  /* fcbsa set to next to fill */
488   5             if (fcbn > fcbmax) or (fcbsa + 16) > memsize then
489   5                 do; call print(.('** Too Many Files **',0));
491   6                 i = 0; fcbn = 1;
493   6                 call multi16;
494   6                 end;
                  /* save index to element for later sort */
495   5             finx(i) = i;
496   5                 do kb = 0 to fnam;
497   6                 fcbv(kb) = bfcb(kb);
498   6                 end;
499   5             fcbe(i),fcbb(i),fcbk(i),fcbr(i) = 0;
```



```
500   5             end;
              /* entry is at, or was placed at location i in fcbs */
501   4         fcbe(i) = fcbe(i) + 1; /* extent incremented */
              /* record count */
502   4         fcbr(i) = fcbr(i) + bfcb(frc)
                      + (bfcb(fext) and extmsk) * 128;
              /* count kilobytes */
503   4     countbytes: /* label for debug */
              lb = 1;
504   4         if maxall > 255 then lb = 2; /* double precision inx */
506   4             do kb = fdm to fdl by lb;
507   5                 mb = bfcb(kb);
508   5                 if lb = 2 then /* double precision inx */
509   5                     mb = mb or bfcb(kb+1);
510   5                 if mb <> 0 then /* allocated */
511   5                 call add$block(.fcbk(i),.fcbb(i));
512   5             end;
513   4         call searchn; /* to next entry in directory */
514   4         end; /* of do while dcnt <> 255 */

515   3     display: /* label for debug */
              /* now display the collected data */
              if fcbn = 0 then call print(.('File Not Found',0)); else
517   3     if scase = 255 then /* display collected data */
518   3         do;
              /* sort the file names in ascending order */
519   4         if fcbn > 1 then /* requires at least two to sort */
520   4             do; l = 1;
522   5                 do while l > 0; /* bubble sort */
523   6                 l = 0;
524   6                     do m = 0 to fcbn - 2;
525   7                     i = finx(m+1); call multi16; bfcba = fcbsa; i
529   7                     call multi16; /* sets fcbsa, basing fcbv */
530   7                         do kb = 1 to fnam; /* compare for less or
531   8                         if (b:=bfcb(kb)) < (f:=fcbv(kb)) then /* s
532   8                             do; k = finx(m); finx(m) = finx(m + 1)
535   9                             finx(m + 1) = k; l = l + 1; kb = fnam;
538   9                             end;
539   8                         else if b > f then kb = fnam; /* stop comp
                                  end;
542   7                     end;
543   6                 end;
544   5             end;
545   4         if sizeset then
546   4             call print(.(' Size ',0)); else
547   4             call crlf;
548   4         call printx(.(' Recs  Bytes  Ext Acc',0));
549   4         i = 0;
550   4             do while i < fcbn;
551   5             i = finx(i); /* i is the index to next in order */
552   5             call multi16; call crlf;
                  /* print the file length */
554   5             call move(16,.fcbv(0),fcba);
555   5             fcb(0) = 0;
556   5             if sizeset then
557   5                 do; call getfilesize(fcba);
559   6                 if rovf <> 0 then call printx(.('65536',0)); else
```

```
561   6                       call pdecimal(rrec,10000);
562   6                   call printb;
563   6                   end;
564   5               call pdecimal(fcbr(1),10000); /* rrrrr */
565   5               call printb; /* blank */
566   5               call pdecimal(fcbk(1),10000);  /* bbbbbk */
567   5               call printchar('k'); call printb;
569   5               call pdecimal(fcbe(1),1000);   /* eeee */
570   5               call printb;
571   5               call printchar('R');
572   5               call printchar('/');
573   5               if rol(fcbv(rofile),1) then
574   5                   call printchar('O'); else
575   5                   call printchar('W');
576   5               call printb;
577   5               call printchar('A'+cselect); call printchar(':');
                      /* print filename.typ */
579   5               if (mb:=rol(fcbv(infile),1)) then call printchar('(');
581   5               call printfn;
582   5               if mb then call printchar(')');
584   5               l = l + 1;
585   5               end;
586   4           call nralloc;
587   4           end; else
588   3       setfileatt: /* label for debug */
                  /* set file attributes */
                      do;
589   4                 l = 0;
590   4                 do while l < fcbn;
591   5                 if break then
592   5                     do; call abortmsg; return;
595   6                     end;
596   5                 l = l;
597   5                 call mult116;
598   5                 call crlf;
599   5                 call printfn;
600   5                     do case scase;
                          /* set to r/o */
601   6                     fcbv(rofile) = fcbv(rofile) or 80h;
                          /* set to r/w */
602   6                     fcbv(rofile) = fcbv(rofile) and 7fh;
                          /* set to sys */
603   6                     fcbv(infile) = fcbv(infile) or 80h;
                          /* set to dir */
604   6                     fcbv(infile) = fcbv(infile) and 7fh;
605   6                     end;
                      /* place name into default fcb location */
606   5               call move(16,fcbsa,fcba);
607   5               fcb(0) = 0; /* in case matched user# > 0 */
608   5               call setind; /* indicators set */
609   5               call printx(.(' set to ',0));
610   5               call printx(.fstatlist(shl(scase,2)));
611   5               l = l + 1;
612   5               end;
613   4           end;
614   3       end getfile;
```



```
615   2       setdrivestatus: procedure;
                  /* handle possible drive status assignment */
616   3           call scan; /* remove drive name */
617   3           call scan; /* check for = */
618   3           if accum(0) = '=' then
619   3               do; call scan; /* get assignment */
621   4               if compare(.('R/O '),
622   4                   do; call setdisk; /* a: ... */
624   5                   call writeprot;
625   5                   end; else
626   4               call print(.('Invalid Disk Assignment',0));
627   4               end;
                  else /* not a disk assignment */
628   3               do; call setdisk;
630   4               if match(.dev1,8) = 8 then call drive$status; else
632   4                   call getfile;
633   4               end;
634   3           end setdrivestatus;

                 /* save stack pointer and reset */
635   2       oldsp = stackptr;
636   2       stackptr = .stack(length(stack));
                 /* process request */
637   2       if version < cpmversion then
638   2           call print(.('Wrong CP/M Version (Requires 2.0)',0));
                  else
639   2               do;
                      /* size display if $S set in command */
640   3               ibp = 1; /* initialize buffer pointer */
641   3               if fcb(0) = 0 and fcb(1) = ' ' then /* stat only */
642   3                   call prstatus; else
643   3                   do;
644   4                   if fcb(0) <> 0 then
645   4                       call setdrivestatus; else
646   4                       do;
647   5                       if not devreq then /* must be file name */
648   5                           call getfile;
649   5                       end;
650   4                   end;
651   3               end;
                 /* restore old stack before exit */
652   2           stackptr = oldsp;
653   2           end status;
654   1       end;
```

```
MODULE INFORMATION:

     CODE AREA SIZE     = 1349H    4937D
     VARIABLE AREA SIZE = 14B2H    5298D
     MAXIMUM STACK SIZE = 0010H      16D
     893 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
0000 STAT#
0000 STAT#
0433 16     0490 20     0494 22     049F 23     04A0 24
04A0 25     04A5 26     04AA 27     04AB 28     04AB 29
04B0 30     04B1 31     04B7 34     04C0 35     04C7 36
04CE 37     04D1 38     04D2 39     04D8 41     04DB 42
04E3 43     04E4 44     04E4 45     04ED 46     04ED 48
04ED 49     04F6 50     04F6 51     04FA 53     0505 54
0506 55     050C 57     0518 58     0519 59     051F 61
052B 62     052C 63     052C 64     0537 65     0538 66
0538 67     0541 68     0541 69     0547 71     0550 72
0551 73     0551 74     055A 75     055A 76     055A 77
0563 78     0563 79     0563 80     056B 81     056C 82
056C 83     0575 84     0575 85     0575 86     057D 87
057E 88     057E 89     0589 90     058A 91     058A 92
0593 93     0593 94     0597 96     05A2 97     05A3 98
05A9 100    05B2 101    05B3 105    05B3 106    05B6 107
05CB 108    05CC 109    05D0 111    05D7 112    05DA 113
05DB 114    05E1 116    05FF 117    05FE 119    0604 123
0612 124    062C 125    062F 126    0636 127    0639 128
0639 129    06FB 131    06FF 133    06FC 134    0700 135
0639 136    0648 137    064C 138    064F 139    0654 140
065C 141    0670 142    067A 143    067F 144    06D1 145
06DF 146    06E3 147    06E6 148    06EA 149    0701 150
070B 152    0710 153    071C 154    072A 155    0734 156
0740 157    075B 158    0761 160    0766 161    076F 162
076F 163    0772 164    0773 165    077D 169    078C 170
0799 171    07A8 172    07B3 173    07B6 174    07B7 175
07BB 178    07C4 179    07C8 180    07DF 181    07E6 182
07F1 183    07F8 184    0801 185    080E 186    0812 187
0812 188    0812 189    081B 190    0819 191    0819 195
081F 196    082B 197    0831 198    083F 199    084A 200
0851 201    0857 202    085D 203    0865 204    087F 205
088D 206    0890 207    0893 208    08A1 209    08AF 210
08BB 211    08C2 212    08C3 213    09C0 215    09C6 217
09C9 218    09D4 219    09D9 220    09DC 221    08C3 222
08C9 223    08D2 224    08D7 225    08DD 226    08EC 227
091C 228    0925 229    092D 230    0933 231    093D 232
0943 233    0951 234    0957 235    0969 236    096F 237
0986 238    098C 239    0994 240    099A 241    09A3 242
09A9 243    09B6 244    09BC 245    09BF 246    09DD 247
09DD 249    09E3 250    09E8 251    09F4 252    09FC 254
0A03 255    0A06 256    0A06 257    0A13 258    0A15 259
0A18 260    0A19 261    0A21 264    0A2B 265    0A37 266
0A3C 267    0A4A 268    0A64 269    0A69 270    0A6D 271
0A72 272    0A79 273    0A7D 274    0AB4 275    0AB7 276
0AB7 278    0C69 281    0C6F 283    0C7B 284    0C7F 285
0C86 286    0C89 287    0C8E 288    0A87 289    0A8C 290
0A8C 291    0A8F 292    0A9F 293    0AA8 294    0AAC 295
0AB4 297    0ABA 298    0ABF 299    0ACB 300    0ADC 301
0AE2 302    0AF9 303    0B01 304    0B0B 305    0B0E 306
0B15 307    0B18 308    0B20 310    0B26 311    0B2C 312
0B32 313    0B38 314    0B3E 315    0B4C 316    0B4F 317
0B60 318    0B66 319    0B85 320    0B8A 321    0BA1 322
0BA4 323    0BAB 324    0BAE 325    0BB6 327    0BB9 328
0BBC 329    0BBF 330    0BC7 331    0BCD 333    0BDB 334
0BDE 335    0BE6 337    0BEC 338    0BEF 339    0BEF 340
0BF2 341    0C0B 343    0C11 344    0C14 345    0C14 346
0C19 347    0C25 348    0C2D 349    0C35 350    0C38 351
0C46 352    0C46 353    0C49 354    0C51 355    0C54 356
```

```
0C5C 358    0C62 359    0C65 360    0C65 361    0C68 362
0C8F 363    0C95 365    0C9B 366    0CA0 367    0CAC 368
0CBA 369    0CC4 370    0CD0 371    0CEB 373    0CF0 374
0CF9 375    0CF9 376    0CFC 377    0D01 378    0D04 379
0D05 380    0D05 381    0D0B 382    0D14 383    0D1A 384
0D1B 385    0D1B 386    0D25 387    0D26 388    0D26 389
0D2C 390    0D2F 391    0D32 392    0D33 393    0D33 396
0D39 397    0D3F 398    0D44 399    0D50 400    0D58 402
0D5F 403    0D62 404    0D68 405    0D70 406    0D78 407
0D7D 408    0DB3 409    0DA6 410    0D86 411    0D93 412
0DA0 413    0DA2 414    0DA5 415    0DA8 416    0DA9 417
0DA9 418    0DB1 419    0DB9 420    0DBA 421    135D 425
135D 426    136B 427    136C 430    136C 432    1374 433
1377 434    13B7 435    139F 436    13A7 437    13B7 438
13BD 439    13C0 440    13C0 441    13C0 443    13CE 444
13E3 446    13EB 447    13F0 448    13F7 449    13F7 450
1401 451    0DBA 452    0DBD 453    0DC0 454    0DC5 455
0DCA 456    0DD1 458    0DD9 459    0DDA 460    0DE1 461
0DE4 462    0DEC 464    0DEF 465    0DF0 466    0DF0 467
0DFA 468    0E04 469    0E0A 470    0E12 471    0E24 472
0E29 473    0E2F 474    0E45 475    0E48 476    0E56 477
0E71 478    0E79 479    0E84 480    0E8E 481    0E95 482
0E98 483    0E9F 484    0EA9 486    0EB3 487    0EB6 488
0ED8 490    0EDE 491    0EE4 492    0EEA 493    0EED 494
0EED 495    0EFE 496    0F0C 497    0F24 498    0F2E 499
0F60 500    0F60 501    0F79 502    0FB9 503    0FBE 504
0FCD 505    0FD2 506    0FF1 507    0FFF 508    1007 509
101C 510    1024 511    103A 512    103D 513    1040 514
1043 515    104F 516    1058 517    1060 519    106B 521
1071 522    107C 523    1082 524    1097 525    10A6 526
10A9 527    10AF 528    10BE 529    10C1 530    10CF 531
10F3 533    1102 534    111B 535    112C 536    1133 537
1138 538    113B 539    1145 540    114A 541    1154 542
1161 543    1164 544    1164 545    116B 546    1174 547
1177 548    117D 549    1183 550    118F 551    119E 552
11A1 553    11A4 554    11B8 555    11BD 556    11C4 558
11CA 559    11D2 560    11DB 561    11E6 562    11E9 563
11E9 564    11FA 565    11FD 566    120E 567    1213 568
1216 569    1227 570    122A 571    122F 572    1234 573
1241 574    1249 575    124E 576    1251 577    125A 578
125F 579    126F 580    1274 581    1277 582    127E 583
1283 584    128A 585    128D 586    1290 587    1293 588
1293 589    1299 590    12A5 591    12AC 593    12AF 594
12B0 595    12B0 596    12B6 597    12B9 598    12BC 599
12BF 600    12CF 601    12E1 602    12F3 603    1305 604
1317 605    131F 606    1333 607    1338 608    133B 609
1341 610    1352 611    1359 612    135C 613    135C 614
1402 615    1402 616    1405 617    1408 618    1410 620
1413 621    141D 623    1420 624    1423 625    1426 626
142C 627    142F 629    1432 630    143F 631    1445 632
1448 633    1448 634    0433 635    043A 636    043E 637
0446 638    044F 640    0454 641    046C 642    0472 644
047A 645    0480 647    0488 648    048B 649    048B 650
048B 651    048B 652    048F 653
0000 MODULE#
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE ED
OBJECT MODULE PLACED IN ED.OBJ
COMPILER INVOKED BY:  PLM80 ED.PLM DEBUG


  1          ED:
             DO;
                 /* MODIFIED FOR .PRL OPERATION MAY, 1979 */
                 /* MODIFIED FOR OPERATION WITH CP/M 2.0 AUGUST 1979 */
  2    1     DECLARE
                 /* JMP EDCOMMAND - 3 (TO ADDRESS LXI SP) */
                 EDJMP BYTE DATA(0C3H),
                 EDADR ADDRESS DATA(.EDCOMMAND-3);

  3    1     DECLARE
                 BDISK     BYTE    EXTERNAL, /* BOOT DISK 0004H */
                 MAXB      ADDRESS EXTERNAL, /* MAX BASE 0006H */
                 FCB (33)  BYTE    EXTERNAL, /* FCB 005CH */
                 BUFF (128)BYTE    EXTERNAL, /* BUFFER 0080H */
                 SECTSHF   LITERALLY '7',    /* SHL(1,SECTSHF) = SECTSIZE */
                 SECTSIZE  LITERALLY '80H';  /* SECTOR SIZE */

  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     BOOT: PROCEDURE EXTERNAL;
                 /* SYSTEM REBOOT */
 11    2         END BOOT;

               /*  E D :   T H E   C P / M   C O N T E X T   E D I T O R  */

                 /*      COPYRIGHT (C) 1976, 1977, 1978, 1979
                         DIGITAL RESEARCH
                         BOX 579 PACIFIC GROVE
                         CALIFORNIA 93950
                 */
 12    1     DECLARE COPYRIGHT(*) BYTE DATA
                 (' COPYRIGHT (C) 1979, DIGITAL RESEARCH ');

               /* COMMAND          FUNCTION
                  -------          --------
                   A          APPEND LINES OF TEXT TO BUFFER
                   B          MOVE TO BEGINNING OR END OF TEXT
                   C          SKIP CHARACTERS
                   D          DELETE CHARACTERS
                   E          END OF EDIT
                   F          FIND STRING IN CURRENT BUFFER
                   H          MOVE TO TOP OF FILE (HEAD)
                   I          INSERT   CHARACTERS FROM KEYBOARD
                              UP TO NEXT <ENDFILE>
                   J          JUXTAPOSITION OPERATION - SEARCH FOR FIRST STRIN
```



```
                   INSERT SECOND STRING, DELETE UNTIL THIRD STRING
   K               DELETE LINES
   L               SKIP LINES
   M               MACRO DEFINITION (SEE COMMENT BELOW)
   N               FIND NEXT OCCURRENCE OF STRING
                   WITH AUTO SCAN THROUGH FILE
   O               RE-EDIT OLD FILE
   P               PAGE AND DISPLAY (MOVES UP OR DOWN 24 LINES AND
                   DISPLAYS 24 LINES)
   Q               QUIT EDIT WITHOUT UPDATING THE FILE
   R<FILENAME>     READ FROM FILE <FILENAME>.LIB UNTIL <ENDFILE> AN
                   INSERT INTO TEXT
   S               SEARCH FOR FIRST STRING, REPLACE BY SECOND STRIN
   T               TYPE LINES
   U               TRANSLATE TO UPPER CASE (-U CHANGES TO NO TRANSL
   W               WRITE LINES OF TEXT TO FILE
   X               TRANSFER (XFER) LINES TO TEMP FILE
   Z               SLEEP FOR 1/2 SECOND (USED IN MACROS TO STOP DIS
   <CR>            MOVE UP OR DOWN AND PRINT ONE LINE
```

IN GENERAL, THE EDITOR ACCEPTS SINGLE LETTER COMMANDS WITH OPT INTEGER VALUES PRECEDING THE COMMAND. THE EDITOR ACCEPTS BOTH UPP CASE COMMANDS AND VALUES, AND PERFORMS TRANSLATION TO UPPER CASE U LOWING CONDITIONS. IF THE COMMAND IS TYPED IN UPPER CASE, THEN TH FOLLOWS IS TRANSLATED TO UPPER CASE. THUS, IF THE "I" COMMAND IS UPPER CASE, THEN ALL INPUT IS AUTOMATICALLY TRANSLATED (ALTHOUGH F LOWER CASE, AS TYPED). IF THE "A" COMMAND IS TYPED IN UPPER CASE, INPUT IS TRANSLATED AS READ FROM THE DISK. GLOBAL TRANSLATION TO CAN BE CONTROLLED BY THE "U" COMMAND (-U TO NEGATE ITS EFFECT). I OPERATING WITH AN UPPER CASE ONLY TERMINAL, THEN OPERATION IS AUTO SIMILARLY, IF YOU ARE OPERATING WITH A LOWER CASE TERMINAL, AND TR TO UPPER CASE IS NOT SPECIFIED, THEN LOWER CASE CHARACTERS CAN BE

A NUMBER OF COMMANDS CAN BE PRECEDED BY A POSITIVE OR NEGATIVE INTEGER BETWEEN 0 AND 65535 (1 IS DEFAULT IF NO VALU IS SPECIFIED). THIS VALUE DETERMINES THE NUMBER OF TIMES THE COMMAND IS APPLIED BEFORE RETURNING FOR ANOTHER COMMAND.

THE COMMANDS

C D K L T P U <CR>

CAN BE PRECEDED BY AN UNSIGNED, POSITIVE, OR NEGATIVE NUMBER, THE COMMANDS

A F J N W Z

CAN BE PRECEDED BY AN UNSIGNED OR POSITIVE NUMBER, THE COMMANDS

E H O Q

CANNOT BE PRECEDED BY A NUMBER. THE COMMANDS

F I J M R S

ARE ALL FOLLOWED BY ONE OR MORE STRINGS OF CHARACTERS WHICH C BE OPTIONALLY SEPARATED OR TERMINATED BY EITHER <ENDFILE> OR THE <ENDFILE> IS GENERALLY USED TO SEPARATE THE SEARCH STRING IN THE S AND J COMMANDS, AND IS USED AT THE END OF THE COMMAN ADDITIONAL COMMANDS FOLLOW. FOR EXAMPLE, THE FOLLOWING COMMA SEQUENCE SEARCHES FOR THE STRING 'GAMMA', SUBSTITUTES THE ST 'DELTA', AND THEN TYPES THE FIRST PART OF THE LINE WHERE THE CHANGE OCCURRED, FOLLOWED BY THE REMAINDER OF THE LINE WHICH CHANGED:

```
                SGAMMA<ENDFILE DELTA<ENDFILE ØT1<CR>

THE CONTROL-L CHARACTER IN SEARCH AND SUBSTITUTE STRINGS IS
REPLACED ON INPUT BY <CR><LF> CHARACTERS.  THE CONTROL-I KEY
IS TAKEN AS A TAB CHARACTER.

   THE COMMAND R MUST BE FOLLOWED BY A FILE NAME (WITH ASSUME
TYPE OF 'LIB') WITH A TRAILING <CR> OR <ENDFILE>.  THE COMMAN
I IS FOLLOWED BY A STRING OF SYMBOLS TO INSERT, TERMINATED BY
A <CR> OR <ENDFILE>.  IF SEVERAL LINES OF TEXT ARE TO BE INSE
THE I CAN BE DIRECTLY FOLLOWED BY AN <ENDFILE> OR <CR> IN WHI
CASE THE EDITOR ACCEPTS LINES OF INPUT TO THE NEXT <ENDFILE>.
THE COMMAND ØT PRINTS THE FIRST PART OF THE CURRENT LINE,
AND THE COMMAND ØL MOVES THE REFFRENCE TO THE BEGINNING OF TH
CURRENT LINE.  THE COMMAND ØP PRINTS THE CURRENT PAGE ONLY, W
THE COMMAND ØZ READS THE CONSOLE RATHER THAN WAITING (THIS IS
AGAIN WITHIN MACROS TO STOP THE DISPLAY - THE MACRO EXPANSION
STOPS UNTIL A CHARACTER IS READ.  IF THE CHARACTER IS NOT A B
THEN THE MACRO EXPANSION CONTINUES NORMALLY).

   NOTE THAT A POUND SIGN IS TAKEN AS THE NUMBER 65535, ALL
UNSIGNED NUMBERS ARE ASSUMED POSITIVE, AND A SINGLE - IS ASSU

A NUMBER OF COMMANDS CAN BE GROUPED TOGETHER AND EXECUTED
REPETITIVELY USING THE MACRO COMMAND WHICH TAKES THE FORM

        <NUMBER>MC1C2...CN<DELIMITER>

WHERE <NUMBER> IS A NON-NEGATIVE INTEGER N, AND <DELIMITER> I
<ENDFILE> OR <CR>.  THE COMMANDS C1 ... CN  FOLLOWING THE M A
EXECUTED N TIMES, STARTING AT THE CURRENT POSITION IN THE BUF
IF N IS Ø, 1, OR OMITTED, THE COMMANDS ARE EXECUTED UNTIL THE
IF THE BUFFER IS ENCOUNTERED.

THE FOLLOWING MACRO, FOR EXAMPLE, CHANGES ALL OCCURRENCES OF
THE NAME 'GAMMA' TO 'DELTA', AND PRINTS THE LINES WHICH
WERE CHANGED:

                MFGAMMA<ENDFILE>-5DIDELTA<ENDFILE>ØLT<CR>

(NOTE: AN <ENDFILE> IS THE CP/M END OF FILE MARK - CONTROL-Z)

IF ANY KEY IS DEPRESSED DURING TYPING OR MACRO EXPANSION, THE
FUNCTION IS CONSIDERED TERMINATED, AND CONTROL RETURNS TO THE
OPERATOR.

ERROR CONDITIONS ARE INDICATED BY PRINTING ONE OF THE CHARACTE

 SYMBOL                  ERROR CONDITION
 ------  ---------------------------------------------------
 GREATER FREE MEMORY IS EXHAUSTED - ANY COMMAND CAN BE ISS
         WHICH DOES NOT INCREASE MEMORY REQUIREMENTS.
 QUESTION UNRECOGNIZED COMMAND OR ILLEGAL NUMERIC FIELD
 POUND   CANNOT APPLY THE COMMAND THE NUMBER OF TIMES SPEC
         (OCCURS IF SEARCH STRING CANNOT BE FOUND)
 LETTER O CANNOT OPEN <FILENAME>.LIB IN R COMMAND
```



```
                THE ERROR CHARACTER IS ALSO ACCOMPANIED BY THE LAST CHARACTER
                SCANNED WHEN THE ERROR OCCURRED.                        */

  13   1      DECLARE LIT LITERALLY 'LITERALLY',
                  DCL LIT 'DECLARE',
                  PROC LIT 'PROCEDURE',
                  ADDR LIT 'ADDRESS',
                  CTLL LIT '0CH',
                  CTLR LIT '12H',  /* REPEAT LINE IN INSERT MODE */
                  CTLU LIT '15H',  /* LINE DELETE IN INSERT MODE */
                  CTLX LIT '18H',  /* EQUIVALENT TO CTLU */
                  CTLH LIT '08H',  /* BACKSPACE */
                  TAB LIT '09H',        /* TAB CHARACTER */
                  LCA  LIT '110$0001B',  /* LOWER CASE A */
                  LCZ  LIT '111$1010B',  /* LOWER CASE Z */
                  ENDFILE LIT '1AH';    /* CP/M END OF FILE */

  14   1      DECLARE
                  MAX ADDRESS,                  /* .MEMORY(MAX)=0 (END) */
                  MAXM ADDRESS,                 /* MINUS 1 */
                  HMAX ADDRESS;                 /* = MAX/2 */

  15   1      DECLARE
                  RO LITERALLY '9',  /* R/O FILE INDICATOR */
                  SY LITERALLY '10', /* SYSTEM FILE ATTRIBUTE */
                  EX LITERALLY '12', /* EXTENT NUMBER POSITION */
                  UB LITERALLY '13', /* UNFILLED BYTES */
                  MD LITERALLY '14', /* MODULE NUMBER POSITION */
                  NR LITERALLY '32', /* NEXT RECORD FIELD */
                  FS LITERALLY '33', /* FCB SIZE */
                  RFCB (FS) BYTE /* READER FILE CONTROL BLOCK */
                      INITIAL(0, /* FILE NAME */ '          ',
                                 /* FILE TYPE */ 'LIB',0,0,0),
                  RBP BYTE,      /* READ BUFFER POINTER */
                  XFCB (FS) BYTE /* XFER FILE CONTROL BLOCK */
                      INITIAL(0, 'X$$$$$$$$','LIB',0,0,0),
                  XFCBE BYTE AT(.XFCB(EX)),  /* XFCB EXTENT */
                  XFCBM BYTE AT(.XFCB(MD)),  /* MODULE NUMBER */
                  XFCBR BYTE AT(.XFCB(NR)),  /* XFCB RECORD # */
                  XBUFF (SECTSIZE) BYTE, /* XFER BUFFER */
                  XBP BYTE,              /* XFER POINTER */
                  XFERON BYTE,           /* TRUE IF XFER ACTIVE */

                  NBUF BYTE,                    /* NUMBER OF BUFFERS */
                  BUFFLENGTH ADDRESS,           /* NBUF * SECTSIZE */
                  SFCB (FS) BYTE AT(.FCB),      /* SOURCE FCB = DEFAULT FCB */
                  SBUFFADR ADDRESS,             /* SOURCE BUFFER ADDRESS */
                  SBUFF BASED SBUFFADR (128) BYTE, /* SOURCE BUFFER */

                  DFCB (FS) BYTE,               /* DEST FILE CONTROL BLOCK */
                  DFUB BYTE AT(.DFCB(UB)),  /* UNFILLED BYTES IN LAST RECORD */
                  DBUFFADR ADDRESS,         /* DESTINATION BUFFER ADDRESS */
                  DBUFF BASED DBUFFADR (128) BYTE, /* DEST BUFFER */
                  NSOURCE ADDRESS,          /* NEXT SOURCE CHARACTER */
                  NDEST   ADDRESS;          /* NEXT DESTINATION CHAR */

  16   1      DECLARE SDISK BYTE,     /* SOURCE FILE DISK */
P
```

```
                DDISK BYTE;             /* DESTINATION FILE DISK */

             /* IO SECTION */

 17   1     READCHAR: PROCEDURE BYTE; RETURN MON2(1,0);
 19   2         END READCHAR;

 20   1     DECLARE TRUE LITERALLY '1', FALSE LITERALLY '0',
             FOREVER LITERALLY 'WHILE TRUE',
             CR LITERALLY '13',
             LF LITERALLY '10',
             WHAT LITERALLY '63';

 21   1     DECLARE
             PRINTSUPPRESS BYTE; /* TRUE IF PRINT SUPPRESSED */

 22   1     PRINTCHAR: PROCEDURE(CHAR);
 23   2         DECLARE CHAR BYTE;
 24   2         IF PRINTSUPPRESS THEN RETURN;
 26   2         CALL MON1(2,CHAR);
 27   2         END PRINTCHAR;

 28   1     DECLARE
             COLUMN  BYTE, /* CONSOLE COLUMN POSITION */
             SCOLUMN BYTE, /* STARTING COLUMN IN "I" MODE */
             TCOLUMN BYTE, /* TEMP DURING BACKSPACE */
             QCOLUMN BYTE; /* TEMP DURING BACKSPACE */

 29   1     TTYCHAR: PROCEDURE(CHAR);
 30   2         DECLARE CHAR BYTE;
 31   2         IF CHAR >= ' ' THEN COLUMN = COLUMN + 1;
 33   2         IF CHAR = LF THEN COLUMN = 0;
 35   2         CALL PRINTCHAR(CHAR);
 36   2         END TTYCHAR;

 37   1     BACKSPACE: PROCEDURE;
             /* MOVE BACK ONE POSITION */
 38   2         IF COLUMN = 0 THEN RETURN;
 40   2         CALL TTYCHAR(CTLH); /* COLUMN = COLUMN - 1 */
 41   2         CALL TTYCHAR(' '); /* COLUMN = COLUMN + 1 */
 42   2         CALL TTYCHAR(CTLH); /* COLUMN = COLUMN - 1 */
 43   2         COLUMN = COLUMN - 2;
 44   2         END BACKSPACE;

 45   1     PRINTABS: PROCEDURE(CHAR);
 46   2         DECLARE (CHAR,I,J) BYTE;
 47   2         I = CHAR = TAB AND 7 - (COLUMN AND 7);
 48   2         IF CHAR = TAB THEN CHAR = ' ';
 50   2             DO J = 0 TO I;
 51   3             CALL TTYCHAR(CHAR);
 52   3             END;
 53   2         END PRINTABS;

 54   1     GRAPHIC: PROCEDURE(C) BYTE;
 55   2         DECLARE C BYTE;
             /* RETURN TRUE IF GRAPHIC CHARACTER */
 56   2         IF C >= ' ' THEN RETURN TRUE;
```



```
 58   2         RETURN C = CR OR C = LF OR C = TAB;
 59   2         END GRAPHIC;

 60   1     PRINTC: PROCEDURE(C);
 61   2         DECLARE C BYTE;
 62   2         IF NOT GRAPHIC(C) THEN
 63   2             DO; CALL PRINTABS('^');
 65   3             C = C + '@';
 66   3             END;
 67   2         CALL PRINTABS(C);
 68   2         END PRINTC;

 69   1     CRLF: PROCEDURE;
 70   2         CALL PRINTC(CR); CALL PRINTC(LF);
 72   2         END CRLF;

 73   1     PRINTM: PROCEDURE(A);
 74   2         DECLARE A ADDRESS;
 75   2         CALL MON1(9,A);
 76   2         END PRINTM;

 77   1     PRINT: PROCEDURE(A);
 78   2         DECLARE A ADDRESS;
 79   2         CALL CRLF;
 80   2         CALL PRINTM(A);
 81   2         END PRINT;

 82   1     READ: PROCEDURE(A);
 83   2         DECLARE A ADDRESS;
 84   2         CALL MON1(10,A);
 85   2         END READ;

 86   1     DECLARE DCNT BYTE;

 87   1     OPEN: PROCEDURE(FCB);
 88   2          DECLARE FCB ADDRESS;
 89   2          DCNT = MON2(15,FCB);
 90   2          END OPEN;

 91   1     CLOSE: PROCEDURE(FCB);
 92   2         DECLARE FCB ADDRESS;
 93   2         DCNT = MON2(16,FCB);
 94   2         END CLOSE;

 95   1     SEARCH: PROCEDURE(FCB);
 96   2         DECLARE FCB ADDRESS;
 97   2         DCNT = MON2(17,FCB);
 98   2         END SEARCH;

 99   1     DELETE: PROCEDURE(FCB);
100   2         DECLARE FCB ADDRESS;
101   2         CALL MON1(19,FCB);
102   2         END DELETE;

103   1     DISKREAD: PROCEDURE(FCB) BYTE;
104   2         DECLARE FCB ADDRESS;
```

```
105   2          RETURN MON2(20,.FCB);
106   2          END DISKREAD;

107   1      DISKWRITE: PROCEDURE(FCB) BYTE;
108   2          DECLARE FCB ADDRESS;
109   2          RETURN MON2(21,.FCB);
110   2          END DISKWRITE;

111   1      MAKE: PROCEDURE(FCB);
112   2          DECLARE FCB ADDRESS;
113   2          DCNT = MON2(22,.FCB);
114   2          END MAKE;

115   1      RENAME: PROCEDURE(FCB);
116   2          DECLARE FCB ADDRESS;
117   2          CALL MON1(23,.FCB);
118   2          END RENAME;

119   1      DECLARE (MAXLEN,COMLEN) BYTE, COMBUFF(128) BYTE,
               (TCBP,CBP) BYTE;

120   1      READCOM: PROCEDURE;
121   2          MAXLEN = 128; CALL READ(.MAXLEN);
123   2          END READCOM;

124   1      BREAK$KEY: PROCEDURE BYTE;
125   2          IF MON2(11,0) THEN
126   2              DO; /* CLEAR CHAR */
127   3              CALL MON1(1,0); RETURN TRUE;
129   3              END;
130   2          RETURN FALSE;
131   2          END BREAK$KEY;

132   1      CSELECT: PROCEDURE BYTE;
               /* RETURN CURRENT DRIVE NUMBER */
133   2          RETURN MON2(25,0);
134   2          END CSELECT;

135   1      SELECT: PROCEDURE(DISK);
136   2          DECLARE DISK BYTE;
               /* SET DRIVE NUMBER */
137   2          CALL MON1(14,DISK);
138   2          END SELECT;

139   1      SETDMA: PROCEDURE(A);
140   2          DECLARE A ADDRESS;
               /* SET DMA ADDRESS */
141   2          CALL MON1(26,A);
142   2          END SETDMA;

143   1      REBOOT: PROCEDURE;
144   2          IF XFERON THEN CALL DELETE(.XFCB);
146   2          CALL BOOT;
147   2          END REBOOT;

148   1      DECLARE /* LINE COUNTERS *
               BASELINE ADDRESS,       /* CURRENT LINE */
```



```
               RELLINE  ADDRESS,       /* RELATIVE LINE IN TYPEOUT */
               LINESET BYTE;           /* TRUE IF LINE #'S PRINTED */

             /*  INPUT / OUTPUT BUFFERING ROUTINES */

           /* THE PL/M BUILT-IN PROCEDURE "MOVE" IS USED TO MOVE STORAGE,
              ITS DEFINITION IS:
               MOVE: PROCEDURE(COUNT,SOURCE,DEST);
                   DECLARE (COUNT,SOURCE,DEST) ADDRESS;
                   / MOVE DATA FROM SOURCE TO DEST ADDRESSES, FOR COUNT BYTES
                   END MOVE;
            */

149   1      ABORT: PROCEDURE(A);
150   2          DECLARE A ADDRESS;
151   2          CALL PRINT(A);
152   2          CALL CRLF;
153   2          CALL REBOOT;
154   2          END ABORT;

155   1      FERR: PROCEDURE;
156   2          CALL CLOSE(.DFCB); /* ATTEMPT TO CLOSE FILE FOR LATER RECOVERY
157   2          CALL ABORT (.('DISK OR DIRECTORY FULL$'));
158   2          END FERR;

159   1      SETTYPE: PROCEDURE(A);
160   2          DECLARE A ADDRESS;
161   2          CALL MOVE(3,A,.DFCB+9);
162   2          END SETTYPE;

163   1      SETUP: PROCEDURE;
164   2          NSOURCE = BUFFLENGTH; NDEST = 0;
166   2          SFCB(EX), SFCB(MD), SFCB(NR) = 0;
               /* REEL AND RECORD ZEROED */
               /* COPY NAME TO DESTINATION FCB */
167   2          CALL MOVE(33,.FCB,.DFCB);
               /* SOURCE AND DESTINATION DISKS SET */

               /* IF SOURCE AND DESTINATION DISKS DIFFER, CHECK FOR
               AN EXISTING SOURCE FILE ON THE DESTINATION DISK - THERE
               COULD BE A FATAL ERROR CONDITION WHICH COULD DESTROY A
               FILE IF THE USER HAPPENED TO BE ADDRESSING THE WRONG
               DISK */
168   2          IF SDISK <> DDISK THEN
169   2              DO; CALL SELECT(DDISK);
171   3              CALL SEARCH(.FCB);
172   3              IF DCNT <> 255 THEN /* SOURCE FILE PRESENT ON DEST DISK */
173   3                  CALL ABORT(.('FILE EXISTS, ERASE IT$'));
174   3              END;
175   2          CALL SELECT(SDISK);
176   2          CALL OPEN(.FCB);
177   2          IF DCNT = 255 THEN
178   2              DO; CALL MAKE(.FCB);
180   3              IF DCNT = 255 THEN CALL FERR;
182   3              CALL PRINT(.('NEW FILE$'));
183   3              CALL CRLF;
184   3              END; ELSE
```

```
185   2           IF ROL(FCB(RO),1) THEN
186   2               DO;
187   3               CALL PRINT(.('** FILE IS READ/ONLY **$'));
188   3               CALL CRLF;
189   3               END; ELSE
190   2           IF ROL(FCB(SY),1) THEN
191   2               CALL ABORT(.('"SYSTEM" FILE NOT ACCESSIBLE$'));
            CALL SETTYPE(.('BAK'));
193   2           CALL DELETE(.DFCB);
194   2           IF SDISK <> DDISK THEN
195   2               DO; /* REMOVE BAK FILES FROM DESTINATION DISK */
196   3               CALL SELECT(DDISK);
197   3               CALL DELETE(.DFCB);
198   3               END;
199   2           CALL SETTYPE(.('$$$'));
200   2           CALL DELETE(.DFCB);
201   2           CALL MAKE(.DFCB);
202   2           DFCB(32) = 0;  /* NEXT RECORD IS ZERO */
203   2           IF DCNT = 255 THEN CALL FERR;
                /* THE TEMP FILE IS NOW CREATED */
205   2           BASELINE = 1; /* START WITH LINE 1 */
206   2           END SETUP;

207   1       XCLEAR: PROCEDURE;
                /* CLEAR THE XFER FILE PARAMETERS */
208   2           XFERON, XFCBE, XFCBR, XBP = 0;
209   2           END XCLEAR;

210   1       SETXDMA: PROCEDURE;
211   2           CALL SELECT(SDISK);
212   2           CALL SETDMA(.XBUFF);
213   2           END SETXDMA;

214   1       FILLSOURCE: PROCEDURE;
215   2           DECLARE I BYTE;
216   2           ZN: PROCEDURE;
217   3               NSOURCE = 0;
218   3               END ZN;

219   2            CALL ZN;
220   2           CALL SELECT(SDISK);
221   2               DO I = 0 TO NBUF;
222   3               CALL SETDMA(SBUFFADR+NSOURCE);
223   3               IF (DCNT := DISKREAD(.FCB)) <> 0 THEN
224   3                   DO; IF DCNT > 1 THEN CALL FERR;
227   4                   SBUFF(NSOURCE) = ENDFILE;
228   4                   I = NBUF;
229   4                   END;
                    ELSE
230   3                   NSOURCE = NSOURCE + SECTSIZE;
231   3               END;
232   2           CALL ZN;
233   2           END FILLSOURCE;

234   1       GETSOURCE: PROCEDURE BYTE;
235   2           DECLARE B BYTE;
236   2           IF NSOURCE >= BUFFLENGTH THEN CALL FILLSOURCE;
```



```
238   2           IF 'B := SBUFF(NSOURCE)) <> ENDFILE THEN
239   2               NSOURCE = NSOURCE + 1;
240   2           RETURN B;
241   2           END GETSOURCE;

242   1       WRITEDEST: PROCEDURE;
                /* WRITE OUTPUT BUFFER UP TO (NOT INCLUDING) NDEST.
                LOW 7 BITS OF NDEST ARE ZERO */
243   2           DECLARE (I,N) BYTE;
244   2           ZN: PROCEDURE;
245   3               NDEST = 0;
246   3               END ZN;

247   2           CALL SELECT(DDISK);
248   2           IF LOW((N := SHR(NDEST,SECTSHF) - 1)) = 255 THEN RETURN;
250   2           CALL ZN;
251   2               DO I = 0 TO N;
252   3               CALL SETDMA(DBUFFADR+NDEST);
253   3               IF DISKWRITE(.DFCB) <> 0 THEN CALL FERR;
255   3               NDEST = NDEST + SECTSIZE;
256   3               END;
257   2           CALL ZN;
258   2           END WRITEDEST;

259   1       PUTDEST: PROCEDURE(B);
260   2           DECLARE B BYTE;
261   2           IF NDEST >= BUFFLENGTH THEN CALL WRITEDEST;
263   2           DBUFF(NDEST) = B;
264   2           NDEST = NDEST + 1;
265   2           END PUTDEST;

266   1       PUTXFER: PROCEDURE(C);
267   2           DECLARE C BYTE;
                /* WRITE C TO XFER FILE */
268   2           IF XBP >= SECTSIZE THEN /* BUFFER OVERFLOW */
269   2               DO; CALL SETXDMA;
271   3               IF DISKWRITE(.XFCB) <> 0 THEN CALL FERR;
273   3               XBP = 0;
274   3               END;
275   2           XBUFF(XBP) = C; XBP = XBP + 1;
277   2           END PUTXFER;

278   1       FINIS: PROCEDURE;
279   2           MOVEUP: PROCEDURE;
280   3           CALL MOVE(16,.DFCB,.DFCB+16);
281   3           END MOVEUP;

                /* CLEAR OUTPUT */
282   2           DFUB = 0 ; /* SET UNFILLED BYTES - USED FOR ISIS-II COMPATIBIL
283   2               DO WHILE (LOW(NDEST) AND 7FH) <> 0;
284   3               DFUB = DFUB + 1; /* COUNTS UNFILLED BYTES IN LAST RECORD *
285   3               CALL PUTDEST(ENDFILE);
286   3               END;
287   2           CALL WRITEDEST;

288   2           CALL CLOSE(.DFCB);
289   2           IF DCNT = 255 THEN CALL FERR;
```

```
                /* RENAME OLD FILE TO BAK */
291   2         CALL SETTYPE(.('BAK')); CALL MOVEUP;
293   2         CALL SELECT(SDISK);
294   2         CALL MOVE(16,.FCB,.DFCB);
295   2         CALL RENAME(.DFCB);
296   2         CALL MOVEUP;
                /* RENAME $$$ TO OLD NAME */
297   2         CALL SETTYPE(.('$$$'));
298   2         CALL SELECT(DDISK);
299   2         CALL RENAME(.DFCB);
300   2         END FINIS;


301   1       DECLARE
                  LPP LIT '23',             /* LINES PER PAGE */
                  FORWARD LIT '1',
                  BACKWARD LIT '0',
                  RUBOUT LIT '07FH',
                  POUND LIT '23H',
                  MACSIZE LIT '128',        /* MAX MACRO SIZE */
                  SCRSIZE LIT '100',        /* SCRATCH BUFFER SIZE */
                  COMSIZE LIT 'ADDRESS';  /* DETERMINES MAX COMMAND NUMBER*/

302   1       DCL MACRO(MACSIZE) BYTE,
                  SCRATCH(SCRSIZE) BYTE,      /* SCRATCH BUFFER FOR F,N,S */
                  (WBP, WBE, WBJ)  BYTE,      /* END OF F STRING, S STRING, J STR
                  (FLAG, MP, MI, XP) BYTE,
                  MT COMSIZE;

303   1       DCL (START, RESTART, OVERCOUNT, OVERFLOW, RESET, BADCOM) LABEL;

304   1       DCL INSERTING BYTE,     /* TRUE IF INSERTING CHARACTERS */
                  READBUFF  BYTE;     /* TRUE IF END OF READ BUFFER */

305   1       DECLARE
                  EOS LITERALLY '0FFH';



306   1       PRINTNMAC: PROCEDURE(CHAR);
307   2           DECLARE CHAR BYTE;
                  /* PRINT IF NOT IN MACRO EXPANSION */
308   2           IF MP <> 0 THEN RETURN;
310   2           CALL PRINTC(CHAR);
311   2           END PRINTNMAC;



312   1       DECLARE TRANSLATE BYTE, /* TRUE IF TRANSLATION TO UPPER CASE */
                  UPPER BYTE;         /* TRUE IF GLOBALLY TRANLATING TO UC */

313   1       LOWERCASE: PROCEDURE(C) BYTE;
314   2           DECLARE C BYTE;
                  /* RETURN TRUE IF LOWER CASE ALPHABETIC */
315   2           RETURN C >= LCA AND C <= LCZ;
316   2           END LOWERCASE;
```



```
317   1       UCASE: PROCEDURE(C) BYTE;
318   2           DECLARE C BYTE;
                  /* TRANSLATE C TO UPPER CASE */
319   2           IF LOWERCASE(C) THEN RETURN C AND 5FH;
321   2           RETURN C;
322   2           END UCASE;

323   1       UTRAN: PROCEDURE(C) BYTE;
324   2           DECLARE C BYTE;
                  /* TRANSLATE TO UPPER CASE IF ALPHABETIC LOWER AND TRANSLATE *
325   2           IF TRANSLATE THEN RETURN UCASE(C);
327   2           RETURN C;
328   2           END UTRAN;

329   1       PRINTVALUE: PROCEDURE(V);
                  /* PRINT THE LINE VALUE V */
330   2           DECLARE (D,ZERO) BYTE,
                      (K,V) ADDRESS;
331   2           K = 10000;
332   2           ZERO = FALSE;
333   2               DO WHILE K <> 0;
334   3               D = LOW(V/K); V = V MOD K;
336   3               K = K / 10;
337   3               IF ZERO OR  D <> 0 THEN
338   3                   DO; ZERO = TRUE;
340   4                   CALL PRINTC('0'+D);
341   4                   END; ELSE
342   3                   CALL PRINTC(' ');
343   3                   END;
344   2           END PRINTVALUE;

345   1       PRINTLINE: PROCEDURE(V);
346   2           DECLARE V ADDRESS;
347   2           IF NOT LINESET THEN RETURN;
349   2           CALL PRINTVALUE(V);
350   2           CALL PRINTC(':');
351   2           CALL PRINTC(' ');
352   2           IF INSERTING THEN CALL PRINTC(' '); ELSE
354   2               CALL PRINTC('*');
355   2           END PRINTLINE;


356   1       PRINTBASE: PROCEDURE;
357   2           CALL PRINTLINE(BASELINE);
358   2           END PRINTBASE;

359   1       PRINTNMBASE: PROCEDURE;
360   2           IF MP <> 0 THEN RETURN;
362   2           CALL PRINTBASE;
363   2           END PRINTNMBASE;

364   1       READC: PROCEDURE BYTE;
                  /* MAY BE MACRO EXPANSION */
365   2           IF MP > 0 THEN
366   2               DO;
367   3               IF BREAK$KEY THEN GO TO OVERCOUNT;
```

```
369   3               IF XP > MP THEN
370   3                   DO; /* START AGAIN */
371   4                   IF MT <> 0 THEN
372   4                       DO; IF (MT:=MT-1) = 0 THEN
374   5                           GO TO OVERCOUNT;
375   5                       END;
376   4                   XP = 0;
377   4                   END;
378   3               RETURN UTRAN(MACRO((XP := XP + 1) - 1));
379   3               END;
380   2           IF INSERTING THEN RETURN UTRAN(READCHAR);

                  /* GET COMMAND LINE */
382   2           IF READBUFF THEN
383   2               DO; READBUFF = FALSE;
385   3               IF LINESET AND COLUMN = 0 THEN
386   3                   DO;
387   4                   IF BACK > MAXM THEN
388   4                       CALL PRINTLINE(0); ELSE
389   4                       CALL PRINTBASE;
390   4                   END; ELSE
391   3               CALL PRINTC('*');
392   3               CALL READCOM; CBP = 0;
394   3               CALL PRINTC(LF);
395   3               COLUMN = 0;
396   3               END;
397   2           IF (READBUFF := CBP = COMLEN ) THEN COMBUFF(CBP) = CR;
399   2           RETURN UTRAN(COMBUFF((CBP := CBP +1) -1));
400   2           END READC;

401   1       SETRDMA: PROCEDURE;
                    /* SET READ LIB DMA ADDRESS */
402   2           CALL SELECT(SDISK);
403   2           CALL SETDMA(.BUFF);
404   2           END SETRDMA;

405   1       READFILE: PROCEDURE BYTE;
406   2           IF RBP >= SECTSIZE THEN
407   2               DO; CALL SETRDMA;
409   3               IF DISKREAD(.RFCB) <> 0 THEN RETURN ENDFILE;
411   3               RBP = 0;
412   3               END;
413   2           RETURN UTRAN(BUFF((RBP := RBP + 1) - 1));
414   2           END READFILE;

415   1       DCL (DISTANCE, TDIST) COMSIZE,
                (DIRECTION, CHAR) BYTE,
                ( FRONT, BACK, FIRST, LAST) ADDR;

416   1       SETFF: PROCEDURE;
417   2           DISTANCE = 0FFFFH;
418   2           END SETFF;

419   1       DISTZERO: PROCEDURE BYTE;
                    /* RETURN TRUE IF DISTANCE IS ZERO */
420   2           RETURN DISTANCE = 0;
421   2           END DISTZERO;
```



```
422   1       ZERODIST: PROCEDURE;
423   2           DISTANCE = 0;
424   2           END ZERODIST;

425   1       DISTNZERO: PROCEDURE BYTE;
                    /* CHECK FOR ZERO DISTANCE AND DECREMENT */
426   2           IF NOT DISTZERO THEN
427   2               DO; DISTANCE = DISTANCE - 1;
429   3               RETURN TRUE;
430   3               END;
431   2           RETURN FALSE;
432   2           END DISTNZERO;

433   1       SETLIMITS: PROC;
434   2           DCL (I,K,L,M) ADDR, (MIDDLE,LOOPING) BYTE;
435   2           RELLINE = 1; /* RELATIVE LINE COUNT */
436   2           IF DIRECTION = BACKWARD THEN
437   2               DO; DISTANCE = DISTANCE+1; I = FRONT; L = 0; K = 0FFFFH;
442   3               END; ELSE
443   2               DO; I = BACK; L = MAXM; K = 1;
447   3               END;

448   2           LOOPING = TRUE;
449   2           DO WHILE LOOPING;
450   3               DO WHILE (MIDDLE := I <> L) AND
                            MEMORY(M:=I+K) <> LF;
451   4               I = M;
452   4               END;
453   3           RELLINE = RELLINE - 1;
454   3           LOOPING = (DISTANCE := DISTANCE - 1) <> 0;
455   3           IF NOT MIDDLE THEN
456   3               DO; LOOPING = FALSE;
458   4               I = I - K;
459   4               END; ELSE
460   3           IF LOOPING THEN I = M;
                  END;

463   2           IF DIRECTION = BACKWARD THEN
464   2               DO; FIRST = I; LAST = FRONT - 1;
467   3               END; ELSE
468   2               DO; FIRST = BACK + 1; LAST = I + 1;
471   3               END;
472   2           END SETLIMITS;

473   1       INCFRONT: PROC; FRONT = FRONT + 1;
475   2           END INCFRONT;
476   1       INCBACK: PROCEDURE; BACK = BACK + 1;
478   2           END INCBACK;
479   1       DECFRONT: PROC; FRONT = FRONT - 1;
481   2           END DECFRONT;
482   1       DECBACK: PROC; BACK = BACK - 1;
484   2           END DECBACK;
485   1       INCBASE: PROCEDURE;
486   2           BASELINE = BASELINE + 1;
487   2           END INCBASE;
```

```
488   1       MEM$MOVE: PROC(MOVEFLAG);
489   2           DECLARE (MOVEFLAG,C) BYTE;
                  /* MOVE IF MOVEFLAG IS TRUE */
490   2           IF DIRECTION = FORWARD THEN
491   2               DO WHILE BACK < LAST; CALL INCBACK;
493   3               IF MOVEFLAG THEN
494   3                   DO;
495   4                   IF (C := MEMORY(BACK)) = LF THEN CALL INCBASE;
497   4                   MEMORY(FRONT) = C; CALL INCFRONT;
499   4                   END;
500   3               END; ELSE
501   2               DO WHILE FRONT > FIRST; CALL DECFRONT;
503   3               IF (C := MEMORY(FRONT)) = LF THEN BASELINE = BASELINE - 1;
505   3               IF MOVEFLAG THEN
506   3                   DO; MEMORY(BACK) = C; CALL DECBACK;
509   4                   END;
510   3               END;
511   2           END MEM$MOVE;

512   1       MOVER: PROC;
513   2           CALL MEM$MOVE(TRUE);
514   2           END MOVER;

515   1       SETPTRS: PROC;
516   2           CALL MEM$MOVE(FALSE);
517   2           END SETPTRS;

518   1       MOVELINES: PROC;
519   2           CALL SETLIMITS;
520   2           CALL MOVER;
521   2           END MOVELINES;

522   1       SETCLIMITS: PROC;
523   2           IF DIRECTION = BACKWARD THEN
524   2               DO; LAST = BACK;
526   3               IF DISTANCE > FRONT THEN
527   3                   FIRST = 1; ELSE FIRST = FRONT - DISTANCE;
529   3               END; ELSE
530   2               DO; FIRST = FRONT;
532   3               IF DISTANCE >= MAX - BACK THEN
533   3                   LAST = MAXM; ELSE LAST = BACK + DISTANCE;
535   3               END;
536   2           END SETCLIMITS;

537   1       READLINE: PROCEDURE;
538   2           DECLARE B BYTE;
                  /* READ ANOTHER LINE OF INPUT */
539   2           CTRAN: PROCEDURE(B) BYTE;
540   3               DECLARE B BYTE;
                      /* CONDITIONALLY TRANSLATE TO UPPER CASE ON INPUT */
541   3               IF UPPER THEN RETURN UTRAN(B);
543   3               RETURN B;
544   3               END CTRAN;
545   2           DO FOREVER;
546   3           IF FRONT >= BACK THEN GO TO OVERFLOW;
548   3           IF (B := CTRAN(GETSOURCE)) = ENDFILE THEN
```



```
549   3               DO; CALL ZERODIST; RETURN;
552   4               END;
553   3           MEMORY(FRONT) = B;
554   3           CALL INCFRONT;
555   3           IF B = LF THEN
556   3               DO; CALL INCBASE;
558   4               RETURN;
559   4               END;
560   3           END;
561   2           END READLINE;

562   1       WRITELINE: PROCEDURE;
                  /* WRITE ONE LINE OUT */
563   2           DECLARE B BYTE;
564   2               DO FOREVER;
565   3               IF BACK >= MAXM THEN /* EMPTY *
566   3                   DO; CALL ZERODIST; RETURN;
569   4                   END;
570   3               CALL INCBACK;
571   3               CALL PUTDEST(B:=MEMORY(BACK));
572   3               IF B = LF THEN
573   3                   DO; CALL INCBASE;
575   4                   RETURN;
576   4                   END;
577   3               END;
578   2           END WRITELINE;

579   1       WRHALF: PROCEDURE;
                   /* WRITE LINES UNTIL AT LEAST HALF THE BUFFER IS EMPTY */
580   2           CALL SETFF;
581   2               DO WHILE DISTNZERO;
582   3               IF HMAX >= (MAXM - BACK) THEN CALL ZERODIST; ELSE
584   3               CALL WRITELINE;
585   3               END;
586   2           END WRHALF;

587   1       WRITEOUT: PROCEDURE;
                  /* WRITE LINES DETERMINED BY 'DISTANCE',
                  CALLED FROM W AND E COMMANDS */
588   2           DIRECTION = BACKWARD; FIRST = 1; LAST = BACK;
591   2           CALL MOVER;
592   2           IF DISTZERO THEN CALL WRHALF;
                  /* DISTANCE = 0 IF CALL WRHALF */
594   2               DO WHILE DISTNZERO;
595   3               CALL WRITELINE;
596   3               END;
597   2           IF BACK < LAST THEN
598   2               DO; DIRECTION = FORWARD; CALL MOVER;
601   3               END;
602   2           END WRITEOUT;

603   1       CLEARMEM: PROCEDURE;
                  /* CLEAR MEMORY BUFFER */
604   2           CALL SETFF;
605   2           CALL WRITEOUT;
606   2           END CLEARMEM;
```

```
607   1       TERMINATE: PROCEDURE;
                  /* CLEAR BUFFERS */
608   2           CALL CLEARMEM;
609   2               DO WHILE (CHAR := GETSOURCE) <> ENDFILE;
610   3               CALL PUTDEST(CHAR);
611   3               END;
612   2           CALL FINIS;
613   2           END TERMINATE;


614   1       INSERT: PROCEDURE;
                  /* INSERT CHAR INTO MEMORY BUFFER */
615   2           IF FRONT = BACK THEN GO TO OVERFLOW;
617   2           MEMORY(FRONT) = CHAR; CALL INCFRONT;
619   2           IF CHAR = LF THEN CALL INCBASE;
621   2           END INSERT;

622   1       SCANNING: PROCEDURE BYTE;
                  /* READ A CHARACTER AND CHECK FOR ENDFILE OR CR */
623   2           RETURN NOT ((CHAR := READC) = ENDFILE OR
                               (CHAR = CR AND NOT INSERTING));
624   2           END SCANNING;

625   1       COLLECT: PROCEDURE;
                  /* READ COMMAND BUFFER AND INSERT CHARACTERS INTO
                  SCRATCH 'TIL NEXT CONTROL-Z OR CR FOR  FIND, NEXT, JUXT, OR
                  SUBSTITUTE COMMANDS - FILL AT WBE AND INCREMENT WBE SO IT
                  ADDRESSES NEXT EMPTY POSITION OF SCRATCH */
626   2           SETSCR: PROCEDURE;
627   3               SCRATCH(WBE) = CHAR;
628   3               IF (WBE := WBE + 1) >= SCRSIZE THEN GO TO OVERFLOW;
630   3               END SETSCR;
631   2               DO WHILE SCANNING;
632   3               IF CHAR = CTLL THEN
633   3                   DO; CHAR = CR; CALL SETSCR;
636   4                   CHAR = LF;
637   4                   END;
638   3               IF CHAR = 0 THEN GO TO BADCOM;
640   3               CALL SETSCR;
641   3               END;
642   2           END COLLECT;

643   1       FIND: PROCEDURE(PA,PB) BYTE;
644   2           DECLARE (PA,PB) BYTE;
                  /* FIND THE STRING IN SCRATCH STARTING AT PA AND ENDING AT PB
645   2           DECLARE J ADDRESS,
                      (K, MATCH) BYTE;
646   2           J = BACK ;
647   2           MATCH = FALSE;
648   2               DO WHILE NOT MATCH AND (MAXM > J);
649   3               LAST,J = J + 1; /* START SCAN AT J */
650   3               K = PA ; /* ATTEMPT STRING MATCH AT K */
651   3                   DO WHILE SCRATCH(K) = MEMORY(LAST) AND
                            NOT (MATCH := K = PB);
                        /* MATCHED ONE MORE CHARACTER */
652   4                   K = K + 1; LAST = LAST + 1;
654   4                   END;
```



```
655   3               END;
656   2           IF MATCH THEN /* MOVE STORAGE */
657   2               DO; LAST = LAST - 1; CALL MOVER;
660   3               END;
661   2           RETURN MATCH;
662   2           END FIND;

663   1       SETFIND: PROCEDURE;
                  /* SETUP THE SEARCH STRING FOR F,N, AND S COMMANDS */
664   2           WBE = 0; CALL COLLECT; WBP = WBE;
667   2           END SETFIND;

668   1       CHKFOUND: PROCEDURE;
                  /* CHECK FOR FOUND STRING IN F AND S COMMANDS */
669   2           IF NOT FIND(0,WBP) THEN /* NO MATCH */ GO TO OVERCOUNT;
671   2           END CHKFOUND;



672   1       SETRFCB: PROCEDURE;
                  /* PLACE CHAR INTO READ FILE CONTROL BLOCK AND INCREMENT */
673   2           RFCB((RBP := RBP + 1) - 1) = UCASE(CHAR);
674   2           END SETRFCB;

675   1       PRINTREL: PROCEDURE;
676   2           CALL PRINTLINE(BASELINE+RELLINE);
677   2           END PRINTREL;

678   1       TYPELINES: PROCEDURE;
679   2           DCL I ADDR;
680   2           DCL C BYTE;
681   2           CALL SETLIMITS;
                  /* DISABLE THE * PROMPT */
682   2           INSERTING = TRUE;
683   2           IF DIRECTION = FORWARD THEN
684   2               DO; RELLINE = 0; I = FRONT;
687   3               END; ELSE
688   2               I = FIRST;
689   2           IF (C := MEMORY(I-1)) = LF AND COLUMN <> 0 THEN
690   2               CALL CRLF;
691   2               DO I = FIRST TO LAST;
692   3               IF C = LF THEN
693   3                   DO;
694   4                   CALL PRINTREL;
695   4                   RELLINE = RELLINE + 1;
696   4                   IF BREAK$KEY THEN GO TO OVERCOUNT;
698   4                   END;
699   3               CALL PRINTC(C:=MEMORY(I));
700   3               END;
701   2           END TYPELINES;

702   1       SETLPP: PROCEDURE;
                  /* SET DISTANCE TO LINES PER PAGE */
703   2           DISTANCE = LPP;
704   2           END SETLPP;

705   1       SAVEDIST: PROCEDURE;
P
```

```
706   2          TDIST = DISTANCE;
707   2          END SAVEDIST;

708   1      RESTDIST: PROCEDURE;
709   2          DISTANCE = TDIST;
710   2          END RESTDIST;

711   1      PAGE: PROCEDURE;
712   2          DECLARE I BYTE;
713   2          CALL SAVEDIST;
714   2          CALL SETLPP;
715   2          CALL MOVELINES;
716   2          I = DIRECTION;
717   2          DIRECTION = FORWARD;
718   2          CALL SETLPP;
719   2          CALL TYPELINES;
720   2          DIRECTION = I;
721   2          IF LAST = MAXM OR FIRST = 1 THEN CALL ZERODIST;
723   2              ELSE CALL RESTDIST;
724   2          END PAGE;

725   1      WAIT: PROCEDURE;
                  /* 1/2 SECOND TIME OUT */
726   2          DECLARE I BYTE;
727   2              DO I = 0 TO 19;
728   3              IF BREAK$KEY THEN GO TO RESET;
730   3              CALL TIME(250);
731   3              END;
732   2          END WAIT;

733   1      SETFORWARD: PROCEDURE;
734   2          DIRECTION = FORWARD;
735   2          DISTANCE = 1;
736   2          END SETFORWARD;

737   1      APPHALF: PROCEDURE;
                  /* APPEND 'TIL BUFFER IS AT LEAST HALF FULL */
738   2          CALL SETFF; /* DISTANCE = 0FFFFH */
739   2              DO WHILE DISTNZERO;
740   3              IF FRONT >= HMAX THEN CALL ZERODIST; ELSE
742   3              CALL READLINE;
743   3              END;
744   2          END APPHALF;

745   1      INSCRLF: PROCEDURE;
                  /* INSERT CR LF CHARACTERS */
746   2          CHAR = CR; CALL INSERT;
748   2          CHAR = LF; CALL INSERT;
750   2          END INSCRLF;

751   1      TESTCASE: PROCEDURE;
752   2          DECLARE T BYTE;
                  /* TEST FOR UPPER OR LOWER CASE COMMAND AND SET TRANSLATE
                  FLAG (USED TO DETERMINE IF CHARACTERS WHICH FOLLOW GO TO UPPER
753   2          TRANSLATE = TRUE;
754   2          T = LOWERCASE(CHAR);
755   2          CHAR = UTRAN(CHAR);
```



```
756   2          TRANSLATE = UPPER OR NOT T;
757   2          END TESTCASE;

758   1      READCTRAN: PROCEDURE;
                  /* SET TRANSLATE TO FALSE AND READ NEXT CHARACTER */
759   2          TRANSLATE = FALSE;
760   2          CHAR = READC;
761   2          CALL TESTCASE;
762   2          END READCTRAN;

763   1      SINGLECOM: PROCEDURE(C) BYTE;
                  /* RETURN TRUE IF COMMAND IS ONLY CHARACTER, NOT IN MACRO */
764   2          DECLARE C BYTE;
765   2          RETURN CHAR = C AND COMLEN = 1 AND MP = 0;
766   2          END SINGLECOM;

767   1      SINGLERCOM: PROCEDURE(C) BYTE;
768   2          DECLARE C BYTE;
                  /* RETURN TRUE IF COMMAND IS ONLY CHARACTER, NOT IN MACRO, AND
                  THE OPERATOR HAS RESPONDED WITH 'Y' TO A Y/N REQUEST */
769   2          IF SINGLECOM(C) THEN
770   2              DO; CALL CRLF; CALL PRINTCHAR(C);
773   3              CALL MON1(9,.('~(Y/N)',WHAT,'$'));
774   3              C = UCASE(READCHAR); CALL CRLF;
776   3              IF C <> 'Y' THEN GO TO START;
778   3              RETURN TRUE;
779   3              END;
780   2          RETURN FALSE;
781   2          END SINGLERCOM;
```



```
            /* INITIALIZE THE SYSTEM */

782   1      EDCOMMAND: /* PAST LXI SP,STACK */
                  /* I/O BUFFER REGION IS 1 B AVAILABLE MEMORY */
                  NBUF = SHR(MAX := MAXB - .MEMORY,SECTSHF+3) - 1;
                  /* NBUF IS NUMBER OF BUFFERS - 1 */
783   1          BUFFLENGTH = SHL(DOUBLE(NBUF+1),SECTSHF+1);
                  /* NOW SET MAX AS REMAINDER OF FREE MEMORY */
784   1          IF BUFFLENGTH + 1024 > MAX THEN
785   1              DO; CALL PRINT(.('NO MEMORY$'));
787   2              CALL BOOT;
788   2              END;
                  /* REMOVE BUFFER SPACE AND 00 AT END OF MEMORY VECTOR */
789   1          MAX = MAX - BUFFLENGTH - 1;
                  /* RESET BUFFER LENGTH FOR I AND O */
790   1          BUFFLENGTH = SHR(BUFFLENGTH,1);
791   1          SBUFFADR = MAXB - BUFFLENGTH;
792   1          DBUFFADR = SBUFFADR - BUFFLENGTH;
793   1          MEMORY(MAX) = 0; /* STOPS MATCH AT END OF BUFFER */
794   1          MAXM = MAX - 1;
795   1          HMAX = SHR(MAXM,1);
                  /* NO TRANSLATE, WITH LINE NUMBERS */
```

```
796   1          UPPER, PRINTSUPPRESS = FALSE;
797   1          LINESET = TRUE;
                 /* GET SOURCE AND DESTINATION DISKS */
798   1          IF (FCB(1) = ' ') OR (FCB(17) <> ' ') THEN CALL FERR;
800   1          IF (SDISK := FCB(0)) = 0 THEN SDISK = CSELECT; ELSE
802   1              DO; SDISK = SDISK - 1; FCB(0) = 0; /* CLEAR DISK NAME */
805   2              END;
806   1          IF (DDISK := FCB(16)) = 0 THEN DDISK = SDISK; ELSE
808   1              DDISK = DDISK - 1;
                 /* CLEAR THE XFER FILE */
809   1          CALL XCLEAR;

810   1      RESTART:
                 CALL SETUP;
811   1          MEMORY(0) = LF;
812   1          FRONT = 1; BACK = MAXM;
814   1          COLUMN = 0;
815   1          GO TO START;

816   1      OVERCOUNT: FLAG = POUND; GO TO RESET;

818   1      BADCOM: FLAG = WHAT; GO TO RESET;

820   1      OVERFLOW: /* ARRIVE HERE ON OVERFLOW CONDITION (I,F,S COMMAND) */
                 FLAG = '>';

821   1      RESET: /* ARRIVE HERE ON ERROR CONDITION */
                 PRINTSUPPRESS = FALSE;
822   1          CALL PRINT(.('BREAK "$'));
823   1          CALL PRINTC(FLAG);
824   1          CALL PRINTM(.('" AT $'));
825   1          CALL PRINTC(CHAR);
826   1          CALL CRLF;


827   1      START:
                 READBUFF = TRUE;
828   1          MP = 0;


829   1          DO FOREVER; /* OR UNTIL THE POWER IS TURNED OFF */

                 /* ************************************************************
                 SIMPLE COMMANDS (CANNOT BE PRECEDED BY DIRECTION/DISTANCE):
                          E       END THE EDIT NORMALLY
                          H       MOVE TO HEAD OF EDITED FILE
                          I       INSERT CHARACTERS
                          O       RETURN TO THE ORIGINAL FILE
                          R       READ FROM LIBRARY FILE
                          Q       QUIT EDIT WITHOUT CHANGES TO ORIGINAL FILE
                    ************************************************************



830   2          INSERTING = FALSE;
831   2          CALL READCTRAN;
832   2          MI = CBP; /* SAVE STARTING ADDRESS FOR <CR> COMMAND */
```



```
833   2          IF SINGLECOM('E') THEN
834   2              DO; CALL TERMINATE;
836   3              IF SDISK <> DDISK THEN /* CHANGE DISKS */
                         /* USER CODE IN HIGH NIBBLE */
837   3                  BDISK = (BDISK AND 0F0H) OR (DDISK AND 0FH);
838   3              CALL REBOOT;
839   3              END; ELSE


840   2          IF SINGLECOM('H') THEN /* GO TO TOP */
841   2              DO; CALL TERMINATE;
843   3              CHAR = DDISK; DDISK = SDISK; SDISK = CHAR;
                     /* PING - PONG DISKS */
846   3              GO TO RESTART;
847   3              END; ELSE


848   2          IF CHAR = 'I' THEN /* INSERT CHARACTERS */
849   2              DO;
850   3              IF (INSERTING := (CBP = COMLEN) AND (MP = 0)) THEN
851   3                  CALL PRINTNMBASE;
852   3              SCOLUMN = COLUMN; /* STARTING COLUMN POSITION */
853   3              DO WHILE SCANNING;
854   4                  DO WHILE CHAR <> 0;
855   5                  IF CHAR=CTLU OR CHAR=CTLX OR CHAR=CTLR THEN
                             /* LINE DELETE OR RETYPE */
856   5                      DO;
857   6                      DISTANCE = 0; DIRECTION = BACKWARD;
                             /* ELIMINATE OR REPEAT THE LINE */
859   6                      IF CHAR = CTLR THEN
860   6                          DO; CALL CRLF;
862   7                          CALL TYPELINES;
863   7                          END; ELSE
                                 /* LINE DELETE */
864   6                          DO; CALL SETLIMITS; CALL SETPTRS;
867   7                          IF CHAR = CTLU THEN
868   7                              DO; CALL CRLF; CALL PRINTNMBASE;
871   8                              END; ELSE
                                 /* MUST BE CTLX */
872   7                              DO WHILE COLUMN > SCOLUMN;
873   8                              CALL BACKSPACE;
874   8                              END;
875   7                          END;
876   6                      END; ELSE
877   5                  IF CHAR = CTLH THEN
878   5                      DO;
879   6                      IF (TCOLUMN := COLUMN) > 0 THEN
880   6                          CALL PRINTNMAC(' '); /* RESTORE AFT BACKSP */
881   6                      IF FRONT > 1 AND TCOLUMN > SCOLUMN THEN
882   6                          DO;
883   7                          IF MEMORY(FRONT-1) <> LF THEN
884   7                              DO; /* CHARACTER CAN BE ELIMINATED */
885   8                              CALL DECFRONT;
886   8                              PRINTSUPPRESS = TRUE;
                                     /* BACKSPACE CHARACTER ACCEPTED */
887   8                              COLUMN = 0;
888   8                              DISTANCE = 0; DIRECTION = BACKWARD;
```

```
890   8                     CALL TYPELINES;
891   8                     PRINTSUPPRESS = FALSE;
                            /* COLUMN POSITION NOW RESET */
892   8                     IF (QCOLUMN := COLUMN) < SCOLUMN THEN
893   8                         QCOLUMN = SCOLUMN;
894   8                     COLUMN = TCOLUMN; /* ORIGINAL VALUE */
895   8                         DO WHILE COLUMN > QCOLUMN;
896   9                         CALL BACKSPACE;
897   9                         END;
898   8                     TCOLUMN = COLUMN;
899   8                     END;
900   7                 END;
901   6                 CHAR = 0;
902   6                 COLUMN = TCOLUMN;
903   6                 END; ELSE
904   5             IF CHAR = RUBOUT THEN
905   5                 DO; IF FRONT = 1 THEN GO TO RESET;
906   6                 CALL DECFRONT; CALL PRINTC(CHAR:=MEMORY(FRONT));
910   6                 IF CHAR = LF THEN BASELINE=BASELINE-1;
912   6                 CHAR = 0;
913   6                 END; ELSE
                      /* NOT A SPECIAL CASE */
914   5                 DO; IF NOT GRAPHIC(CHAR) THEN
916   6                     DO; CALL PRINTNMAC('^');
918   7                     CALL PRINTNMAC(CHAR + '@');
919   7                     END;
920   6                 IF CHAR = CTLL THEN
921   6                     CALL INSCRLF; ELSE
922   6                     DO; /* COLUMN COUNT GOES UP IF GRAPHIC */
                          /* COMPUTE OUTPUT COLUMN POSITION */
923   7                     IF MP = 0 THEN
924   7                         DO; IF CHAR >= ' ' THEN
926   8                         COLUMN = COLUMN + 1; ELSE
927   8                         IF CHAR = TAB THEN
928   8                         COLUMN = COLUMN + (8 - (COLUMN AND 111B));
                              END;
930   7                     CALL INSERT;
931   7                     END;
932   6                 END;
933   5             IF CHAR = LF THEN CALL PRINTNMBASE;
935   5             IF CHAR = CR THEN
936   5                 CALL PRINTNMAC(CHAR:=LF); ELSE CHAR = 0;
938   5             END;
939   4         END;
940   3         IF CHAR <> ENDFILE THEN  * MUST HAVE STOPPED ON CR */
941   3             CALL INSCRLF;
942   3         IF INSERTING AND LINESET THEN CALL CRLF;
944   3         END; ELSE


945   2     IF SINGLERCOM('O') THEN /* FORGET THIS EDIT */
946   2         GO TO RESTART; ELSE


947   2     IF CHAR = 'R' THEN
948   2         DO; DECLARE I BYTE;
                  /* READ FROM LIB FILE *
```



```
950   3             RBP = 1; CALL SETRDMA;
952   3                 DO WHILE SCANNING;
953   4                 IF RBP > 8 THEN GO TO OVERCOUNT;
955   4                 CALL SETRFCB;
956   4                 END;
957   3             CHAR = ' ';
958   3             IF (FLAG := RBP = 1) THEN /* READ FROM XFER FILE */
959   3                 DO;
960   4                 CALL MOVE(8,.XFCB(1),.RFCB(1));
961   4                 CALL CLOSE(.XFCB);
962   4                 END; ELSE /* LIB NAME SPECIFIED */
963   3                 DO WHILE RBP <= 8;
964   4                 CALL SETRFCB;
965   4                 END;
966   3             RFCB(12), RFCB(32) = 0; /* FILL REEL, AND NEXT RECORD */
967   3             CALL OPEN(.RFCB); RBP = SECTSIZE;
969   3             IF DCNT = 255 THEN
970   3                 DO; FLAG = 'O'; GO TO RESET;
973   4                 END;
974   3                 DO WHILE (CHAR := READBIIE) <> ENDFILE;
975   4                 CALL INSERT;
976   4                 END;
977   3             I = 0;
978   3             IF FLAG THEN /* MAY BE XFER DATA IN BUFFER */
979   3                 DO WHILE I < XBP;
980   4                 CHAR = XBUFF(I); I = I + 1;
982   4                 CALL INSERT;
983   4                 END;
984   3             END; ELSE


985   2         IF SINGLERCOM('Q') THEN
986   2             DO; CALL DELETE(.DFCB); CALL REBOOT;
989   3             END; ELSE



                  /* MAY BE A COMMAND WHICH HAS AN OPTIONAL DIRECTION AND DISTAN
990   2             DO; /* SCAN A SIGNED INTEGER VALUE (IF ANY) */
991   3             DCL I BYTE;

992   3             DIGIT: PROCEDURE BYTE;
993   4                 RETURN (I := CHAR - '0') <= 9;
994   4                 END DIGIT;

995   3             NUMBER: PROCEDURE;
996   4                 DISTANCE = 0;
997   4                     DO WHILE DIGIT;
998   5                     DISTANCE = SHL(DISTANCE,3) +
                                     SHL(DISTANCE,1) + I;
999   5                     CALL READCTRAN;
1000  5                     END;
                          /* RETURN WITH DISTANCE = NUMBER, CHAR = NEXT */
1001  4                 END NUMBER;

1002  3             RELDISTANCE: PROCEDURE;
1003  4                 IF DISTANCE > BASELINE THEN
```

```
1004  4                 DO; DIRECTION = FORWARD;
1006  5                 DISTANCE = DISTANCE - BASELINE;
1007  5                 END; ELSE
1008  4                 DO; DIRECTION = BACKWARD;
1010  5                 DISTANCE = BASELINE - DISTANCE;
1011  5                 END;
1012  4             END RELDISTANCE;

1013  3             CALL SETFORWARD;

1014  3             IF CHAR = '-' THEN
1015  3                 DO; CALL READCTRAN; DIRECTION = BACKWARD;
1018  4                 END;

1019  3             IF CHAR = POUND THEN
1020  3                 DO; CALL SETFF; CALL READCTRAN;
1023  4                 END; ELSE

1024  3             IF DIGIT THEN
1025  3                 DO; CALL NUMBER;
                          /* MAY BE ABSOLUTE LINE REFERENCE */
1027  4                 IF CHAR = ':' THEN
1028  4                     DO; CHAR = 'L';
1030  5                     CALL RELDISTANCE;
1031  5                     END;
1032  4                 END; ELSE

1033  3             IF CHAR = ':' THEN /* LEADING COLON */
1034  3                 DO; CALL READCTRAN; /* CLEAR THE COLON */
1036  4                 CALL NUMBER;
1037  4                 CALL RELDISTANCE;
1038  4                 IF DIRECTION = FORWARD THEN
1039  4                     DISTANCE = DISTANCE + 1;
1040  4                 END;

                      IF DISTZERO THEN DIRECTION = BACKWARD;
                      /* DIRECTION AND DISTANCE ARE NOW SET */


            /* ************************************************************
            MAY BE A COMMAND WHICH HAS DIRECTION AND DISTANCE SPECIFIED:
                    B       BEGINNING/BOTTOM OF BUFFER
                    C       MOVE CHARACTER POSITIONS
                    D       DELETE CHARACTERS
                    K       KILL LINES
                    L       MOVE LINE POSITION
                    P       PAGE UP OR DOWN (LPP LINES AND PRINT)
                    T       TYPE LINES
                    U       UPPER CASE TRANSLATE
                    V       VERIFY LINE NUMBERS
                   <CR>     MOVE UP OR DOWN LINES AND PRINT LINE
               ************************************************************/


1043  3             IF CHAR = 'B' THEN
1044  3                 DO; DIRECTION = 1 - DIRECTION;
1046  4                 FIRST = 1; LAST = MAXM; CALL MOVER;
```



```
1049  4                 END; ELSE


1050  3             IF CHAR = 'C' THEN
1051  3                 DO; CALL SETCLIMITS; CALL MOVER;
1054  4                 END; ELSE


1055  3             IF CHAR = 'D' THEN
1056  3                 DO; CALL SETCLIMITS;
1058  4                 CALL SETPTRS; /* SETS BACK/FRONT */
1059  4                 END; ELSE


1060  3             IF CHAR = 'K' THEN
1061  3                 DO; CALL SETLIMITS;
1063  4                 CALL SETPTRS;
1064  4                 END; ELSE


1065  3             IF CHAR = 'L' THEN CALL MOVELINES; ELSE

1067  3             IF CHAR = 'P' THEN /* PAGE MODE PRINT */
1068  3                 DO; IF DISTZERO THEN
1070  4                     DO; DIRECTION = FORWARD;
1072  5                     CALL SETLPP; CALL TYPELINES;
1074  5                     END; ELSE
1075  4                     DO WHILE DISTNZERO; CALL PAGE;
1077  5                     CALL WAIT;
1078  5                     END;
1079  4                 END; ELSE


1080  3             IF CHAR = 'T' THEN
1081  3                 CALL TYPELINES; ELSE


1082  3             IF CHAR = 'U' THEN
1083  3                 UPPER = DIRECTION = FORWARD; ELSE

1084  3             IF CHAR = 'V' THEN
1085  3                 DO; /* ØV DISPLAYS BUFFER STATE */
1086  4                 IF DISTZERO THEN
1087  4                     DO; CALL PRINTVALUE(BACK-FRONT);
1089  5                     CALL PRINTC('/');
1090  5                     CALL PRINTVALUE(MAXM);
1091  5                     CALL CRLF;
1092  5                     END; ELSE
1093  4                 LINESET = DIRECTION = FORWARD;
1094  4                 END; ELSE

1095  3             IF CHAR = CR THEN /* MAY BE MOVE/TYPE COMMAND */
1096  3                 DO;
1097  4                 IF MI = 1 AND MP = Ø THEN /* FIRST COMMAND */
1098  4                     DO; CALL MOVELINES; CALL SETFORWARD; CALL TYPELINE
1102  5                     END;
```

```
1103   4                END; ELSE

1104   3            IF DIRECTION = FORWARD OR DISTZERO THEN
1105   3            DO;

                /* ************************************************************
                COMMANDS WHICH ALLOW ONLY A PRECEDING NUMBER:
                        A       APPEND LINES
                        F       FIND NTH OCCURRENCE
                        M       APPLY MACRO
                        N       SAME AS F WITH AUTOSCAN THROUGH FILE
                        S       PERFORM N SUBSTITUTIONS
                        W       WRITE LINES TO OUTPUT FILE
                        X       TRANSFER (XFER) LINES TO TEMP FILE
                        Z       SLEEP
                   ************************************************************/



1106   4            IF CHAR = 'A' THEN
1107   4                DO; DIRECTION = FORWARD;
1109   5                FIRST = FRONT; LAST = MAXM; CALL MOVER;
                        /* ALL STORAGE FORWARD */
1112   5                IF DISTZERO THEN CALL APPHALF;
                        /* DISTANCE = 0 IF APPHALF CALLED */
1114   5                    DO WHILE DISTNZERO;
1115   6                    CALL READLINE;
1116   6                    END;
1117   5                DIRECTION = BACKWARD; CALL MOVER;
                        /* POINTERS REPOSITIONED */
1119   5                END; ELSE


1120   4            IF CHAR = 'F' THEN
1121   4                DO; CALL SETFIND; /* SEARCH STRING SCANNED
                        AND SETUP BETWEEN 0 AND WBP-1 IN SCRATCH */
1123   5                    DO WHILE DISTNZERO; CALL CHKFOUND;
1125   6                    END;
1126   5                END; ELSE


1127   4            IF CHAR = 'J' THEN /* JUXTAPOSITION OPERATION */
1128   4                DO; DECLARE T ADDRESS;
1130   5                CALL SETFIND; CALL COLLECT;
1132   5                WBJ = WBE; CALL COLLECT;
                        /* SEARCH FOR STRING 0 - WBP-1, INSERT STRING WBP TO W
                        AND THEN DELETE UP TO STRING WBJ TO WBE-1 */
1134   5                DO WHILE DISTNZERO; CALL CHKFOUND;
1136   6                /* INSERT STRING */ MI = WBP - 1;
1137   6                    DO WHILE (MI := MI + 1) < WBJ;
1138   7                    CHAR = SCRATCH(MI); CALL INSERT;
1140   7                    END;
1141   6                T = FRONT; /* SAVE POSITION FOR DELETE */
1142   6                IF NOT FIND(WBJ,WBE) THEN GO TO OVERCOUNT;
                        /* STRING FOUND, SO MOVE IT BACK */
1144   6                FIRST = FRONT - (WBE - WBJ);
1145   6                DIRECTION = BACKWARD; CALL MOVER;
```



```
                        /* NOW REMOVE THE INTERMEDIATE STRING */
1147   6                FRONT = T;
1148   6                END;
1149   5            END; ELSE


1150   4            IF CHAR = 'M' AND MP = 0 THEN /* MACRO DEFINITION */
1151   4                DO; XP = 255;
1153   5                IF DISTANCE = 1 THEN CALL ZERODIST;
1155   5                    DO WHILE (MACRO(XP := XP + 1) := READC) <> CR;
1156   6                    END;
1157   5                MP = XP; XP = 0; MT = DISTANCE;
1160   5                END; ELSE


1161   4            IF CHAR = 'N' THEN
1162   4                DO; /* SEARCH FOR STRING WITH AUTOSCAN */
1163   5                CALL SETFIND; /* SEARCH STRING SCANNED */
1164   5                    DO WHILE DISTNZERO;
                            /* FIND ANOTHER OCCURRENCE OF STRING */
1165   6                        DO WHILE NOT FIND(0,WBP); /* NOT IN BUFFER */
1166   7                        IF BREAK$KEY THEN GO TO RESET;
1168   7                        CALL SAVEDIST; CALL CLEARMEM;
                                /* MEMORY BUFFER WRITTEN */
1170   7                        CALL APPHALF;
1171   7                        DIRECTION = BACKWARD; FIRST = 1; CALL MOVER;
1174   7                        CALL RESTDIST; DIRECTION = FORWARD;
                                /* MAY BE END OF FILE */
1176   7                        IF BACK >= MAXM THEN GO TO OVERCOUNT;
1178   7                        END;
1179   6                    END;
1180   5                END; ELSE


1181   4            IF CHAR = 'S' THEN /* SUBSTITUTE COMMAND */
1182   4                DO; CALL SETFIND;
1184   5                CALL COLLECT;
                        /* FIND STRING FROM 0 TO WBP-1, SUBSTITUTE STRING
                        BETWEEN WBP AND WBE-1 IN SCRATCH */
1185   5                    DO WHILE DISTNZERO;
1186   6                    CALL CHKFOUND;
                            /* FRONT AND BACK NOW POSITIONED AT FOUND
                            STRING - REPLACE IT */
1187   6                    FRONT = FRONT - (MI := WBP); /* BACKED UP */
1188   6                        DO WHILE MI < WBE;
1189   7                        CHAR = SCRATCH(MI);
1190   7                        MI = MI + 1; CALL INSERT;
1192   7                        END;
1193   6                    END;
1194   5                END; ELSE


1195   4            IF CHAR = 'W' THEN
1196   4                CALL WRITEOUT; ELSE

1197   4            IF CHAR = 'X' THEN /* TRANSFER LINES */
1198   4                DO;
```

```
1199   5                CALL SETXDMA;
1200   5                IF DISTZERO THEN /* CLEAR THE FILE */
1201   5                    DO; CALL XCLEAR;
1203   6                    CALL DELETE(.XFCB);
1204   6                    END; ELSE
                        /* TRANSFER LINES */
1205   5                    DO; DECLARE I ADDRESS;
1207   6                    IF NOT XFERON THEN /* CREATE XFER FILE */
1208   6                        DO; CALL XCLEAR;
1210   7                        XFERON = TRUE;
1211   7                        CALL DELETE(.XFCB); /* OLD VERSION GONE */
1212   7                        CALL MAKE(.XFCB);
1213   7                        IF DONT = 255 THEN CALL FERR;
1215   7                        END;
1216   6                    CALL SETLIMITS;
1217   6                        DO I = FIRST TO LAST;
1218   7                        CALL PUTXFER(MEMORY(I));
1219   7                        END;
1220   6                    END;
1221   5                END; ELSE

1222   4            IF CHAR = 'Z' THEN /* SLEEP */
1223   4                DO;
1224   5                IF DISTZERO THEN
1225   5                    DO; IF READCHAR = ENDFILE THEN GO TO RESET;
1228   6                    END;
1229   5                    DO WHILE DISTNZERO; CALL WAIT;
1231   6                    END;
1232   5                END; ELSE
1233   4            IF CHAR <> 0 THEN /* NOT BREAK LEFT OVER FROM STOP */
                    /* DIRECTION FORWARD, BUT NOT ONE OF THE ABOVE */
1234   4            GO TO BADCOM;


                    END; ELSE /* DIRECTION NOT FORWARD */
1236   3                GO TO BADCOM;
1237   3                END;
1238   2            END;
1239   1        END;
```

```
0000 ED#
0000 ED#
0AF3 17      0AF3 18      0AFC 19      0AFC 22      0B00 24
0B07 25      0B08 26      0B13 27      0B14 29      0E18 31
0B20 32      0B24 33      0B2C 34      0B31 35      0B38 36
0B39 37      0B39 38      0B41 39      0B42 40      0B47 41
0B4C 42      0B51 43      0B56 44      0B57 45      0B5B 47
0B73 48      0B78 49      0B7D 50      0B8C 51      0B93 52
0B9A 53      0B9E 54      0B9F 56      0BA7 57      0BAA 58
0BCB 59      0BCB 60      0BCF 62      0BDA 64      0BDF 65
0BE7 66      0BE7 67      0BEE 68      0BEF 69      0BEF 70
0BF4 71      0BF9 72      0BFA 73      0C00 75      0C09 76
0C0A 77      0C10 79      0C13 80      0C1B 81      0C1C 82
0C22 84      0C2B 85      0C2C 87      0C32 89      0C3E 90
0C3F 91      0C45 93      0C51 94      0C52 95      0C58 97
0C64 98      0C65 99      0C6B 101     0C74 102     0C75 103
0C7B 105     0C85 106     0C85 107     0C8B 109     0C95 110
0C95 111     0C9B 113     0CA7 114     0CAB 115     0CAE 117
0CB7 118     0CB8 120     0CB8 121     0CBD 122     0CC3 123
0CC4 124     0CC4 125     0CD0 127     0CD8 128     0CDB 129
0CDB 130     0CDE 131     0CDE 132     0CDE 133     0CE7 134
0CE7 135     0CEB 137     0CF6 138     0CF7 139     0CFD 141
0D06 142     0D07 143     0D07 144     0D0E 145     0D14 146
0D17 147     0D18 149     0D1E 151     0D26 152     0D29 153
0D2C 154     0D2D 155     0D2D 156     0D33 157     0D39 158
0D3A 159     0D40 161     0D54 162     0D55 163     0D55 164
0D5B 165     0D61 166     0D70 167     0D80 168     0D8A 170
0D91 171     0D97 172     0D9F 173     0DA5 174     0DA5 175
0DAC 176     0DB2 177     0DBA 179     0DC0 180     0DC8 181
0DCB 182     0DD1 183     0DD4 184     0DD7 185     0DDF 187
0DE5 188     0DE8 189     0DE6 190     0DF3 191     0DF9 192
0DFF 193     0E05 194     0E0F 196     0E16 197     0E1C 198
0E1C 199     0E22 200     0E28 201     0E2E 202     0E33 203
0E3B 204     0E3E 205     0E44 206     0E45 207     0E45 208
0E59 209     0E5A 210     0E5A 211     0E61 212     0E67 213
0E68 214     0ED3 216     0ED3 217     0ED9 218     0E68 219
0E6B 220     0E72 221     0E81 222     0E8E 223     0E9C 225
0EA5 226     0EA8 227     0EB2 228     0EB8 229     0EBB 230
0EC5 231     0ECF 232     0ED2 233     0EDA 234     0EDA 236
0EE6 237     0EF9 238     0EFA 239     0F01 240     0F05 241
0F05 242     0F66 244     0F66 245     0F6C 246     0F05 247
0F0C 248     0F20 249     0F21 250     0F24 251     0F33 252
0F40 253     0F4B 254     0F4E 255     0F58 256     0F62 257
0F65 258     0F6D 259     0F71 261     0F7D 262     0F80 263
0F8C 264     0F93 265     0F94 266     0F98 268     0FA0 270
0FA3 271     0FAE 272     0FB1 273     0FB6 274     0FB6 275
0FC3 276     0FCA 277     0FCB 278     103B 279     103B 280
104B 281     0FCB 282     0FD0 283     0FDB 284     0FE2 285
0FE7 286     0FEA 287     0FED 288     0FF3 289     0FFB 290
0FFE 291     1004 292     1007 293     100E 294     101E 295
1024 296     1027 297     102D 298     1034 299     103A 300
104C 306     1050 308     1058 309     1059 310     1060 311
1061 313     1065 315     1079 316     1079 317     107D 319
1088 320     108E 321     1092 322     1092 323     1096 325
109D 326     10A5 327     10A9 328     10A9 329     10AF 331
10B5 332     10BA 333     10C6 334     10D4 335     10E1 336
10EF 337     10FF 339     1104 340     110D 341     1110 342
1115 343     1118 344     1119 345     111F 347     1127 348
1128 349     1130 350     1135 351     113A 352     1141 353
1149 354     114E 355     114F 356     114F 357     1157 358

1158 359     1158 360     1160 361     1161 362     1164 363
1165 364     1165 365     116E 367     1175 368     1178 369
1182 371     118E 373     119E 374     11A1 375     11A1 376
11A6 377     11A6 378     11BA 379     11BA 380     11C1 381
11C9 382     11D0 384     11D5 385     11E5 387     11F1 388
11FA 389     11FD 390     1200 391     1205 392     1208 393
120D 394     1212 395     1217 396     1217 397     1228 398
1233 399     1247 400     1247 401     1247 402     124E 403
1254 404     1255 405     1255 406     125D 408     1260 409
126B 410     126E 411     1273 412     1273 413     1287 414
1287 416     1287 417     128D 418     128E 419     128E 420
129B 421     129B 422     129B 423     12A1 424     12A2 425
12A2 426     12AA 428     12B1 429     12B4 430     12B4 431
12B7 432     12B7 433     12B7 435     12BD 436     12C5 438
12CC 439     12D2 440     12D8 441     12DC 442     12DF 444
12E5 445     12E6 446     12F1 447     12F1 448     12F6 449
12FD 450     132A 451     1330 452     1333 453     133A 454
134D 455     1355 457     135A 458     1368 459     136B 460
1372 461     1378 462     137B 463     1383 465     1389 466
1390 467     1393 469     139A 470     13A1 471     13A1 472
13A2 473     13A2 474     13A9 475     13AA 476     13AA 477
13B1 478     13B2 479     13B2 480     13B9 481     13BA 482
13BA 483     13C1 484     13C2 485     13C2 486     13C9 487
13CA 488     13CE 490     13D6 491     13E2 492     13E5 493
13EC 495     13FC 496     13FF 497     140A 498     140D 499
140D 500     1413 501     141F 502     1422 503     1432 504
1439 505     1440 507     144B 508     144E 509     144E 510
1451 511     1452 512     1452 513     1457 514     1458 515
1458 516     145D 517     145E 518     145E 519     1461 520
1464 521     1465 522     1465 523     146D 525     1473 526
147F 527     1488 528     1494 529     1497 531     149D 532
14AF 533     14B8 534     14C3 535     14C3 536     14C4 537
1504 539     1508 541     150F 542     1517 543     151B 544
14C4 545     14C4 546     14D0 547     14D3 548     14E2 550
14E5 551     14E6 552     14E6 553     14F1 554     14F4 555
14FC 557     14FF 558     1500 559     1500 560     1503 561
151B 562     151B 564     151B 565     1527 567     152A 568
152B 569     152B 570     152E 571     153D 572     1545 574
1548 575     1549 576     1549 577     154C 578     154D 579
154D 580     1550 581     1557 582     1569 583     156F 584
1572 585     1575 586     1576 587     1576 588     157B 589
1581 590     1587 591     158A 592     1591 593     1594 594
159B 595     159E 596     15A1 597     15AD 599     15B2 600
15B5 601     15B5 602     15B6 603     15B6 604     15B9 605
15BC 606     15BD 607     15BD 608     15C0 609     15CB 610
15D2 611     15D5 612     15D8 613     15D9 614     15D9 615
15E6 616     15E9 617     15F4 618     15F7 619     15FF 620
1602 621     1603 622     1603 623     1624 624     1624 625
1652 626     165E 627     165F 628     1633 634     1638 635
1624 631     162B 632     1633 634     164B 640     164E 641
1640 637     1640 638     1648 639     167B 647     1680 648
1651 642     166F 643     1675 646     16D3 652     16DA 653
1696 649     16A0 650     16A6 651     16E7 656     16FC 663
16E1 654     16E4 655     16E7 656     16FC 663     170B 668
16F8 660     16F8 661     1704 666     171D 672     1737 674
1701 665     1719 670     171C 671     1745 677     1746 681
1719 670     1738 675     1738 676     1756 685     1762 687
1738 675     1749 682     174E 683     175C 686     176E 691
1765 688     1765 688     176B 689     178C 690     17A1 692
```

```
17A9 694    17AC 695    17B3 696    17BA 697    17BD 698
17BD 699    17CC 700    17D9 701    17DA 702    17DA 703
17E0 704    17E1 705    17E1 706    17E7 707    17E8 708
17E8 709    17EE 710    17EF 711    17EF 713    17F2 714
17F5 715    17F8 716    17FL 717    1803 718    1806 719
1809 720    180F 721    1830 722    1836 723    1839 724
183A 725    183A 727    1848 728    184F 729    1852 730
1857 731    1861 732    1862 733    1862 734    1867 735
186D 736    186E 737    186E 738    1871 739    1878 740
1884 741    188A 742    188D 743    1890 744    1891 745
1891 746    1896 747    1899 748    189E 749    18A1 750
18A2 751    18A2 753    18A7 754    18B1 755    18BB 756
18C5 757    18C6 758    18C6 759    18CB 760    18D1 761
18D4 762    18D5 763    18D9 765    18FC 766    18FC 767
1900 769    190B 771    190E 772    1915 773    191D 774
1927 775    192A 776    1932 777    1935 778    1938 779
1938 780    193B 781    01C0 782    01DA 783    01E9 784
01F9 786    01FF 787    0202 788    0202 789    0211 790
021E 791    022A 792    0236 793    023F 794    0246 795
0253 796    025D 797    0262 798    027A 799    027D 800
0288 801    0291 803    0298 804    029D 805    029D 806
02A8 807    02B1 808    02B8 809    02BB 810    02BE 811
02C3 812    02C9 813    02CF 814    02D4 815    02D7 816
02DF 817    02E2 818    02EA 819    02ED 820    02F5 821
02FD 822    0303 823    030A 824    0310 825    0317 826
031A 827    0322 828    0327 829    0327 830    032C 831
032F 832    0335 833    033E 835    0341 836    034B 837
035C 838    035F 839    0362 840    036B 842    036E 843
0374 844    037A 845    0380 846    0383 847    0386 848
038E 850    03AB 851    03AE 852    03B4 853    03BB 854
03C3 855    03E7 857    03ED 858    03F2 859    03F9 861
03FC 862    03FF 863    0402 865    0405 866    0408 867
0410 869    0413 870    0416 871    0419 872    0423 873
0426 874    0429 875    0429 876    042C 877    0434 879
0441 880    0446 881    045F 883    046D 885    0470 886
0475 887    047A 888    0480 889    0485 890    0488 891
048D 892    049A 893    04A0 894    04A6 895    04B0 896
04B3 897    04B6 898    04BC 899    04BC 900    04BC 901
04C1 902    04C7 903    04CA 904    04D2 906    04DE 907
04E1 908    04E4 909    04F3 910    04FB 911    0502 912
0507 913    050A 915    0516 917    051B 918    0524 919
0524 920    052C 921    0532 923    053A 925    0542 926
054C 927    0554 928    0562 929    0562 930    0565 931
0565 932    0565 933    056D 934    0570 935    0578 936
0585 937    058A 938    058L 939    0590 940    0598 941
059B 942    05A6 943    05A9 944    05AC 945    05B5 946
05B8 947    05C0 950    05C5 951    05C8 952    05CF 953
05D8 954    05DB 955    05DE 956    05E1 957    05F6 958
05F5 960    0605 961    060B 962    060E 963    0617 964
061A 965    061D 966    0627 967    062D 968    0632 969
063A 971    063F 972    0642 973    0642 974    064D 975
0650 976    0653 977    065B 978    065F 979    0669 980
0676 981    067D 982    0680 983    0683 984    0686 985
068F 987    0695 988    069B 989    193B 992    193B 993
194A 994    194A 995    194A 996    1950 997    1957 998
196F 999    1972 1000   1975 1001   1976 1002   1976 1003
1982 1005   1987 1006   1995 1007   1998 1009   199D 1010
19A9 1011   19A9 1012   069B 1013   069E 1014   06A6 1016
06A9 1017   06AE 1018   06AE 1019   06B6 1021   06B9 1022
06BC 1023   06BF 1024   06C6 1026   06C9 1027   06D1 1029
06D6 1030   06D9 1031   06D9 1032   06DC 1033   06E4 1035
06E7 1036   06EA 1037   06ED 1038   06F5 1039   06FC 1040
06FC 1041   0703 1042   0708 1043   0710 1045   0717 1046
071D 1047   0723 1048   0726 1049   0729 1050   0731 1052
0734 1053   0737 1054   073A 1055   0742 1057   0745 1058
0748 1059   074B 1060   0753 1062   0756 1063   0759 1064
075C 1065   0764 1066   076A 1067   0772 1069   0779 1071
077E 1072   0781 1073   0784 1074   0787 1075   078E 1076
0791 1077   0794 1078   0797 1079   079A 1080   07A2 10E1
07A8 1082   07B0 1083   07BE 1084   07C6 1086   07CD 1088
07DB 1089   07E0 1090   07E8 1091   07EB 1092   07EE 1093
07F9 1094   07FC 1095   0804 1097   081C 1099   081F 1100
0822 1101   0825 1102   0825 1103   082B 1104   083B 1106
0843 1108   0848 1109   084E 1110   0854 1111   0857 1112
085E 1113   0861 1114   0868 1115   086B 1116   086E 1117
0873 1118   0876 1119   0879 1120   0881 1122   0884 1123
088B 1124   088E 1125   0891 1126   0894 1127   089C 1130
089F 1131   08A2 1132   08A8 1133   08AB 1134   08B2 1135
08B5 1136   08BC 1137   08CA 1138   08D7 1139   08DA 1140
08DD 1141   08E3 1142   08F3 1143   08F6 1144   0906 1145
090B 1146   090E 1147   0914 1148   0917 1149   091A 1150
0932 1152   0937 1153   0943 1154   0946 1155   0961 1156
0964 1157   096A 1158   096F 1159   0979 1160   097C 1161
0984 1163   0987 1164   098E 1165   099C 1166   09A3 1167
09A6 1168   09A9 1169   09AC 1170   09AF 1171   09B4 1172
09BA 1173   09BD 1174   09C0 1175   09C5 1176   09D1 1177
09D4 1178   09D7 1179   09DA 1180   09DD 1181   09E5 1183
09E8 1184   09EB 1185   09F2 1186   09F5 1187   0A06 1188
0A10 1189   0A1D 1190   0A24 1191   0A27 1192   0A2A 1193
0A2D 1194   0A30 1195   0A38 1196   0A3E 1197   0A46 1199
0A49 1200   0A50 1202   0A53 1203   0A59 1204   0A5C 1207
0A64 1209   0A67 1210   0A6C 1211   0A72 1212   0A78 1213
0A80 1214   0A83 1215   0A83 1216   0A86 1217   0A9B 1218
0AA3 1219   0AB0 1220   0AB0 1221   0AB3 1222   0ABB 1224
0AC2 1226   0ACA 1227   0ACD 1228   0ACD 1229   0AD4 1230
0AD7 1231   0ADA 1232   0ADD 1233   0AE5 1234   0AE8 1235
0AEB 1236   0AEE 1237   0AEE 1238   0AF1 1239
0000 MODULE#
```

```
                /* LOADCOM LOADS TRANSIENT COMMAND FILES TO THE DISK FROM THE
                   CURRENTLY DEFINED READER PERIPHERAL.  THE LOADER PLACES THE MA
                   CODE INTO A FILE WHICH APPEARS IN THE LOADCOM COMMAND */

 21   2        DECLARE
                   TRUE LITERALLY '1',
                   FALSE LITERALLY '0',
                   FOREVER LITERALLY 'WHILE TRUE',
                   CR LITERALLY '13',
                   LF LITERALLY '10',
                   WHAT LITERALLY '63';

 22   2        PRINTCHAR: PROCEDURE(CHAR);
 23   3            DECLARE CHAR BYTE;
 24   3            CALL MON1(2,CHAR);
 25   3            END PRINTCHAR;

 26   2        CRLF: PROCEDURE;
 27   3            CALL PRINTCHAR(CR);
 28   3            CALL PRINTCHAR(LF);
 29   3            END CRLF;

 30   2        PRINTNIB: PROCEDURE(N);
 31   3            DECLARE N BYTE;
 32   3                IF N < 9 THEN CALL PRINTCHAR(N+'A'-10); ELSE
 34   3                CALL PRINTCHAR(N+'0');
 35   3            END PRINTNIB;

 36   2        PRINTHEX: PROCEDURE(B);
 37   3            DECLARE B BYTE;
 38   3                CALL PRINTNIB(SHR(B,4)); CALL PRINTNIB(B AND 0FH);
 40   3            END PRINTHEX;

 41   2        PRINTADDR: PROCEDURE(A);
 42   3            DECLARE A ADDRESS;
 43   3            CALL PRINTHEX(HIGH(A)); CALL PRINTHEX(LOW(A));
 45   3            END PRINTADDR;

 46   2        PRINTM: PROCEDURE(A);
 47   3            DECLARE A ADDRESS;
 48   3            CALL MON1(9,A);
 49   3            END PRINTM;

 50   2        PRINT: PROCEDURE(A);
 51   3            DECLARE A ADDRESS;
                   /* PRINT THE STRING STARTING AT ADDRESS A UNTIL THE
                   NEXT DOLLAR SIGN IS ENCOUNTERED WITH PRECEDING CRLF */
 52   3            CALL CRLF;
 53   3            CALL PRINTM(A);
 54   3            END PRINT;

 55   2        DECLARE LA ADDRESS;  /* CURRENT LOAD ADDRESS */

 56   2        PERROR: PROCEDURE(A);
                   /* PRINT ERROR MESSAGE */
 57   3            DECLARE A ADDRESS;
 58   3            CALL PRINT(.('ERROR: $'));
```



```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE LOAD
OBJECT MODULE PLACED IN LOAD.OBJ
COMPILER INVOKED BY:  PLM80 LOAD.PLM DEBUG


  1          LOAD:
             DO;
                 /* C P / M   C O M M A N D   F I L E   L O A D E R

                 COPYRIGHT (C) 1976, 1977, 1978
                 DIGITAL RESEARCH
                 BOX 579 PACIFIC GROVE
                 CALIFORNIA 93950

                 */

  2   1      DECLARE
                 TPA    LITERALLY '0100H',    /* TRANSIENT PROGRAM AREA */
                 DFCBA LITERALLY '005CH',     /* DEFAULT FILE CONTROL BLOCK */
                 DBUFF LITERALLY '0080H';     /* DEFAULT BUFFER ADDRESS */

             /* JMP LOADCOM TO START LOAD */
  3   1      DECLARE JUMP BYTE DATA(0C3H);
  4   1      DECLARE JUMPA ADDRESS DATA(.LOADCOM);

  5   1      DECLARE COPYRIGHT(*) BYTE DATA
                 (' COPYRIGHT (C) 1978, DIGITAL RESEARCH ');

  6   1      MON1: PROCEDURE(F,A) EXTERNAL;
  7   2          DECLARE F BYTE, A ADDRESS;
  8   2          END MON1;

  9   1      MON2: PROCEDURE(F,A) BYTE EXTERNAL;
 10   2          DECLARE F BYTE, A ADDRESS;
 11   2          END MON2;

 12   1      DECLARE SP ADDRESS;

 13   1      BOOT: PROCEDURE;
 14   2          STACKPTR = SP;
 15   2          RETURN;
 16   2          END BOOT;


 17   1      LOADCOM: PROCEDURE;
 18   2          DECLARE FCB (33) BYTE AT (DFCBA),
                     FCBA LITERALLY 'DFCBA';
 19   2          DECLARE BUFFER (128) BYTE AT (DBUFF),
                     BUFFA LITERALLY 'DBUFF';
 20   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 */
```

```
 59   3          CALL PRINTM(.A);
 60   3          CALL PRINTM(.(', LOAD ADDRESS $'));
 61   3          CALL PRINTADDR(LA);
 62   3          CALL BOOT;
 63   3          END PERROR;

 64   2      DECLARE DCNT BYTE;

 65   2      OPEN: PROCEDURE(FCB);
 66   3          DECLARE FCB ADDRESS;
 67   3          DCNT = MON2(15,FCB);
 68   3          END OPEN;

 69   2      CLOSE: PROCEDURE(FCB);
 70   3          DECLARE FCB ADDRESS;
 71   3          DCNT = MON2(16,FCB);
 72   3          END CLOSE;

 73   2      SEARCH: PROCEDURE(FCB);
 74   3          DECLARE FCB ADDRESS;
 75   3          DCNT = MON2(17,FCB);
 76   3          END SEARCH;

 77   2      SEARCHN: PROCEDURE;
 78   3          DCNT = MON2(18,0);
 79   3          END SEARCHN;

 80   2      DELETE: PROCEDURE(FCB);
 81   3          DECLARE FCB ADDRESS;
 82   3          CALL MON1(19,FCB);
 83   3          END DELETE;

 84   2      DISKREAD: PROCEDURE(FCB) BYTE;
 85   3          DECLARE FCB ADDRESS;
 86   3          RETURN MON2(20,FCB);
 87   3          END DISKREAD;

 88   2      DISKWRITE: PROCEDURE(FCB) BYTE;
 89   3          DECLARE FCB ADDRESS;
 90   3          RETURN MON2(21,FCB);
 91   3          END DISKWRITE;

 92   2      MAKE: PROCEDURE(FCB);
 93   3          DECLARE FCB ADDRESS;
 94   3          DCNT = MON2(22,FCB);
 95   3          END MAKE;

 96   2      RENAME: PROCEDURE(FCB);
 97   3          DECLARE FCB ADDRESS;
 98   3          CALL MON1(23,FCB);
 99   3          END RENAME;

100   2          MOVE: PROCEDURE(S,D,N);
101   3              DECLARE (S,D) ADDRESS, N BYTE,
                      A BASED S BYTE, B BASED D BYTE;
102   3                  DO WHILE (N:=N-1) <> 255;
103   4                      B = A; S=S+1; D=D+1;
```



```
106   4                  END;
107   3              END MOVE;

108   2          GETCHAR: PROCEDURE BYTE;
                     /* GET NEXT CHARACTER */
109   3              DECLARE I BYTE;
110   3              IF (SBP := SBP+1) <= LAST(SBUFF) THEN
111   3                  RETURN SBUFF(SBP);
                     /* OTHERWISE READ ANOTHER BUFFER FULL */
112   3                  DO SBP = 0 TO LAST(SBUFF) BY 128;
113   4                  IF (I:=DISKREAD(.SFCB)) = 0 THEN
114   4                      CALL MOVE(80H,.SBUFF(SBP),80H); ELSE
115   4                      DO;
116   5                      IF I<>1 THEN CALL PERROR(.('DISK READ$'));
118   5                      SBUFF(SBP) = EOFILE;
119   5                      SBP = LAST(SBUFF);
120   5                      END;
121   4                  END;
122   3              SBP = 0; RETURN SBUFF(0);
124   3              END GETCHAR;
125   2      DECLARE
                 STACKPOINTER LITERALLY 'STACKPTR';

               /* INTEL HEX FORMAT LOADER */

126   2      RELOC: PROCEDURE;
127   3          DECLARE (RL, CS, RT) BYTE;
128   3          DECLARE
                     TA ADDRESS,      /* TEMP ADDRESS */
                     SA ADDRESS,      /* START ADDRESS */
                     FA ADDRESS,      /* FINAL ADDRESS */
                     NB ADDRESS,      /* NUMBER OF BYTES LOADED */

                 MBUFF(256) BYTE,
                 P BYTE,
                 L ADDRESS;

129   3          SETMEM: PROCEDURE(B);
                     /* SET MBUFF TO B AT LOCATION LA MOD LENGTH(MBUFF) */
130   4              DECLARE (B,I) BYTE;
131   4              IF LA < L THEN
132   4                  CALL PERROR(.('INVERTED LOAD ADDRESS$'));
133   4                  DO WHILE LA > L + LAST(MBUFF); /* WRITE A PARAGRAPH */
134   5                      DO I = 0 TO 127; /* COPY INTO BUFFER */
135   6                      BUFFER(I) = MBUFF(LOW(L)); L = L + 1;
137   6                      END;
                         /* WRITE BUFFER ONTO DISK */
138   5                  P = P + 1;
139   5                  IF DISKWRITE(FCBA) <> 0 THEN
140   5                      DO; CALL PERROR(.('DISK WRITE$'));
142   6                      END;
143   5                  END;
144   4              MBUFF(LOW(LA)) = B;
145   4              END SETMEM;

146   3          DIAGNOSE: PROCEDURE;
```

```
 147   4          DECLARE M BASED TA BYTE;

 148   4          NEWLINE: PROCEDURE;
 149   5              CALL CRLF; CALL PRINTADDR(TA); CALL PRINTCHAR(':');
 152   5              CALL PRINTCHAR(' ');
 153   5              END NEWLINE;

                  /* PRINT DIAGNOSTIC INFORMATION AT THE CONSOLE */
 154   4              CALL PRINT(.('LOAD  ADDRESS $')); CALL PRINTADDR(TA);
 156   4              CALL PRINT(.('ERROR ADDRESS $')); CALL PRINTADDR(LA);

 158   4              CALL PRINT(.('BYTES READ:$')); CALL NEWLINE;
 160   4              DO WHILE TA < LA;
 161   5              IF (LOW(TA) AND 0FH) = 0  THEN CALL NEWLINE;
 163   5              CALL PRINTHEX(MBUFF(TA-L)); TA=TA+1;
 165   5              CALL PRINTCHAR(' ');
 166   5              END;
 167   4          CALL CRLF;
 168   4          CALL BOOT;
 169   4          END DIAGNOSE;

 170   3          READHEX: PROCEDURE BYTE;
                      /* READ ONE HEX CHARACTER FROM THE INPUT */
 171   4              DECLARE H BYTE;
 172   4              IF (H := GETCHAR) - '0' <= 9 THEN RETURN H - '0';
 174   4              IF H - 'A' > 5 THEN
 175   4                  DO; CALL PRINT(.('INVALID HEX DIGIT$'));
 177   5                  CALL DIAGNOSE;
 178   5                  END;
 179   4              RETURN H - 'A' + 10;
 180   4              END READHEX;

 181   3          READBYTE: PROCEDURE BYTE;
                      /* READ TWO HEX DIGITS */
 182   4              RETURN SHL(READHEX,4) OR READHEX;
 183   4              END READBYTE;

 184   3          READCS: PROCEDURE BYTE;
                      /* READ BYTE WHILE COMPUTING CHECKSUM */
 185   4              DECLARE B BYTE;
 186   4              CS = CS + (B := READBYTE);
 187   4              RETURN B;
 188   4              END READCS;

 189   3          MAKE$DOUBLE: PROCEDURE(H,L) ADDRESS;
                      /* CREATE A DOUBLE BYTE VALUE FROM TWO SINGLE BYTES */
 190   4              DECLARE (H,L) BYTE;
 191   4              RETURN SHL(DOUBLE(H),8) OR L;
 192   4              END MAKE$DOUBLE;


                  /* INITIALIZE */
 193   3          SA, FA, NB = 0;
 194   3          P = 0; /* PARAGRAPH COUNT */
 195   3          TA,L = TPA;  /* BASE ADDRESS OF TRANSIENT ROUTINES */
 196   3          SBUFF(0) = EOFILE;
```



```
                  /* READ RECORDS UNTIL :00XXXX IS ENCOUNTERED */
 197   3              DO FOREVER;
                      /* SCAN THE : */
 198   4                  DO WHILE GETCHAR <> ':';
 199   5                  END;

                      /* SET CHECK SUM TO ZERO, AND SAVE THE RECORD LENGTH */
 200   4              CS = 0;
                      /* MAY BE THE END OF TAPE */
 201   4              IF (RL := READCS) = 0 THEN
 202   4                      GO TO FIN;
 203   4              NB = NB + RL;

 204   4              TA, LA = MAKE$DOUBLE(READCS,READCS);
 205   4              IF SA = 0 THEN SA = LA;


                      /* READ THE RECORD TYPE (NOT CURRENTLY USED) */
 207   4              RT = READCS;

                      /* PROCESS EACH BYTE */
 208   4                  DO WHILE (RL := RL - 1) <> 255;
 209   5                  CALL SETMEM(READCS); LA = LA+1;
 211   5                  END;
 212   4              IF LA > FA THEN FA = LA - 1;

                      /* NOW READ CHECKSUM AND COMPARE */
 214   4              IF CS + READBYTE <> 0 THEN
 215   4                  DO; CALL PRINT(.('CHECK SUM ERROR $'));
 217   5                  CALL DIAGNOSE;
 218   5                  END;
 219   4              END;

 220   3          FIN:
                  /* EMPTY THE BUFFERS */
                  TA = LA;
 221   3              DO WHILE L < TA;
 222   4              CALL SETMEM(0); LA = LA+1;
 224   4              END;
                  /* PRINT FINAL STATISTICS */
 225   3          CALL PRINT(.('FIRST ADDRESS $')); CALL PRINTADDR(SA);
 227   3          CALL PRINT(.('LAST  ADDRESS $')); CALL PRINTADDR(FA);
 229   3          CALL PRINT(.('BYTES READ    $')); CALL PRINTADDR(NB);
 231   3          CALL PRINT(.('RECORDS WRITTEN $')); CALL PRINTHEX(P);
 233   3          CALL CRLF;

 234   3          END RELOC;

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

               /* SET UP STACKPOINTER IN THE LOCAL AREA */
 235   2      DECLARE STACK(16) ADDRESS;
 236   2      SP    STACKPOINTER; STACKPOINTER = .STACK(LENGTH(STACK));
 238   2      LA = TPA;
```

```
PL/M-80 COMPILER


 239   2       SBP   LENGTH(SBUFF);
                  /* SET UP THE SOURCE FILE */
 240   2          CALL MOVE(FCBA,.SFCB,33);
 241   2          CALL MOVE(.('HEX',0),.SFCB(9),4);
 242   2          CALL OPEN(.SFCB);
 243   2          IF DCNT = 255 THEN CALL PERROR(.('CANNOT OPEN SOURCE$'));

 245   2          CALL MOVE(.('COM'),FCBA+9,3);

              /* REMOVE ANY EXISTING FILE BY THIS NAME */
 246   2          CALL DELETE(FCBA);
              /* THEN OPEN A NEW FILE */
 247   2          CALL MAKE(FCBA); CALL OPEN(FCBA);
 249   2          IF DCNT = 255 THEN CALL PERROR(.('NO MORE DIRECTORY SPACE$')); EL
 251   2                DO; CALL RELOC;
 253   3                CALL CLOSE(FCBA);
 254   3                IF DCNT = 255 THEN CALL PERROR(.('CANNOT CLOSE FILE$'));
 256   3                END;
 257   2          CALL CRLF;

 258   2       CALL BOOT;
 259   2       END LOADCOM;
 260   1       END;
```



```
0000 LOAD#
0000 LOAD#
023B 13     023B 14     023F 15     0240 16     0240 17
02D0 22     02D4 24     02DF 25     02E0 26     02E0 27
02E5 28     02EA 29     02EB 30     02EF 32     02F8 33
0306 34     030F 35     0310 36     0314 38     0321 39
032A 40     032B 41     0331 43     0339 44     0341 45
0342 46     0348 48     0351 49     0352 50     0358 52
035B 53     0363 54     0364 56     036A 58     0370 59
0378 60     037E 61     0386 62     0389 63     038A 65
0390 67     039C 68     039D 69     03A3 71     03AF 72
03B0 73     03B6 75     03C2 76     03C3 77     03C3 78
03CE 79     03CF 80     03D5 82     03DE E3     03DF 84
03E5 86     03EF 87     03EF 88     03F5 90     03FF 91
03FF 92     0405 94     0411 95     0412 96     041E 98
0421 99     0422 100    0431 102    043D 103    0447 104
044E 105    0455 106    045E 107    0459 108    0459 110
0469 111    0472 112    0497 113    04A5 114    04BA 116
04C2 117    04C8 118    04D1 119    04D7 120    04D7 121
04DA 122    04E0 123    04E4 124    04E4 126    05FD 129
0601 131    060D 132    0613 133    0624 134    0632 135
064A 136    0651 137    0658 138    065C 139    0667 141
066D 142    066D 143    0670 144    067F 145    06E0 146
06E6 148    06E6 149    06E9 150    06F1 151    06F6 152
06FB 153    0600 154    060E 155    060E 156    0694 157
069C 158    06A2 159    06A5 160    06B1 161    06FC 162
06BF 163    06D0 164    06D7 165    06DC 166    06DF 167
06E2 168    06E5 169    06FC 170    06FC 172    0700 173
0711 174    071D 176    0723 177    0726 178    0726 179
072E 180    072E 181    072E 182    073D 183    073D 184
073D 186    0748 187    074C 188    074C 189    0752 191
0763 192    04E4 193    04F0 194    04F4 195    04FD 196
0502 197    0502 198    050A 199    050D 200    0512 201
051D 202    0520 203    052E 204    0541 205    054D 206
0553 207    0559 208    0565 209    056C 210    0573 211
0576 212    0582 213    0589 214    0595 216    059B 217
059E 218    059E 219    05A1 220    05A7 221    05B3 222
05B8 223    05BF 224    05C2 225    05C8 226    05D0 227
05D6 228    05DE 229    05E4 230    05EC 231    05F2 232
05F9 233    05FC 234    0240 236    0247 237    024B 238
0251 239    0257 240    0263 241    026F 242    0275 243
027D 244    0283 245    028F 246    0295 247    029B 248
02A1 249    02A9 250    02B2 252    02B5 253    02BB 254
02C3 255    02C9 256    02C9 257    02CC 258    02CF 259
0000 MODULE#

0
```

```
CP/M MACRO ASSEM 2.0     #002    CP/M VERSION 2.0 SYSTEM RELOCATOR - 8/79

0154 29                  DAD     H       ;*4
0155 29                  DAD     H       ;*8
0156 C1                  POP     B       ;*2 IN B,C
0157 09                  DAD     B       ;*10 IN H,L
0158 4F                  MOV     C,A
0159 0600                MVI     B,0
015B 09                  DAD     B       ;*10+X
015C C34001              JMP     CLOOP
                 ECON:   ;END OF CONVERSION, CHECK FOR PROPER RANGE
015F 7C                  MOV     A,H
0160 B7                  ORA     A
0161 C27201              JNZ     CERROR
0164 7D                  MOV     A,L
0165 FE10                CPI     16
0167 DA7201              JC      CERROR
016A 2E00                MVI     L,0
016C 67                  MOV     H,A
016D 29                  DAD     H       ;SHL 1
016E 29                  DAD     H       ;SHL 2 FOR KILOBYTES
                 ;       H,L HAVE TOP OF MEMORY+1
016F C3A501              JMP     SETASC
                 ;
                 CERROR:
0172 117B01              LXI     D,CONMSG
0175 CD7503              CALL    PRINT
0178 C30000              JMP     BOOT
017B 0D0A494E56CONMSG:   DB      CR,LF,'INVALID MEMORY SIZE$'
                 ;
                 ;
                 ;       FIND END OF MEMORY
                 FINDTOP:
0191 210000              LXI     H,0
0194 24          FINDM:  INR     H       ;TO NEXT PAGE
0195 CAA101              JZ      MSIZED  ;CAN OVERFLOW ON 64K SYSTEMS
0198 7E                  MOV     A,M
0199 2F                  CMA
019A 77                  MOV     M,A
019B BE                  CMP     M
019C 2F                  CMA
019D 77                  MOV     M,A     ;BITS INVERTED FOR RAM OPERATIONAL TEST
019E CA9401              JZ      FINDM
                 ;       BITS DIDN'T CHANGE, MUST BE END OF MEMORY
                 ;       ALIGN ON EVEN BOUNDARY
01A1 7C          MSIZED: MOV     A,H
01A2 E6FC                ANI     1111$1100B      ;EVEN 1K BOUNDARY
01A4 67                  MOV     H,A
                 SETASC: ;SET ASCII VALUE OF MEMORY SIZE
01A5 E5                  PUSH    H       ;SAVE FOR LATER
                 ;       **** SERIALIZATION ****
01A6 2A0600              LHLD    BDOS+1
01A9 227A03              SHLD    SER1
                 ;       **** SERIALIZATION ****
01AC E1                  POP     H
01AD E5                  PUSH    H
01AE 7C                  MOV     A,H
01AF 0F                  RRC
```



```
CP/M MACRO ASSEM 2.0     #001    CP/M VERSION 2.0 SYSTEM RELOCATOR - 8/79

                         TITLE   'CP/M VERSION 2.0 SYSTEM RELOCATOR - 8/79'
                 ;       CPM RELOCATOR PROGRAM, INCLUDED WITH THE MODULE TO PERFO
                 ;       THE MOVE FROM 900H TO THE DESTINATION ADDRESS
                 ;
                 ;       COPYRIGHT (C) 1979
                 ;       DIGITAL RESEARCH
                 ;       BOX 579, PACIFIC GROVE CALIFORNIA
                 ;       93950
                 ;
0100                     ORG     100H
0100 C32C01              JMP     PASTCOPY
0103 434F505952COPY:     DB      'COPYRIGHT (C) DIGITAL RESEARCH, 1979'
                 PASTCOPY:
0003 =           BIOSWK  EQU     03H     ;THREE PAGES FOR BIOS WORKSPACE
0800 =           STACK   EQU     800H
0801 =           MODSIZ  EQU     801H    ;MODULE SIZE IS STORED HERE
0014 =           VERSION EQU     20      ;CPM VERSION NUMBER
0100 =           BOOTSIZ EQU     100H    ;SIZE OF THE COLD START LOADER
                 ;       (MAY HAVE FIRST 80H BYTES = 00H)
0800 =           BDOSL   EQU     0800H   ;RELATIVE LOCATION OF BDOS
1600 =           BIOS    EQU     1600H   ;RELATIVE LOCATION OF BIOS
                 ;
0000 =           BOOT    EQU     0000H   ;REBOOT LOCATION
0005 =           BDOS    EQU     0005H
0009 =           PRNT    EQU     9       ;PRINT BUFFER FUNCTION
005C =           FCB     EQU     5CH     ;DEFAULT FCB
0900 =           MODULE  EQU     900H    ;MODULE ADDRESS
                 ;
000D =           CR      EQU     0DH
000A =           LF      EQU     0AH
012C 310008              LXI     SP,STACK
                 ;
                 ;       MAY BE MEMORY SIZE SPECIFIED IN COMMAND
012F 115D00              LXI     D,FCB+1
0132 1A                  LDAX    D
0133 FE20                CPI     ' '
0135 CA9101              JZ      FINDTOP
0138 FE3F                CPI     '?'     ;WAS * SPECIFIED?
013A CA9101              JZ      FINDTOP
                 ;
                 ;       MUST BE MEMORY SIZE SPECIFICATION
013D 210000              LXI     H,0
                 CLOOP:  ;CONVERT TO DECIMAL
0140 1A                  LDAX    D
0141 13                  INX     D
0142 FE20                CPI     ' '
0144 CA5F01              JZ      ECON
0147 B7                  ORA     A
0148 CA5F01              JZ      ECON
                 ;       MUST BE DECIMAL DIGIT
014B D630                SUI     '0'
014D FE0A                CPI     10
014F D27201              JNC     CERROR
                 ;       DECIMAL DIGIT IS IN A
0152 29                  DAD     H       ;*2
0153 E5                  PUSH    H
```

```
CP/M MACRO ASSEM 2.0     #003    CP/M VERSION 2.0 SYSTEM RELOCATOR - 8/79

01B0 0F                  RRC
01B1 E63F                ANI     11$1111B        ;FOR 1K COUNTS
01B3 C2B801              JNZ     NOT64           ;MAY BE 64 K MEM SIZE
01B6 3E40                MVI     A,64            ;SET TO LITERAL IF SO
01B8 47          NOT64:  MOV     B,A             ;READY FOR COUNT DOWN
01B9 218D03              LXI     H,AMEM
01BC 3E30                MVI     A,'0'
01BE 77                  MOV     M,A
01BF 23                  INX     H
01C0 77                  MOV     M,A     ;BOTH ARE SET TO ASCII 0
01C1 218F03      ASC0:   LXI     H,AMEM+1                ;ADDRESS OF ASCII EQUIVA
01C4 34                  INR     M
01C5 7E                  MOV     A,M
01C6 FE3A                CPI     '9'+1
01C8 DACF01              JC      ASC1
01CB 3630                MVI     M,'0'
01CD 2B                  DCX     H
01CE 34                  INR     M
01CF 05          ASC1:   DCR     B       ;COUNT DOWN BY KILOBYTES
01D0 C2C101              JNZ     ASC0
01D3 117E03              LXI     D,MEMSG
01D6 CD7503              CALL    PRINT   ;MEMORY SIZE MESSAGE
                 ;
01D9 210108              LXI     H,MODSIZ
01DC 4E                  MOV     C,M
01DD 23                  INX     H
01DE 46                  MOV     B,M     ;B,C CONTAINS MODULE SIZE
01DF C5                  PUSH    B       ;MODULE SIZE STACKED ON MEM SIZE
                 ;
                 ;       TRY TO FIND THE ASCII STRING 'K CP/M VER X.X' TO SET SIZ
01E0 210009              LXI     H,MODULE
                 ;       B,C CONTAINS MODULE LENGTH
                 SLOOP:  ;SEARCH LOOP
01E3 118F03              LXI     D,AMSG
01E6 78                  MOV     A,B
01E7 B1                  ORA     C
01E8 CA2C02              JZ      ESEAR   ;END OF SEARCH
01EB 0B                  DCX     B       ;COUNT SEARCH LENGTH DOWN
01EC C5                  PUSH    B
01ED 0E0F                MVI     C,LAMSG ;LENGTH OF SEARCH MESSAGE
01EF E5                  PUSH    H       ;SAVE BASE ADDRESS OF SEARCH
                 CHLOOP: ;CHARACTER LOOP, MATCH ON CONTENTS OF D,E AND H,L
01F0 1A                  LDAX    D
01F1 BE                  CMP     M
01F2 C21A02              JNZ     NOMATCH
01F5 13                  INX     D       ;TO NEXT SEARCH CHARACTER
01F6 23                  INX     H       ;TO NEXT MATCH CHARACTER
01F7 0D                  DCR     C       ;COUNT LENGTH DOWN
01F8 CA2002              JZ      FSEAR   ;FOUND SEARCH STRING
01FB C3F001              JMP     CHLOOP
                 ;
                 ;       **** SERIALIZATION ****
01FE 01                  DB      LXI     ;CONFUSE DISASSEMBLER
                 BADSER: ;BAD SERIAL NUMBER, LOOP TO CONFUSE ICE-80
01FF AF                  XRA     A
                 BADSER0:
```

```
CP/M MACRO ASSEM 2.0     #004    CP/M VERSION 2.0 SYSTEM RELOCATOR - 8/79

0200 3D                  DCR     A
0201 C20002              JNZ     BADSER0
                 ;
0204 21F376              LXI     H,DI OR (HLT SHL 8)
0207 227A03              SHLD    PRHLT
020A 217703              LXI     H,PRJMP
020D 36CD                MVI     M,CALL  ;CHANGE JMP BDOS TO CALL
020F 118802              LXI     D,SYNCMSG-5
0212 210500              LXI     H,5
0215 19                  DAD     D       ;TO CONFUSE SEARCHES ON ADDRESSES
0216 EB                  XCHG
0217 C37503              JMP     PRINT
                 ;       **** SERIALIZATION ****
                 ;
                 NOMATCH:
                         ;NOT FOUND AT THIS ADDRESS, LOOK AT NEXT ADDRESS
021A E1                  POP     H
021B 23                  INX     H
021C C1                  POP     B       ;RECALL MODULE LENGTH
021D C3E301              JMP     SLOOP
                 ;
                 FSEAR:
                         ;FOUND STRING, SET MEMORY SIZE
0220 E1                  POP     H       ;START ADDRESS OF STRING BEING MATCHED
0221 C1                  POP     B       ;CLEAR B,C WHICH WAS STACKED
0222 2B                  DCX     H
0223 118F03              LXI     D,AMEM+1
0226 1A                  LDAX    D
0227 77                  MOV     M,A
0228 2B                  DCX     H
0229 1B                  DCX     D
022A 1A                  LDAX    D
022B 77                  MOV     M,A
                 ;       END OF FILL
                 ;
                 ESEAR:  ;END OF SEARCH
                 ;       **** SERIALIZATION ****
                 ;       CHECK FOR LEAST SIGNIFICANT BYTE OF 06 IN SER1
022C 017A03              LXI     B,SER1
022F 0A                  LDAX    B
0230 FE06                CPI     6
0232 3E00                MVI     A,0
0234 C25A02              JNZ     SETJMP  ;BAD SERIALIZATION IF NOT 06
0237 02                  STAX    B       ;STORE 00 TO LEAST SIGNIFICANT BYTE
                 ;       **** SERIALIZATION ****
0238 C1                  POP     B       ;RECOVER MODULE LENGTH
0239 E1                  POP     H       ;H,L CONTAINS END OF MEMORY
023A C5                  PUSH    B       ;SAVE LENGTH FOR RELOCATION BELOW
023B 78                  MOV     A,B
023C C603                ADI     BIOSWK  ;ADD BIOS WORK SPACE TO MODULE LENGTH
023E 47                  MOV     B,A
023F 7D                  MOV     A,L
0240 91                  SUB     C       ;COMPUTE MEMTOP-MODULE SIZE
0241 6F                  MOV     L,A
0242 7C                  MOV     A,H
0243 98                  SBB     B
```

```
0244 67                 MOV     H,A
                ;       H,L CONTAINS THE BASE OF THE RELOCATION AREA
0245 227C03             SHLD    RELBAS  ;SAVE THE RELOCATION BASE
0248 EB                 XCHG            ;MODULE BASE TO D,E
0249 210009             LXI     H,MODULE;READY FOR THE MOVE
024C C1                 POP     B       ;RECOVER ACTUAL MODULE LENGTH
024D C5                 PUSH    B       ;SAVE FOR RELOCATION
024E 3A6D00             LDA     FCB+17  ;CHECK FOR NO MOVE CONDITION
0251 FE20               CPI     ' '
0253 CA6302             JZ      MOVE
                ;       SECOND PARAMETER SPECIFIED, LEAVE THE DATA AT 'MODULE'
0256 09                 DAD     B       ;MOVE H,L TO BIT MAP POSITION
0257 C37002             JMP     RELOC
                ;
                ;       **** SERIALIZATION ****
025A 21FF01     SETJMP: LXI     H,BADSER        ;BAD SERIALIZATION
025D 220703             SHLD    JMPSER+1        ;FILL JUMP INSTRUCTION
0260 C30603             JMP     JMPSER          ;EVENTUAL JUMP TO MESSAGE
                ;       **** SERIALIZATION ****
                ;
                MOVE:
0263 78                 MOV     A,B     ;BC=0?
0264 B1                 ORA     C
0265 CA7002             JZ      RELOC
0268 0B                 DCX     B       ;COUNT MODULE SIZE DOWN TO ZERO
0269 7E                 MOV     A,M     ;GET NEXT ABSOLUTE LOCATION
026A 12                 STAX    D       ;PLACE IT INTO THE RELOC AREA
026B 13                 INX     D
026C 23                 INX     H
026D C36302             JMP     MOVE
                ;
                RELOC:  ;STORAGE MOVED, READY FOR RELOCATION
                ;       HL ADDRESSES BEGINNING OF THE BIT MAP FOR RELOCATION
0270 C1                 POP     B       ;RECALL MODULE LENGTH
0271 E5                 PUSH    H       ;SAVE BIT MAP BASE IN STACK
0272 2A7C03             LHLD    RELBAS
0275 EB                 XCHG
0276 210001             LXI     H,BOOTSIZ
0279 19                 DAD     D       ;TO FIND BIAS VALUE
                ;       REGISTER H CONTAINS BIAS VALUE
                ;
                ;       RELOCATE AT 'MODULE' IF SECOND PARAMETER GIVEN
027A 3A6D00             LDA     FCB+17
027D FE20               CPI     ' '
027F CA8502             JZ      REL0
                ;
                ;       IMAGE NOT MOVED, ADJUST VALUES AT 'MODULE'
0282 110009             LXI     D,MODULE
0285 78         REL0:   MOV     A,B     ;BC=0?
0286 B1                 ORA     C
0287 CAC002             JZ      ENDREL
                ;       **** SERIALIZATION ****
028A C3A402             JMP     PASTSYNC
                SYNCMSG:
028D 0D0A53594E         DB      CR,LF,'SYNCRONIZATION ERROR$'
                PASTSYNC:
```



```
                ;       **** SERIALIZATION ****
                ;
                ;       NOT END OF THE RELOCATION, MAY BE INTO NEXT BYTE OF BIT
02A4 0B                 DCX     B       ;COUNT LENGTH DOWN
02A5 7B                 MOV     A,E
02A6 E607               ANI     111B    ;0 CAUSES FETCH OF NEXT BYTE
02A8 C2B002             JNZ     REL1
                ;       FETCH BIT MAP FROM STACKED ADDRESS
02AB E3                 XTHL
02AC 7E                 MOV     A,M     ;NEXT 8 BITS OF MAP
02AD 23                 INX     H
02AE E3                 XTHL            ;BASE ADDRESS GOES BACK TO STACK
02AF 6F                 MOV     L,A     ;L HOLDS THE MAP AS WE PROCESS 8 LOCATIO
02B0 7D         REL1:   MOV     A,L
02B1 17                 RAL             ;CY SET TO 1 IF RELOCATION NECESSARY
02B2 6F                 MOV     L,A     ;BACK TO L FOR NEXT TIME AROUND
02B3 D2BC02             JNC     REL2    ;SKIP RELOCATION IF CY=0
                ;
                ;       CURRENT ADDRESS REQUIRES RELOCATION
02B6 1A                 LDAX    D
02B7 84                 ADD     H       ;APPLY BIAS IN H
02B8 12                 STAX    D
02B9 C3BC02             JMP     REL2
                ;
02BC 13         REL2:   INX     D       ;TO NEXT ADDRESS
02BD C38502             JMP     REL0    ;FOR ANOTHER BYTE TO RELOCATE
                ;
                ENDREL: ;END OF RELOCATION
02C0 D1                 POP     D       ;CLEAR STACKED ADDRESS
                ;       **** SERIALIZATION ****
02C1 110012             LXI     D,MODULE+BDOSL+BOOTSIZ  ;ADDRESSING NEW SERIAL N
02C4 2A7A03             LHLD    SER1                    ;ADDRESSING HOST SERIAL
02C7 0E06               MVI     C,6                     ;LENGTH OF SERIAL NUMBER
02C9 1A         CHKSER: LDAX    D
02CA BE                 CMP     M
02CB C25A02             JNZ     SETJMP
02CE 23                 INX     H
02CF 13                 INX     D
02D0 0D                 DCR     C
02D1 C2C902             JNZ     CHKSER
                ;       **** SERIALIZATION ****
                ;
02D4 3A6D00             LDA     FCB+17
02D7 FE20               CPI     ' '
02D9 CA6D03             JZ      TRANSFER
                ;       DON'T GO TO THE LOADED PROGRAM, LEAVE IN MEMORY
                ;*      MAY HAVE TO MOVE THE PROGRAM IMAGE DOWN 1/2 PAGE
02DC 0680               MVI     B,128   ;CHECK FOR 128 ZEROES
02DE 210009             LXI     H,MODULE
02E1 7E         TR0:    MOV     A,M
02E2 B7                 ORA     A
02E3 C20903             JNZ     TREND
02E6 23                 INX     H
02E7 05                 DCR     B
02E8 C2E102             JNZ     TR0
                ;
```

```
                ;       ALL ZERO FIRST 1/2 PAGE, MOVE DOWN 80H BYTES
02EB EB                 XCHG            ;NEXT TO GET IN D,E
02EC 2A0108             LHLD    MODSIZ
02EF 0180FF             LXI     B,-128
02F2 09                 DAD     B       ;NUMBER OF BYTES TO MOVE IN H,L
02F3 44                 MOV     B,H
02F4 4D                 MOV     C,L     ;TRANSFERRED TO B,C
02F5 210009             LXI     H,MODULE;DESTINATION IN H,L
02F8 78         TRMOV:  MOV     A,B
02F9 B1                 ORA     C       ;ALL MOVED?
02FA CA0903             JZ      TREND
02FD 0B                 DCX     B
02FE 1A                 LDAX    D
02FF 77                 MOV     M,A     ;ONE BYTE TRANSFERRED
0300 13                 INX     D
0301 23                 INX     H
0302 C3F802             JMP     TRMOV
                ;
                ;
                ;       **** SERIALIZATION ****
0305 01                 DB      LXI
0306 C30603     JMPSER: JMP     JMPSER  ;ADDRESS FIELD FILLED-IN
                ;       **** SERIALIZATION ****
                ;
                TREND:  ;SET ASCII MEMORY IMAGE SIZE
0309 210108             LXI     H,MODSIZ
030C 4E                 MOV     C,M
030D 23                 INX     H
030E 46                 MOV     B,M
030F 210009             LXI     H,MODULE;B,C MODULE SIZE, H,L BASE
0312 09                 DAD     B
0313 44                 MOV     B,H     ;B CONTAINS NUMBER OF PAGES TO SAVE+1
0314 215F03             LXI     H,SAVMEM;ASCII MEMORY SIZE
0317 3E30               MVI     A,'0'
0319 77                 MOV     M,A
031A 23                 INX     H
031B 77                 MOV     M,A
                ;       '00' STORED INTO MESSAGE
                TRCOMP:
031C 05                 DCR     B
031D CA3103             JZ      TRC1
0320 216003             LXI     H,SAVMEM+1      ;ADDRESSING LEAST DIGIT
0323 34                 INR     M
0324 7E                 MOV     A,M
0325 FE3A               CPI     '9'+1
0327 DA1C03             JC      TRCOMP
032A 3630               MVI     M,'0'
032C 2B                 DCX     H
032D 34                 INR     M
032E C31C03             JMP     TRCOMP
                ;       FILL CPMXX.COM FROM SAVMEM
0331 2A8D03     TRC1:   LHLD    AMEM
0334 226503             SHLD    SAVM0
                ;       MESSAGE SET, PRINT IT AND REBOOT
0337 114003             LXI     D,RELOK
033A CD7503             CALL    PRINT
```



```
033D C30000             JMP     BOOT
0340 0D0A524541RELOK:   DB      CR,LF,'READY FOR "SYSGEN" OR'
0357 0D0A225341         DB      CR,LF,'"SAVE '
035F 30302043504SAVMEM: DB      '00 CPM'
0365 30302E434FSAVM0:   DB      '00.COM"$'
                ;
                TRANSFER:
                ;       GO TO THE RELOCATED MEMORY IMAGE
036D 110017             LXI     D,BOOTSIZ+BIOS  ;MODULE
0370 2A7C03             LHLD    RELBAS  ;RECALL BASE OF RELOC AREA
0373 19                 DAD     D       ;INDEX TO 'BOOT' ENTRY POINT
0374 E9                 PCHL            ;GO TO RELOCATED PROGRAM
                ;
                ;       **** SERIALIZATION ****
                PRINT:
0375 0E09               MVI     C,PRNT
0377 C30500     PRJMP:  JMP     BDOS
                PRHLT:
                ;
                ;       DATA    AREAS
037A            SER1:   DS      2       ;SERIAL NUMBER ADDRESS FOR HOST
037C            RELBAS: DS      2       ;RELOCATION BASE
037E 0D0A434F4EMEMSG:   DB      CR,LF,'CONSTRUCTING '
038D 3030       AMEM:   DB      '00'
038F 6B2043502FAMSG:    DB      'k CP/M vers '
039B 322E30             DB      VERSION/10+'0','.',VERSION MOD 10 +'0'
000F =          LAMSG   EQU     $-AMSG  ;LENGTH OF MESSAGE
039E 24                 DB      '$'             ;TERMINATOR FOR MESSAGE
039F                    END
```

```
038D AMEM       038F AMSG       01C1 ASC0       01CF ASC1       01FF BADSER
0200 BADSER0    0800 BDOSL      0005 BDOS       1600 BIOS       0003 BIOSWK
0000 BOOT       0100 BOOTSIZ    0172 CERROR     02C9 CHKSER     01F0 CHLOOP
0140 CLOOP      017B CONMSG     0103 COPY       000D CR         015F ECON
02C0 ENDREL     022C ESEAR      005C FCB        0194 FINDM      0191 FINDTOP
0220 FSEAR      0306 JMPSER     000F LAMSG      000A LF         037E MEMSG
0801 MODSIZ     0900 MODULE     0263 MOVE       01A1 MSIZED     021A NOMATCH
01B8 NOT64      012C PASTCOPY   02A4 PASTSYNC   037A PRHLT      0375 PRINT
0377 PRJMP      0009 PRNT       0285 RELO       02B0 REL1       02BC REL2
037C RELBAS     0270 RELOC      0340 RELOK      0365 SAVM0      035F SAVMEM
037A SER1       01A5 SETASC     025A SETJMP     01E3 SLOOP      0800 STACK
028D SYNCMSG    02E1 TR0        036D TRANSFER   0331 TRC1       031C TRCOMP
0309 TREND      02F8 TRMOV      0014 VERSION
```

```
                        TITLE   'SYSGEN - SYSTEM GENERATION PROGRAM 8/79'
                ;       SYSTEM GENERATION PROGRAM, VERSION FOR MDS
0014 =          VERS    EQU     20      ;X.X
                ;
                ;       COPYRIGHT (C) DIGITAL RESEARCH
                ;                1976, 1977, 1978, 1979
                ;
001A =          NSECTS  EQU     26      ;NO. OF SECTORS PER TRACK
0002 =          NTRKS   EQU     2       ;NO. OF OPERATING SYSTEM TRACKS
0004 =          NDISKS  EQU     4       ;NUMBER OF DISK DRIVES
0080 =          SECSIZ  EQU     128     ;SIZE OF EACH SECTOR
0007 =          LOG2SEC EQU     7       ;LOG 2 SECSIZ
0001 =          SKEW    EQU     1       ;SECTOR SKEW FACTOR
                ;
005C =          FCB     EQU     005CH   ;DEFAULT FCB LOCATION
007C =          FCBCR   EQU     FCB+32  ;CURRENT RECORD LOCATION
0100 =          TPA     EQU     0100H   ;TRANSIENT PROGRAM AREA
0900 =          LOADP   EQU     900H    ;LOAD POINT FOR SYSTEM DURING LOAD/STORE
0005 =          BDOS    EQU     5H      ;DOS ENTRY POINT
0000 =          BOOT    EQU     0       ;JMP TO 'BOOT' TO REBOOT SYSTEM
0001 =          CONI    EQU     1       ;CONSOLE INPUT FUNCTION
0002 =          CONO    EQU     2       ;CONSOLE OUTPUT FUNCTION
000F =          SELF    EQU     14      ;SELECT DISK
000F =          OPENF   EQU     15      ;DISK OPEN FUNCTION
0014 =          DREADF  EQU     20      ;DISK READ FUNCTION
                ;
000A =          MAXTRY  EQU     10      ;MAXIMUM NUMBER OF RETRIES ON EACH READ/
000D =          CR      EQU     0DH     ;CARRIAGE RETURN
000A =          LF      EQU     0AH     ;LINE FEED
0010 =          STACKSIZE       EQU     16      ;SIZE OF LOCAL STACK
                ;
0001 =          WBOOT   EQU     1       ;ADDRESS OF WARM BOOT (OTHER PATCH ENTRY
                ;                       POINTS ARE COMPUTED RELATIVE TO WBOOT)
0018 =          SELDSK  EQU     24      ;WBOOT+24 FOR DISK SELECT
001B =          SETTRK  EQU     27      ;WBOOT+27 FOR SET TRACK FUNCTION
001E =          SETSEC  EQU     30      ;WBOOT+30 FOR SET SECTOR FUNCTION
0021 =          SETDMA  EQU     33      ;WBOOT+33 FOR SET DMA ADDRESS
0024 =          READF   EQU     36      ;WBOOT+36 FOR READ FUNCTION
0027 =          WRITF   EQU     39      ;WBOOT+39 FOR WRITE FUNCTION
                ;
0100                    ORG     TPA     ;TRANSIENT PROGRAM AREA
0100 C37802             JMP     START
0103 4341505952         DB      'COPYRIGHT (C) 1978, DIGITAL RESEARCH '
                ;
                ;       TRANSLATE TABLE - SECTOR NUMBERS ARE TRANSLATED
                ;       HERE TO DECREASE THE SYSGEN TIME FOR MISSED SECTORS
                ;       WHEN SLOW CONTROLLERS ARE INVOLVED.  TRANSLATION TAKES
                ;       PLACE ACCORDING TO THE "SKEW" FACTOR SET ABOVE.
                ;
0128 02         OST:    DB      NTRKS   ;OPERATING SYSTEM TRACKS
0129 1A         SPT:    DB      NSECTS  ;SECTORS PER TRACK (CAN BE PATCHED)
                TRAN:                   ;BASE OF TRANSLATE TABLE
0001 #          TRELT   SET     1       ;FIRST/NEXT TRAN ELEMENT
0001 #          TRBASE  SET     1       ;BASE FOR WRAPAROUND
                        REPT    NSECTS  ;ONCE FOR EACH SECTOR ON A TRACK
                        DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
```



```
                TRELT   SET     TRELT+SKEW
                        IF      TRELT GT NSECTS
                TRBASE  SET     TRBASE+1
                TRELT   SET     TRBASE
                        ENDIF
                        ENDM
012A+01                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
012B+02                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
012C+03                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
012D+04                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
012E+05                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
012F+06                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
0130+07                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
0131+08                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
0132+09                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
0133+0A                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
0134+0B                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
0135+0C                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
0136+0D                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
0137+0E                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
0138+0F                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
0139+10                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
013A+11                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
013B+12                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
013C+13                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
013D+14                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
013E+15                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
013F+16                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
0140+17                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
0141+18                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
0142+19                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
0143+1A                 DB      TRELT   ;GENERATE FIRST/NEXT SECTOR
                ;
                ;       NOW LEAVE SPACE FOR EXTENSIONS TO TRANSLATE TABLE
                        IF      NSECTS LT 64
                        REPT    64-NSECTS
                        DB      0
                        ENDM
0144+00                 DB      0
0145+00                 DB      0
0146+00                 DB      0
0147+00                 DB      0
0148+00                 DB      0
0149+00                 DB      0
014A+00                 DB      0
014B+00                 DB      0
014C+00                 DB      0
014D+00                 DB      0
014E+00                 DB      0
014F+00                 DB      0
0150+00                 DB      0
0151+00                 DB      0
0152+00                 DB      0
0153+00                 DB      0
0154+00                 DB      0
0155+00                 DB      0
```

```
CP/M MACRO ASSEM 2.0     #003     SYSGEN - SYSTEM GENERATION PROGRAM 8/79

0156+00                  DB       0
0157+00                  DB       0
0158+00                  DB       0
0159+00                  DB       0
015A+00                  DB       0
015B+00                  DB       0
015C+00                  DB       0
015D+00                  DB       0
015E+00                  DB       0
015F+00                  DB       0
0160+00                  DB       0
0161+00                  DB       0
0162+00                  DB       0
0163+00                  DB       0
0164+00                  DB       0
0165+00                  DB       0
0166+00                  DB       0
0167+00                  DB       0
0168+00                  DB       0
0169+00                  DB       0
                ;
                ;
                ;
                ;
                ;        UTILITY SUBROUTINES
                MULTSEC:
                         ;MULTIPLY THE SECTOR NUMBER IN A BY THE SECTOR SIZE
016A 6F2600              MOV L,A! MVI H,0 ;SECTOR NUMBER IN HL
                         REPT LOG2SEC     ;LOG 2 OF SECTOR SIZE
                         DAD      H
                         ENDM
016D+29                  DAD      H
016E+29                  DAD      H
016F+29                  DAD      H
0170+29                  DAD      H
0171+29                  DAD      H
0172+29                  DAD      H
0173+29                  DAD      H
0174 C9                  RET ;WITH HL = SECTOR * SECTOR SIZE
                ;
                GETCHAR:
                ;        READ CONSOLE CHARACTER TO REGISTER A
0175 0E01CD0500          MVI C,CONI! CALL BDOS!
                ;        CONVERT TO UPPER CASE BEFORE RETURN
017A FE61D8              CPI 'A' OR 20H ! RC      ;RETURN IF BELOW LOWER CASE A
017D FE7B                CPI ('Z' OR 20H) + 1
017F D0                  RNC      ;RETURN IF ABOVE LOWER CASE Z
0180 E65FC9              ANI 5FH! RET
                ;
                PUTCHAR:
                ;        WRITE CHARACTER FROM A TO CONSOLE
0183 5F0E02CD05          MOV E,A! MVI C,CONO! CALL BDOS! RET
                ;
                CRLF:    ;SEND CARRIAGE RETURN, LINE FEED
018A 3E0D                MVI      A,CR
018C CD8301              CALL     PUTCHAR
```

```
CP/M MACRO ASSEM 2.0     #004     SYSGEN - SYSTEM GENERATION PROGRAM 8/79

018F 3E0A                MVI      A,LF
0191 CD8301              CALL     PUTCHAR
0194 C9                  RET
                ;
                CRMSG:   ;PRINT MESSAGE ADDRESSED BY H,L TIL ZERO
                         ;WITH LEADING CRLF
0195 E5CD8A01E1          PUSH H! CALL CRLF! POP H ;DROP THRU TO OUTMSG0
                OUTMSG:
019A 7EB7C8              MOV A,M! ORA A! RZ
                ;        MESSAGE NOT YET COMPLETED
019D E5CD8301E1          PUSH H! CALL PUTCHAR! POP H! INX H
01A3 C39A01              JMP      OUTMSG
                ;
                SEL:
                ;        SELECT DISK GIVEN BY REGISTER A
01A6 4F2A010011          MOV C,A! LHLD WBOOT! LXI D,SELDSK! DAD D! PCHL
                ;
                TRK:     ;SET UP TRACK
01AF 2A0100              LHLD     WBOOT   ;ADDRESS OF BOOT ENTRY
01B2 111B00              LXI      D,SETTRK        ;OFFSET FOR SETTRK ENTRY
01B5 19                  DAD      D
01B6 E9                  PCHL             ;GONE TO SETTRK
                ;
                SEC:     ;SET UP SECTOR NUMBER
01B7 2A0100              LHLD     WBOOT
01BA 111E00              LXI      D,SETSEC
01BD 19                  DAD      D
01BE E9                  PCHL
                ;
                DMA:     ;SET DMA ADDRESS TO VALUE OF B,C
01BF 2A0100              LHLD     WBOOT
01C2 112100              LXI      D,SETDMA
01C5 19                  DAD      D
01C6 E9                  PCHL
                ;
                READ:    ;PERFORM READ OPERATION
01C7 2A0100              LHLD     WBOOT
01CA 112400              LXI      D,READF
01CD 19                  DAD      D
01CE E9                  PCHL
                ;
                WRITE:   ;PERFORM WRITE OPERATON
01CF 2A0100              LHLD     WBOOT
01D2 112700              LXI      D,WRITF
01D5 19                  DAD      D
01D6 E9                  PCHL
                ;
                DREAD:   ;DISK READ FUNCTION
01D7 0E14                MVI      C,DREADF
01D9 C30500              JMP      BDOS
                ;
                OPEN:    ;FILE OPEN FUNCTION
01DC 0E0FC30500          MVI C,OPENF ! JMP BDOS
                ;
                GETPUT:
                ;        GET OR PUT CP/M (RW=0 FOR READ, 1 FOR WRITE)
```

```
                ;       DISK IS ALREADY SELECTED
                ;
01E1 210009             LXI     H,LOADP ;LOAD POINT IN RAM FOR CP/M DURING SYSGE
01E4 22AA04             SHLD    DMADDR
                ;
                ;       CLEAR TRACK TO 00
01E7 3EFF               MVI     A,-1    ;START WITH TRACK EQUAL -1
01E9 32A704             STA     TRACK
                ;
                RWTRK:  ;READ OR WRITE NEXT TRACK
01EC 21A704             LXI     H,TRACK
01EF 34                 INR     M       ;TRACK = TRACK + 1
01F0 3A2801             LDA     OST     ;NUMBER OF OPERATING SYSTEM TRACKS
01F3 BE                 CMP     M       ;= TRACK NUMBER ?
01F4 CA7702             JZ      ENDRW   ;END OF READ OR WRITE
                ;
                ;       OTHERWISE NOTDONE, GO TO NEXT TRACK
01F7 4E                 MOV     C,M     ;TRACK NUMBER
01F8 CDAF01             CALL    TRK     ;TO SET TRACK
01FB 3EFF               MVI     A,-1    ;COUNTS 0, 1, 2, . . . 25
01FD 32A804             STA     SECTOR  ;SECTOR INCREMENTED BEFORE READ OR WRITE
                ;
                RWSEC:  ;READ OR WRITE SECTOR
0200 3A2901             LDA     SPT     ;SECTORS PER TRACK
0203 21A804             LXI     H,SECTOR
0206 34                 INR     M       ;TO NEXT SECTOR
0207 BE                 CMP     M       ;A=26 AND M=0 1 2...25 (USUALLY)
0208 CA6602             JZ      ENDTRK  ;
                ;
                ;       READ OR WRITE SECTOR TO OR FROM CURRENT DMA ADDR
020B 21A804             LXI     H,SECTOR
020E 5E                 MOV     E,M     ;SECTOR NUMBER
020F 1600               MVI     D,0     ;TO DE
0211 212A01             LXI     H,TRAN
0214 46                 MOV     B,M     ;TRAN(0) IN B
0215 19                 DAD     D       ;SECTOR TRANSLATED
0216 4E                 MOV     C,M     ;VALUE TO C READY FOR SELECT
0217 C5                 PUSH    B       ;SAVE TRAN(0),TRAN(SECTOR)
0218 CDB701             CALL    SEC     ;SET UP SECTOR NUMBER
021B C1                 POP     B       ;RECALL TRAN(0),TRAN(SECTOR)
021C 79                 MOV     A,C     ;TRAN(SECTOR)
021D 90                 SUB     B       ;-TRAN(0)
021E CD6A01             CALL    MULTSEC ;*SECTOR SIZE
0221 EB                 XCHG            ;TO DE
0222 2AAA04             LHLD    DMADDR  ;BASE DMA ADDRESS FOR THIS TRACK
0225 19                 DAD     D       ;+(TRAN(SECTOR)-TRAN(0))*SECSIZ
022C 44                 MOV     B,H
0227 4D                 MOV     C,L     ;TO BC FOR SEC CALL
0228 CDBF01             CALL    DMA     ;DMA ADDRESS SET FROM B,C
                ;       DMA ADDRESS SET, CLEAR RETRY COUNT
022B AF                 XRA     A
022C 32AC04             STA     RETRY   ;SET TO ZERO RETRIES
                ;
                TRYSEC: ;TRY TO READ OR WRITE CURRENT SECTOR
022F 3AAC04             LDA     RETRY
0232 FE0A               CPI     MAXTRY  ;TOO MANY RETRIES?
```



```
0234 DA4B02             JC      TRYOK
                ;
                ;       PAST MAXTRIES, MESSAGE AND IGNORE
0237 211804             LXI     H,ERRMSG
023A CD9A01             CALL    OUTMSG
023D CD7501             CALL    GETCHAR
0240 FE0D               CPI     CR
0242 C26303             JNZ     REBOOT
                ;
                ;       TYPED A CR, OK TO IGNORE
0245 CD8A01             CALL    CRLF
0248 C30002             JMP     RWSEC
                ;
                TRYOK:
                ;       OK TO TRY READ OR WRITE
024B 3C                 INR     A
024C 32AC04             STA     RETRY   ;RETRY=RETRY+1
024F 3AA904             LDA     RW      ;READ OR WRITE?
0252 B7                 ORA     A
0253 CA5C02             JZ      TRYREAD
                ;
                ;       MUST BE WRITE
025E CDCF01             CALL    WRITE
0259 C35F02             JMP     CHKRW   ;CHECK FOR ERROR RETURNS
                TRYREAD:
025C CDC701             CALL    READ
                CHKRW:
025F B7                 ORA     A
0260 CA0002             JZ      RWSEC   ;ZERO FLAG IF R/W OK
                ;
                ;       ERROR, RETRY OPERATION
0263 C32F02             JMP     TRYSEC
                ;
                ;       END OF TRACK
                ENDTRK:
0266 3A2901             LDA     SPT     ;SECTORS PER TRACK
0269 CD6A01             CALL    MULTSEC ;*SECSIZ
026C EB                 XCHG            ;TO DE
026D 2AAA04             LHLD    DMADDR  ;BASE DMA FOR THIS TRACK
0270 19                 DAD     D       ;+SPT*SECSIZ
0271 22AA04             SHLD    DMADDR  ;READY FOR NEXT TRACK
0274 C3EC01             JMP     RWTRK   ;FOR ANOTHER TRACK
                ;
                ENDRW:  ;END OF READ OR WRITE, RETURN TO CALLER
0277 C9                 RET
                ;
                ;
                START:
                ;
0278 31CD04             LXI     SP,STACK        ;SET LOCAL STACK POINTER
027B 217503             LXI     H,SIGNON
027E CD9A01             CALL    OUTMSG
                ;
                ;       CHECK FOR DEFAULT FILE LOAD INSTEAD OF GET
                ;
0281 3A5D00             LDA     FCB+1   ;BLANK IF NO FILE
```

CP/M 2.0
COPYRIGHT © 1978
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA. 93950
SER. # _____

```
CP/M MACRO ASSEM 2.0     #007    SYSGEN - SYSTEM GENERATION PROGRAM 8/79

02B4 FE20                CPI     ' '
02B6 CAD602              JZ      GETSYS  ;SKIP TO GET SYSTEM MESSAGE IF BLANK
02B9 115C00              LXI     D,FCB   ;TRY TO OPEN IT
02BC CDDC01              CALL    OPEN    ;
02BF 3C                  INR     A       ;255 BECOMES 00
02C0 C29C02              JNZ     RDOK    ;OK TO READ IF NOT 255
                         ;
                         ;       FILE NOT PRESENT, ERROR AND REBOOT
                         ;
0293 217804              LXI     H,NOFILE
0296 CD9501              CALL    CRMSG
0299 C36303              JMP     REBOOT
                         ;
                         ;       FILE PRESENT
                         ;         READ TO LOAD POINT
                         ;
                         RDOK:
029C AF                  XRA     A
029D 327C00              STA     FCBCR   ;CURRENT RECORD = 0
                         ;
                         ;       PRE-READ AREA FROM TPA TO LOADP
                         ;
02A0 0E10                MVI     C,(LOADP-TPA)/SECSIZ
                         ;       PRE-READ FILE
                         PRERD:
02A2 C5                  PUSH    B       ;SAVE COUNT
02A3 115C00              LXI     D,FCB   ;INPUT FILE CONTROL COUNT
02A6 CDD701              CALL    DREAD   ;ASSUME SET TO DEFAULT BUFFER
02A9 C1                  POP     B       ;RESTORE COUNT
02AA B7                  ORA     A
02AB C2CD02              JNZ     BADRD   ;CANNOT ENCOUNTER END-OF FILE
02AE 0D                  DCR     C       ;COUNT DOWN
02AF C2A202              JNZ     PRERD   ;FOR ANOTHER SECTOR
                         ;
                         ;       SECTORS SKIPPED AT BEGINNING OF FILE
                         ;
02B2 210009              LXI     H,LOADP
                         RDINP:
02B5 E5                  PUSH    H
02B6 44                  MOV     B,H
02B7 4D                  MOV     C,L     ;READY FOR DMA
02B8 CDBF01              CALL    DMA     ;DMA ADDRESS SET
02BB 115C00              LXI     D,FCB   ;READY FOR READ
02BE CDD701              CALL    DREAD   ;
02C1 E1                  POP     H       ;RECALL DMA ADDRESS
02C2 B7                  ORA     A       ;00 IF READ OK
02C3 C21C03              JNZ     PUTSYS  ;ASSUME EOF IF NOT.
                         ;       MORE TO READ, CONTINUE
02C6 118000              LXI     D,SECSIZ
02C9 19                  DAD     D       ;HL IS NEW LOAD ADDRESS
02CA C3B502              JMP     RDINP
                         ;
                         BADRD:  ;EOF ENCOUNTERED IN INPUT FILE

02CD 218F04              LXI     H,BADFILE
02D0 CD9501              CALL    CRMSG
```

```
CP/M MACRO ASSEM 2.0     #008    SYSGEN - SYSTEM GENERATION PROGRAM 8/79

02D3 C36303              JMP     REBOOT
                         ;
                         ;
                         GETSYS:
02D6 218403              LXI     H,ASKGET        ;GET SYSTEM?
02D9 CD9501              CALL    CRMSG
02DC CD7501              CALL    GETCHAR
02DF FE0D                CPI     CR
02E1 CA1C03              JZ      PUTSYS  ;SKIP IF CR ONLY
                         ;
02E4 D641                SUI     'A'             ;NORMALIZE DRIVE NUMBER
02E6 FE04                CPI     NDISKS          ;VALID DRIVE?
02E8 DAF102              JC      GETC            ;SKIP TO GETC IF SO
                         ;
                         ;       INVALID DRIVE NUMBER
02EB CD6F03              CALL    BADDISK
02EE C3D602              JMP     GETSYS          ;TO TRY AGAIN
                         ;
                         GETC:
                         ;       SELECT DISK GIVEN BY REGISTER A
02F1 C641                ADI     'A'
02F3 32B403              STA     GDISK           ;TO SET MESSAGE
02F6 D641                SUI     'A'
02F8 CDA601              CALL    SEL             ;TO SELECT THE DRIVE
                         ;       GETSYS, SET RW TO READ AND GET THE SYSTEM
02FB CD0A01              CALL    CRLF
02FE 21AA03              LXI     H,GETMSG
0301 CD9A01              CALL    OUTMSG
0304 CD7501              CALL    GETCHAR
0307 FE0D                CPI     CR
0309 C26303              JNZ     REBOOT
030C CD0A01              CALL    CRLF
                         ;
030F AF                  XRA     A
0310 32A904              STA     RW
0313 CDE101              CALL    GETPUT
0316 213F04              LXI     H,DONE
0319 CD9A01              CALL    OUTMSG
                         ;
                         ;       PUT SYSTEM
                         PUTSYS:
031C 21C803              LXI     H,ASKPUT
031F CD9501              CALL    CRMSG
0322 CD7501              CALL    GETCHAR
0325 FE0D                CPI     CR
0327 CA6303              JZ      REBOOT
032A D641                SUI     'A'
032C FE04                CPI     NDISKS
032E DA3703              JC      PUTC
                         ;
                         ;       INVALID DRIVE NAME
0331 CD6F03              CALL    BADDISK
0334 C31C03              JMP     PUTSYS  ;TO TRY AGAIN
                         ;
                         PUTC:
                         ;       SET DISK FROM REGISTER C
```

```
CP/M MACRO ASSEM 2.0     #009     SYSGEN - SYSTEM GENERATION PROGRAM 8/79

 0337 C641               ADI      'A'
 0339 320404             STA      PDISK   ;MESSAGE SET
 033C D641               SUI      'A'
 033E CDA601             CALL     SEL     ;SELECT DEST DRIVE
                 ;       PUT SYSTEM, SET RW TO WRITE
 0341 21F503             LXI      H,PUTMSG
 0344 CD9501             CALL     CRMSG
 0347 CD7501             CALL     GETCHAR
 034A FE0D               CPI      CR
 034C C26303             JNZ      REBOOT
 034F CD8A01             CALL     CRLF
                 ;
 0352 21A904             LXI      H,RW
 0355 3601               MVI      M,1
 0357 CDE101             CALL     GETPUT  ;TO PUT SYSTEM BACK ON DISKETTE
 035A 213F04             LXI      H,DONE
 035D CD9A01             CALL     OUTMSG
 0360 C31C03             JMP      PUTSYS  ;FOR ANOTHER PUT OPERATION
                 ;
                 REBOOT:
 0363 3E00               MVI      A,0
 0365 CDA601             CALL     SEL
 0368 CD8A01             CALL     CRLF
 036B C30000             JMP      BOOT
                 BADDISK:
                         ;BAD DISK NAME
 036E 215104             LXI      H,QDISK
 0371 CD9501             CALL     CRMSG
 0374 C9                 RET
                 ;
                 ;
                 ;
                 ;       DATA AREAS
                 ;       MESSAGES
 0375 5359534745SIGNON:  DB       'SYSGEN VER '
 0380 322F30             DB       VERS/10+'0','.',VERS MOD 10+'0'
 0383 00                 DB       0
 0384 534F555243ASKGET:  DB       'SOURCE DRIVE NAME (OR RETURN TO SKIP)',0
 03AA 534F555243GETMSG:  DB       'SOURCE ON '
 03B4            GDISK:  DS       1       ;FILLED IN AT GET FUNCTION
 03B5 2C20544845         DB       ', THEN TYPE RETURN',0
 03C8 4445535449ASKPUT:  DB       'DESTINATION DRIVE NAME (OR RETURN TO REBOOT)',0
 03F5 4445535449PUTMSG:  DB       'DESTINATION ON '
 0404            PDISK:  DS       1       ;FILLED IN AT PUT FUNCTION
 0405 2C20544845         DB       ', THEN TYPE RETURN',0
 0418 5045524D41ERRMSG:  DB       'PERMANENT ERROR, TYPE RETURN TO IGNORE',0
 043F 46554E4354DONE:    DB       'FUNCTION COMPLETE',0
 0451 494E56414CQDISK:   DB       'INVALID DRIVE NAME (USE A, B, C, OR D)',0
 0478 4E4F20534FNOFILE:  DB       'NO SOURCE FILE ON DISK',0
                 BADFILE:
 048F 534F555243         DB       'SOURCE FILE INCOMPLETE',0
                 ;
                 ;       VARIABLES
 04A6            SDISK:  DS       1       ;SELECTED DISK FOR CURRENT OPERATION
 04A7            TRACK:  DS       1       ;CURRENT TRACK
 04A8            SECTOR: DS       1       ;CURRENT SECTOR
```

```
CP/M MACRO ASSM 2.0     #010     SYSGEN - SYSTEM GENERATION PROGRAM 8/79

 04A9            RW:     DS       1       ;READ IF 0, WRITE IF 1
 04AA            DMADDR: DS       2       ;CURRENT DMA ADDRESS
 04AC            RETRY:  DS       1       ;NUMBER OF TRIES ON THIS SECTOR
 04AD                    DS       STACKSIZE*2
                 STACK:
B04CD                    END
```

```
0384 ASKGET     03C8 ASKPUT     036E BADDISK    048F BADFILE    02CD BADRD
0005 BDOS       0000 BOOT       025F CHKRW      0195 CRMSG      0001 CONI
000D CR         018A CRLF       01BF DMA        04AA DMADDR     0002 CONO
043F DONE       0014 DREADF     01D7 DREAD      0277 ENDRW      0266 ENDTRK
0418 ERRMSG     005C FCB        007C FCBCR      03B4 GDISK      02F1 GETC
0175 GETCHAR    03AA GETMSG     01E1 GETPUT     02D6 GETSYS     000A LF
0900 LOADP      0007 LOG2SEC    000A MAXTRY     016A MULTSEC    0004 NDISKS
0478 NOFILE     001A NSECTS     0002 NTRKS      01DC OPEN       000F OPENF
0128 OST        019A OUTMSG     0404 PDISK      02A2 PRERD      0183 PUTCHAR
0337 PUTC       03F5 PUTMSG     031C PUTSYS     0451 QDISK      02B5 RDINP
029C RDOK       0024 READF      01C7 READ       0363 REBOOT     04AC RETRY
04A9 RW         0200 RWSEC      01EC RWTRK      04A6 SDISK      01B7 SEC
0080 SECSIZ     04A8 SECTOR     001B SELDSK     000E SELF       01A6 SEL
0021 SETDMA     001E SETSEC     001B SETTRK     0375 SIGNON     0001 SKEW
0129 SPT        04CD STACK      0010 STACKSIZE  0270 START      0100 TPA
04A7 TRACK      012A TRAN       01AF TRK        024B TRYOK      025C TRYREAD
022F TRYSEC     0014 VERS       0001 WBOOT      01CF WRITE      0027 WRITF
```

```
CP/M MACRO ASSEM 2.0     #001     ASM COMMON DATA AREA

                                  TITLE   'ASM COMMON DATA AREA'
                         ;
                         ;        COPYRIGHT (C) 1977, 1978
                         ;        DIGITAL RESEARCH
                         ;        BOX 579, PACIFIC GROVE
                         ;        CALIFORNIA, 93950
                         ;
                         ;        COMMON DATA FOR CP/M ASSEMBLER MODULE
0100                             ORG     100H
20F0 =           ENDA    EQU     20F8H   ;END OF ASSEMBLER PROGRAM
0005 =           BDOS    EQU     5H      ;ENTRY TO BDOS, USED TO COMPUTE END MEMORY
0100 318002              LXI     SP,ENDMOD
0103 2A0600              LHLD    BDOS+1
0106 22CD01              SHLD    SYMAX   ;COMPUTE END OF MEMORY
0109 C38002              JMP     ENDMOD
010C 20434F5859COPY:     DB      ' COPYRIGHT(C) 1978, DIGITAL RESEARCH'
010C                     ORG     COPY
                         ;
                         ;        PRINT BUFFER AND PRINT BUFFER POINTER
0078 =           PBMAX   EQU     120     ;MAX PRINT BUFFER
010C             PBUFF:  DS      PBMAX
0184             PBP:    DS      1       ;PRINT BUFFER POINTER
                         ;
                         ;        SCANNER PARAMETERS
0185             TOKEN:  DS      1       ;CURRENT TOKEN
0186             VALUE:  DS      2       ;BINARY VALUE FOR NUMBERS
0188             ACCLEN: DS      1       ;ACCUMULATOR LENGTH
0040 =           ACMAX   EQU     64      ;LENGTH OF ACCUMULATOR
0189             ACCUM:  DS      ACMAX   ;ACCUMULATOR (MUST FOLLOW ACCLEN)
                         ;
                         ;        OPERAND EXPRESSION EVALUATOR PARAMETERS
01C9             EVALUE: DS      2       ;VALUE OF EXPRESSION AFTER EVALUATION
                         ;
                         ;        SYMBOL TABLE  MODULE PARAMETERS
01CB F020        SYTOP:  DW      ENDA    ;FIRST LOCATION AVAILABLE FOR SYMBOL TABLE
01CD             SYMAX:  DS      2       ;LAST AVAILABLE LOCATION FOR SYMBOL TABLE
                         ;
                         ;        MISCELLANEOUS DATA AREAS
01CF             PASS:   DS      1       ;PASS 0 OR 1
01D0             FPC:    DS      2       ;FILL ADDRESS FOR NEXT HEX RECORD
01D2             ASPC:   DS      2       ;ASSEMBLER'S PSEUDO PC
01D4 F020        SYBAS:  DW      ENDA    ;SYMBOL TABLE BASE
01D6             SYADR:  DS      2       ;CURRENT SYMBOL BASE
0200 =           ENDMOD  EQU     ($ AND 0FF00H)+100H
01D8                     END
```



```
CP/M MACRO ASSEM 2.0     #002     ASM COMMON DATA AREA

0188 ACCLEN      0189 ACCUM       0040 ACMAX       01D2 ASPC        0005 BDOS
010C COPY        20F0 ENDA        0200 ENDMOD      01C9 EVALUE      01D0 FPC
01CF PASS        0078 PBMAX       0184 PBP         010C PBUFF       01D6 SYADR
01D4 SYBAS       01CD SYMAX       01CB SYTOP       0185 TOKEN       0186 VALUE
```

```
CP/M MACRO ASSEM 2.0     #001     ASM IO MODULE

                                  TITLE   'ASM IO MODULE'
                         ;        I/O MODULE FOR CP/M ASSEMBLER
                         ;
0200                             ORG     200H
0000 =           BOOT    EQU     0000H   ;REBOOT LOCATION
                         ;        I/O MODULE ENTRY POINTS
0200 C3E80C              JMP     INIT    ;INITIALIZE, START ASSEMBLER
0203 C3A10D              JMP     SETUP   ;FILE SETUP
0206 C3CA0D              JMP     GNC     ;GET NEXT CHARACTER
0209 C3340E              JMP     PNC     ;PUT NEXT OUTPUT CHARACTER
020C C3AA0E              JMP     PNB     ;PUT NEXT HEX BYTE
020F C3DE0E              JMP     PCHAR   ;PRINT CONSOLE CHARACTER
0212 C38C0C              JMP     PCON    ;PRINT CONSOLE BUFFER TO CRLF
0215 C3000F              JMP     WOBUFF  ;WRITE OUTBUFFER
0218 C32F0F              JMP     PERR    ;PLACE ERROR CHARACTER INTO PBUFF
021B C34C1B              JMP     DNEX    ;PLACE HEX BYTE INTO OUTPUT BUFFER
021E C3390F              JMP     EOR     ;END OF ASSEMBLY
                         ;        DATA FOR I/O MODULE
0221             BPC:    DS      2       ;BASE PC FOR CURRENT HEX RECORD
0223             DBL:    DS      1       ;HEX BUFFER LENGTH
0224             DBUFF:  DS      16      ;HEX BUFFER
                         ;
                         ;        DISK NAMES
0234             CDISK:  DS      1       ;CURRENTLY SELECTED DISK
0235             ADISK:  DS      1       ; ASM DISK
0236             PDISK:  DS      1       ; PRN DISK
0237             HDISK:  DS      1       ; HEX DISK
                         ;
                         ;
                         ;
                         ;        COMMON EQUATES
0078 =           QBMAX   EQU     120     ;MAX PRINT SIZE
010C =           QBUFF   EQU     10CH    ;PRINT BUFFER
0184 =           QBP     EQU     QBUFF+QBMAX     ;PRINT BUFFER POINTER
                         ;
0185 =           TOKEN   EQU     QBP+1   ;CURRENT TOKEN UDER SCAN
0186 =           VALUE   EQU     TOKEN+1 ;VALUE OF NUMBER IN BINARY
0188 =           ACCLEN  EQU     VALUE+2 ;ACCUMULATOR LENGTH
0040 =           ACMAX   EQU     64      ;MAX ACCUMULATOR LENGTH
0189 =           ACCUM   EQU     ACCLEN+1
                         ;
01C9 =           EVALUE  EQU     ACCUM+ACMAX     ;VALUE FROM EXPRESSION ANALYSIS
                         ;
01CB =           SYTOP   EQU     EVALUE+2        ;CURRENT SYMBOL TOP
01CD =           SYMAX   EQU     SYTOP+2         ;MAX ADDRESS+1
                         ;
01CF =           PASS    EQU     SYMAX+2 ;CURRENT PASS NUMBER
01D0 =           FPC     EQU     PASS+1  ;FILL ADDRESS FOR DHEX ROUTINE
01D2 =           ASPC    EQU     FPC+2   ;ASSEMBLER'S PSEUDO PC
                         ;
000D =           CR      EQU     0DH     ;CARRIAGE RETURN
000A =           LF      EQU     0AH     ;LINE FEED
001A =           EOF     EQU     1AH     ;END OF FILE MARK
                         ;
                         ;
                         ;        DOS ENTRY POINTS
```

```
CP/M MACRO ASSEM 2.0     0002     ASM IO MODULE

0005 =          BDOS    EQU     5H      ;BDOS ENTRY POINT
0001 =          READC   EQU     1       ;READ CONSOLE DEVICE
0002 =          WRITC   EQU     2       ;WRITE CONSOLE DEVICE
000B =          REDYC   EQU     11      ;CONSOLE CHARACTER READY
000E =          SELECT  EQU     14      ;SELECT DISK SPECIFIED BY REGISTER E
000F =          OPENF   EQU     15      ;OPEN FILE
0010 =          CLOSF   EQU     16      ;CLOSE FILE
0013 =          DELEF   EQU     19      ;DELETE FILE
0014 =          READF   EQU     20      ;READ FILE
0015 =          WRITF   EQU     21      ;WRITE FILE
0016 =          MAKEF   EQU     22      ;MAKE A FILE
0019 =          CSEL    EQU     25      ;RETURN CURRENTLY SELECTED DISK
001A =          SETDM   EQU     26      ;SET DMA ADDRESS
                ;
                ;       FILE AND BUFFERING PARAMETERS
0008 =          NSB     EQU     8       ;NUMBER OF SOURCE BUFFERS
0006 =          NPB     EQU     6       ;NUMBER OF PRINT BUFFERS
0006 =          NHB     EQU     6       ;NUMBER OF HEX BUFFERS
                ;
0400 =          SSIZE   EQU     NSB*128
0300 =          PSIZE   EQU     NPB*128
0300 =          HSIZE   EQU     NHB*128
                ;
                ;       FILE CONTROL BLOCKS
0238            SCB:    DS      9       ;FILE NAME
0241 41534D             DB      'ASM'   ;FILE TYPE
0244            SCBR:   DS      1       ;REEL NUMBER (ZEROED IN SETUP)
0245                    DS      19      ;MISC AND DISK MAP
0258            SCBCR:  DS      1       ;CURRENT RECORD (ZEROED IN SETUP)
                ;
0259            PCB:    DS      9
0262 50524E00           DB      'PRN',0
0266                    DS      19
0279 00                 DB      0       ;RECORD TO WRITE NEXT
                ;
027A            HCB:    DS      9
0283 48455800           DB      'HEX',0
0287                    DS      19
029A 00                 DB      0
                ;
                ;       POINTERS AND BUFFERS
029B 0004       SBP:    DW      SSIZE   ;NEXT CHARACTER POSITION TO READ
029D            SBUFF:  DS      SSIZE
                ;
069D 0000       PBP:    DW      0
069F            PBUFF:  DS      PSIZE
                ;
099F 0000       HBP:    DW      0
09A1            HBUFF:  DS      HSIZE
005C =          FCB     EQU     5CH     ;FILE CONTROL BLOCK ADDRESS
0001 =          FNM     EQU     1       ;POSITION OF FILE NAME
0009 =          FLN     EQU     9       ;FILE NAME LENGTH
0080 =          BUFF    EQU     80H     ;INPUT DISK BUFFER ADDRESS
                ;
                SEL:    ;SELECT DISK IN REG-A
0CA1 213402             LXI     H,CDISK
```



```
CP/M MACRO ASSEM 2.0     0003     ASM IO MODULE

0CA4 BE                 CMP     M       ;SAME?
0CA5 C8                 RZ
0CA6 77                 MOV     M,A     ;CHANGE CURRENT DISK
0CA7 5F                 MOV     E,A
0CA8 0E0E               MVI     C,SELECT
0CAA CD0500             CALL    BDOS
0CAD C9                 RET
                ;
                SCNP:   ;SCAN THE NEXT PARAMETER
0CAE 23                 INX     H
0CAF 7E                 MOV     A,M
0CB0 FE20               CPI     ' '
0CB2 CAB80C             JZ      SCNP0
0CB5 DE41               SBI     'A'     ;NORMALIZE
0CB7 C9                 RET
0CB8 3A3402     SCNP0:  LDA     CDISK
0CBB C9                 RET
                ;
                PCON:   ;PRINT MESSAGE AT H,L TO CONSOLE DEVICE
0CBC 7E                 MOV     A,M
0CBD CDDE0E             CALL    PCHAR
0CC0 7E                 MOV     A,M
0CC1 23                 INX     H
0CC2 FE0D               CPI     CR
0CC4 C2BC0C             JNZ     PCON
0CC7 3E0A               MVI     A,LF
0CC9 CDDE0E             CALL    PCHAR
0CCC C9                 RET
                ;
                FNAME:  ;FILL NAME FROM DEFAULT FILE CONTROL BLOCK
0CCD 115C00             LXI     D,FCB
0CD0 0609               MVI     B,FLN
0CD2 1A         FNAM0:  LDAX    D       ;GET NEXT FILE CHARACTER
0CD3 FE3F               CPI     '?'
0CD5 CABB0D             JZ      FNERR   ;FILE NAME ERROR
0CD8 77                 MOV     M,A     ;STORE TO FILE CNTRL BLOCK
0CD9 23                 INX     H
0CDA 13                 INX     D
0CDB 05                 DCR     B
0CDC C2D20C             JNZ     FNAM0   ;FOR NEXT CHARACTER
0CDF C9                 RET
                ;
                INIT:   ;SET UP STACK AND FILES, START ASSEMBLER
0CE0 21A00F             LXI     H,TITL
0CE3 CDBC0C             CALL    PCON
0CE6 C33F0D             JMP     SET0
                ;
                OPEN:   ;OPEN FILE ADDRESSED BY D,E
0CE9 0E0F               MVI     C,OPENF
0CEB CD0500             CALL    BDOS
0CEE FEFF               CPI     255
0CF0 C0                 RNZ
                ;       OPEN ERROR
0CF1 21B90F             LXI     H,ERROP
0CF4 CDBC0C             CALL    PCON
0CF7 C30000             JMP     BOOT
```

```
                ;
                CLOSE.  ;CLOSE FILE ADDRESSED BY D,E
0CFA 0E10               MVI     C,CLOSF
0CFC CD0500             CALL    BDOS
0CFF FEFF               CPI     255
0D01 C0                 RNZ             ;CLOSE OK
0D02 212910             LXI     H,ERRCL
0D05 CDBC0C             CALL    PCON
0D08 C30000             JMP     BOOT
                ;
                DELETE. ;DELETE FILE ADDRESSED BY D,E
0D0B 0E13               MVI     C,DELEF
0D0D C30500             JMP     BDOS
                ;
                MAKE.   ;MAKE FILE ADDRESSED BY D,E
0D10 0E16               MVI     C,MAKEF
0D12 CD0500             CALL    BDOS
0D15 FEFF               CPI     255
0D17 C0                 RNZ
                ;       MAKE ERROR
0D18 21D00F             LXI     H,ERRMA
0D1B CDBC0C             CALL    PCON
0D1E C30000             JMP     BOOT
                ;
0D21 3A3502     SELA.   LDA     ADISK
0D24 CDA10C             CALL    SEL
0D27 C9                 RET
                ;
                NPR.    ;RETURN ZERO FLAG IF NO PRINT FILE
0D28 3A3602             LDA     PDISK
0D2B FE19               CPI     'Z'-'A'
0D2D C8                 RZ
0D2E FE17               CPI     'X'-'A' ;CONSOLE
0D30 C9                 RET
                ;
0D31 3A3602     SELP.   LDA     PDISK
0D34 CDA10C             CALL    SEL
0D37 C9                 RET
                ;
0D38 3A3702     SELH.   LDA     HDISK
0D3B CDA10C             CALL    SEL
0D3E C9                 RET
                ;
                SET0.   ;SET UP FILES FOR INPUT AND OUTPUT
0D3F 3A5C00             LDA     FCB     ;GET FIRST CHARACTER
0D42 FE20               CPI     ' '     ;MAY HAVE FORGOTTEN NAME
0D44 CA880D             JZ      FNERR   ;FILE NAME ERROR
0D47 0E19               MVI     C,CSEL  ;CURRENT DISK?
0D49 CD0500             CALL    BDOS    ;GET IT TO REG-A
0D4C 323402             STA     CDISK
                ;
                ;       SCAN PARAMETERS
0D4F 216400             LXI     H,FCB+FLN-1
0D52 CDAE0C             CALL    SCNP
0D55 323502             STA     ADISK
0D58 CDAE0C             CALL    SCNP
```



```
0D5B 323702             STA     HDISK
0D5E CDAE0C             CALL    SCNP
0D61 323602             STA     PDISK
                ;
0D64 213002             LXI     H,SCB   ;ADDRESS SOURCE FILE CONTROL BLOCK
0D67 CDCD0C             CALL    FNAME           ;FILE NAME OBTAINED FROM DEFAULT
                ;
0D6A CD200D             CALL    NPR     ;Z OR X?
0D6D CA830D             JZ      NOPR
0D70 215902             LXI     H,PCB   ;ADDRESS PRINT FILE CONTROL BLOCK
0D73 E5                 PUSH    H       ;SAVE A COPY FOR OPEN
0D74 E5                 PUSH    H       ;SAVE A COPY FOR DELETE
0D75 CDCD0C             CALL    FNAME   ;FILL PCB
0D78 CD310D             CALL    SELP
0D7B D1                 POP     D       ;FCB ADDRESS
0D7C CD0B0D             CALL    DELETE
0D7F D1                 POP     D       ;FCB ADDRESS
0D80 CD100D             CALL    MAKE
                ;
                NOPR.   ;TEST FOR HEX FILE
0D83 3A3702             LDA     HDISK
0D86 FE19               CPI     'Z'-'A'
0D88 CA9E0D             JZ      NOHEX
0D8B 217A02             LXI     H,HCB
0D8E E5                 PUSH    H
0D8F E5                 PUSH    H
0D90 CDCD0C             CALL    FNAME
0D93 CD300D             CALL    SELH
0D96 D1                 POP     D
0D97 CD0B0D             CALL    DELETE
0D9A D1                 POP     D
0D9B CD100D             CALL    MAKE
                ;
                ;       FILES SET UP, CALL ASSEMBLER
0D9E C30011     NOHEX.  JMP     ENDMOD
                ;
                SETUP.  ;SETUP INPUT FILE FOR SOURCE PROGRAM
0DA1 210004             LXI     H,SSIZE
0DA4 229B02             SHLD    SBP     ;CAUSE IMMEDIATE READ
0DA7 AF                 XRA     A       ;ZERO VALUE
0DA8 324402             STA     SCBR    ;CLEAR REEL NUMBER
0DAB 325B02             STA     SCBCR           ;CLEAR CURRENT RECORD
0DAE 322302             STA     DBL             ;CLEAR HEX BUFFER LENGTH
0DB1 CD210D             CALL    SELA
0DB4 113002             LXI     D,SCB
0DB7 CDE90C             CALL    OPEN
                ;
0DBA C9                 RET
                ;
                FNERR.  ;FILE NAME ERROR
0DBB 21E30F             LXI     H,ERRFN
0DBE CDBC0C             CALL    PCON
0DC1 C30000             JMP     BOOT
                ;
                ;
                GCOMP.  ;COMPARE D,E AGAINS H,L
```

```
CP/M MACRO ASSEM 2.0    #006    ASM IO MODULE

0DC4 7A                 MOV     A,D
0DC5 BC                 CMP     H
0DC6 C0                 RNZ
0DC7 7B                 MOV     A,E
0DC8 BD                 CMP     L
0DC9 C9                 RET
                ;
                GNC.    ;GET NEXT CHARACTER FROM SOURCE BUFFER
0DCA C5                 PUSH    B
0DCB D5                 PUSH    D
0DCC E5                 PUSH    H       ;ENVIRONMENT SAVED
0DCD 2A9B02             LHLD    SBP
0DD0 110004             LXI     D,SSIZE
0DD3 CDC40D             CALL    GCOMP
0DD6 C2198E             JNZ     GNC2
                ;
                ;       READ ANOTHER BUFFER
0DD9 CD210D             CALL    SELA
0DDC 210000             LXI     H,0
0DDF 229B02             SHLD    SBP
0DE2 0608               MVI     B,NSB   ;NUMBER OF SOURCE BUFFERS
0DE4 219D02             LXI     H,SBUFF
                GNC0.   ;READ 128 BYTES
0DE7 C5                 PUSH    B       ;SAVE COUNT
0DE8 E5                 PUSH    H       ;SAVE BUFFER ADDRESS
0DE9 0E14               MVI     C,READF
0DEB 113802             LXI     D,SCB
0DEE CD0500             CALL    BDOS    ;PERFORM THE READ
0DF1 E1                 POP     H       ;RESTORE BUFFER ADDRESS
0DF2 C1                 POP     B       ;RESTORE BUFFER COUNT
0DF3 B7                 ORA     A       ;SET FLAGS
0DF4 0E80               MVI     C,128
0DF6 C20D0E             JNZ     GNC1
                ;       NORMAL READ OCCURRED
0DF9 110000             LXI     D,BUFF  ;SOURCE BUFFER ADDRESS
0DFC 0E80               MVI     C,128
0DFE 1A         MOV0.   LDAX    D       ;GET CHARACTER
0DFF 77                 MOV     M,A     ;STORE CHARACTER
0E00 13                 INX     D
0E01 23                 INX     H
0E02 0D                 DCR     C
0E03 C2FE0D             JNZ     MOV0
                ;       BUFFER LOADED, TRY NEXT BUFFER
                ;
0E06 05                 DCR     B
0E07 C2E70D             JNZ     GNC0
0E0A C3190E             JMP     GNC2
                ;
                GNC1.   ;EOF OR ERROR
0E0D FE03               CPI     3       ;ALLOW 0,1,2
0E0F D2280E             JNC     FRERR   ;FILE READ ERROR
0E12 361A       GNCE.   MVI     M,EOF   ;STORE AND END OF FILE CHARACTER
0E14 23                 INX     H
0E15 0D                 DCR     C
0E16 C2120E             JNZ     GNCE    ;FILL CURRENT BUFFER WITH EOF'S
                ;
```



```
CP/M MACRO ASSEM 2.0    #007    ASM IO MODULE

                GNC2.   ;GET CHARACTER TO ACCUMULATOR AND RETURN
0E19 119D02             LXI     D,SBUFF
0E1C 2A9B02             LHLD    SBP
0E1F E5                 PUSH    H       ;SAVE CURRENT SBP
0E20 23                 INX     H       ;READY FOR NEXT READ
0E21 229B02             SHLD    SBP
0E24 E1                 POP     H       ;RESTORE PREVIOUS SBP
0E25 19                 DAD     D       ;ABSOLUTE ADDRESS OF CHARACTER
0E26 7E                 MOV     A,M     ;GET IT
0E27 E1                 POP     H
0E28 D1                 POP     D
0E29 C1                 POP     B
0E2A C9                 RET
                ;
0E2B 21FA0F     FRERR.  LXI     H,ERRFR
0E2E CD0C0C             CALL    PCON    ;PRINT READ ERROR MESSAGE
0E31 C30000             JMP     BOOT
                ;
                PNC.    ;SAME AT PNCF, BUT ENVIRONMENT IS SAVED FIRST
0E34 C5                 PUSH    B
                ;       CHECK FOR CONSOLE OUTPUT / NO OUTPUT
0E35 47                 MOV     B,A     ;SAVE CHARACTER
0E36 3A3602             LDA     PDISK   ;Z OR X?
0E39 FE19               CPI     'Z'-'A' ;Z NO OUTPUT
0E3B CA518E             JZ      PNRET
                ;
0E3E FE17               CPI     'X'-'A'
0E40 78                 MOV     A,B     ;RECOVER CHAR FOR CON OUT
0E41 C24A0E             JNZ     PNCO
0E44 CDDE0E             CALL    PCHAR
0E47 C3518E             JMP     PNRET
                ;
                ;       NOT X OR Z, SO PRINT IT
0E4A D5         PNCO.   PUSH    D
0E4B E5                 PUSH    H
0E4C CD530E             CALL    PNCF
0E4F E1                 POP     H
0E50 D1                 POP     D
0E51 C1         PNRET.  POP     B
0E52 C9                 RET
                ;
                PNCF.   ;PRINT NEXT CHARACTER
0E53 2A9D06             LHLD    PBP
0E56 EB                 XCHG
0E57 219F06             LXI     H,PBUFF
0E5A 19                 DAD     D
0E5B 77                 MOV     M,A     ;CHARACTER STORED AT PBP IN PBUFF
0E5C EB                 XCHG            ;PBP TO H,L
0E5D 23                 INX     H       ;POINT TO NEXT CHARACTER
0E5E 229D06             SHLD    PBP     ;REPLACE IT
0E61 EB                 XCHG
0E62 210003             LXI     H,PSIZE
0E65 CDC40D             CALL    GCOMP   ;AT END OF BUFFER?
0E68 C0                 RNZ             ;RETURN IF NOT
                ;
                ;       OVERFLOW, WRITE BUFFER
```

```
0E69 CD318D             CALL    SELP
0E6C 210000             LXI     H,0
0E6F 229D86             SHLD    PBP
0E72 219F86             LXI     H,PBUFF
0E75 115982             LXI     D,PCB   ;D,E ADDRESS FILE CONTROL BLOCK
0E78 0686               MVI     B,NPB   ;NUMBER OF BUFFERS TO B
                ;       (DROP THROUGH TO WBUFF)
                ;
                WBUFF.  ;WRITE BUFFERS STARTING AT H,L FOR B BUFFERS
                ;       CHECK FOR EOF'S
0E7A 7E                 MOV     A,M
0E7B FE1A               CPI     EOF
0E7D C8                 RZ              ;DON'T DO THE WRITE
                ;
0E7E C5                 PUSH    B       ;SAVE NUMBER OF BUFFERS
0E7F D5                 PUSH    D       ;SAVE FCB ADDRESS
0E80 0E80               MVI     C,128   ;READY FOR MOVE
0E82 118000             LXI     D,BUFF
                WBUF0.  ;MOVE TO BUFFER
0E85 7E                 MOV     A,M     ;GET CHARACTER
0E86 12                 STAX    D       ;PUT CHARACTER
0E87 23                 INX     H
0E88 13                 INX     D
0E89 0D                 DCR     C
0E8A C2858E             JNZ     WBUF0
                ;
                ;       WRITE BUFFER
0E8D D1                 POP     D       ;RECOVER FCB ADDRESS
0E8E D5                 PUSH    D       ;SAVE IT AGAIN FOR LATER
0E8F E5                 PUSH    H       ;SAVE BUFFER ADDRESS
0E90 0E15               MVI     C,WRITF ;BDOS WRITE FUNCTION
0E92 CD0500             CALL    BDOS
0E95 E1                 POP     H       ;RECOVER BUFFER ADDRESS
0E96 D1                 POP     D       ;RECOVER FCB ADDRESS
0E97 C1                 POP     B       ;RECOVER BUFFER COUNT
0E98 B7                 ORA     A       ;SET ERROR RETURN FLAGS
0E99 C2A10E             JNZ     FWERR
                ;
                ;       WRITE OK
0E9C 05                 DCR     B
0E9D C8                 RZ              ;RETURN IF NO MORE BUFFERS TO WRITE
0E9E C37A8E             JMP     WBUFF
                ;
                FWERR.  ;ERROR IN WRITE
0EA1 211110             LXI     H,ERRFW
0EA4 CD8C0C             CALL    PCON    ;ERROR MESSAGE OUT
0EA7 C3770F             JMP     EDRC    ;TO CLOSE AND REBOOT
                ;
                ;
                PNB.    ;PUT NEXT HEX BYTE
0EAA C5                 PUSH    B
0EAB D5                 PUSH    D
0EAC E5                 PUSH    H
0EAD CD848E             CALL    PNBF
0EB0 E1                 POP     H
0EB1 D1                 POP     D
```



```
0EB2 C1                 POP     B
0EB3 C9                 RET
                ;
                PNBF.   ;PUT NEXT BYTE
                ;       (SIMILAR TO THE PNCF SUBROUTINE)
0EB4 2A9F89             LHLD    HBP
0EB7 EB                 XCHG
0EB8 21A189             LXI     H,HBUFF
0EBB 19                 DAD     D
0EBC 77                 MOV     M,A     ;CHARACTER STORED AT HBP IN HBUFF
0EBD EB                 XCHG
0EBE 23                 INX     H       ;HBP INCREMENTED
0EBF 229F89             SHLD    HBP
0EC2 EB                 XCHG            ;BACK TO D,E
0EC3 210083             LXI     H,HSIZE
0EC6 CDC48D             CALL    GCOMP   ;EQUAL?
0EC9 C0                 RNZ
                ;
                ;       OVERFLOW, WRITE BUFFERS
0ECA CD388D             CALL    SELH
0ECD 210000             LXI     H,0
0ED0 229F89             SHLD    HBP
0ED3 21A189             LXI     H,HBUFF
0ED6 117A82             LXI     D,HCB   ;FILE CONTROL BLOCK FOR HEX FILE
0ED9 0606               MVI     B,NHB
0EDB C37A8E             JMP     WBUFF   ;WRITE BUFFERS
                ;
                PCHAR.  ;PRINT CHARACTER IN REGISTER A
0EDE C5                 PUSH    B
0EDF D5                 PUSH    D
0EE0 E5                 PUSH    H
0EE1 0E02               MVI     C,WRITC
0EE3 5F                 MOV     E,A
0EE4 CD0500             CALL    BDOS
0EE7 E1                 POP     H
0EE8 D1                 POP     D
0EE9 C1                 POP     B
0EEA C9                 RET
                ;
                WOCHAR. ;WRITE CHARACTER IN REG-A WITH REFLECT AT CONSOLE IF ERR
0EEB 4F                 MOV     C,A     ;SAVE THE CHAR
0EEC CD348E             CALL    PNC     ;PRINT CHAR
0EEF 3A8C81             LDA     OBUFF
0EF2 FE20               CPI     ' '
0EF4 C8                 RZ
                ;       ERROR IN LINE
0EF5 3A3682             LDA     PDISK
0EF8 FE17               CPI     'X'-'A'
0EFA C8                 RZ              ;ALREADY PRINTED IF 'X'
                ;
0EFB 79                 MOV     A,C     ;RECOVER CHARACTER
0EFC CDDE0E             CALL    PCHAR   ;PRINT IT
0EFF C9                 RET
                ;
                WOBUFF. ;WRITE THE OUTPUT BUFFER TO THE PRINT FILE
0F00 3A8481             LDA     OBP     ;GET CHARACTER COUNT
```

```
CP/M MACRO ASSEM 2.0    0010    ASM IO MODULE

0F03 210C01             LXI     H,QBUFF ;BASE OF BUFFER
0F06 B7         WOB0.   ORA     A       ;ZERO COUNT?
0F07 CA150F             JZ      WOBE
                ;       NOT END, SAVE COUNT AND GET CHARACTER
0F0A 47                 MOV     B,A     ;SAVE COUNT
0F0B 7E                 MOV     A,M
0F0C CDEB0E             CALL    WOCHAR  ;WRITE CHARACTER
0F0F 23                 INX     H       ;ADDRESS NEXT CHARACTER OF BUFFER
0F10 78                 MOV     A,B     ;GET COUNT
0F11 3D                 DCR     A
0F12 C3060F             JMP     WOB0
                ;
                WOBE.   ;END OF PRINT - ZERO QBP
0F15 320401             STA     QBP
                ;       FOLLOW BY CR LF
0F18 3E0D               MVI     A,CR
0F1A CDEB0E             CALL    WOCHAR
0F1D 3E0A               MVI     A,LF
0F1F CDEB0E             CALL    WOCHAR
0F22 210C01             LXI     H,QBUFF
0F25 3E78               MVI     A,QBMAX ;READY TO BLANK OUT
0F27 3620       WOB2.   MVI     M,' '
0F29 23                 INX     H
0F2A 3D                 DCR     A
0F2B C2270F             JNZ     WOB2
0F2E C9                 RET
                ;
                ;
                PERR.   ;FILL QBUFF ERROR MESSAGE POSITION
0F2F 47                 MOV     B,A     ;SAVE CHARACTER
0F30 210C01             LXI     H,QBUFF
0F33 7E                 MOV     A,M
0F34 FE20               CPI     ' '
0F36 C0                 RNZ     ;DON'T CHANGE IT IF ALREADY SET
0F37 70                 MOV     M,B     ;STORE ERROR CHARACTER
0F38 C9                 RET
                ;
                EOR.    ;END OF ASSEMBLER
0F39 CD280D             CALL    MPR     ;Z OR A?
0F3C CA4F0F             JZ      EOPR
                ;       FILL OUTPUT FILES WITH EOF'S
0F3F 2A9D06     EOR2.   LHLD    PBP
0F42 7D                 MOV     A,L
0F43 B4                 ORA     H       ;VALUE ZERO?
0F44 CA4F0F             JZ      EOPR
0F47 3E1A               MVI     A,EOF   ;CTL-Z IS END OF FILE
0F49 CD349E             CALL    PNC     ;PUT ENDFILES IN PRINT BUFFER
0F4C C33F0F             JMP     EOR2    ;EVENTUALLY BUFFER IS WRITTEN
                ;
                EOPR.   ;END OF PRINT FILE, CHECK HEX
0F4F 3A3702             LDA     HDISK
0F52 FE19               CPI     'Z'-'A'
0F54 CA770F             JZ      EORC
                EOR0.   ;WRITE TERMINATING RECORD INTO HEX FILE
0F57 3A2302             LDA     DBL     ;MAY BE ZERO ALREADY
0F5A B7                 ORA     A
```



```
CP/M MACRO ASSEM 2.0    0011    ASM IO MODULE

0F5B C4B810             CNZ     WHEX    ;WRITE HEX BUFFER IF NOT ZERO
0F5E 2AD001             LHLD    FPC     ;GET CURRENT FPC AS LAST ADDRESS
0F61 222102             SHLD    BPC     ;RECORD LENGTH ZERO, BASE ADDRESS 0000 .
0F64 CDB810             CALL    WHEX    ;WRITE HEX BUFFER
                ;
                ;       NOW CLEAR OUTPUT BUFFER FOR HEX FILE
0F67 2A9F09     EOR1.   LHLD    HBP
0F6A 7D                 MOV     A,L
0F6B B4                 ORA     H
0F6C CA770F             JZ      EORC
0F6F 3E1A               MVI     A,EOF
0F71 CDAA8E             CALL    PNB
0F74 C3670F             JMP     EOR1
                ;
                ;       CLOSE FILES AND TERMINATE
                EORC.
0F77 CD280D             CALL    MPR
0F7A CA860F             JZ      EORPC
0F7D CD310D             CALL    SELP
0F80 115902             LXI     D,PCB
0F83 CDFA0C             CALL    CLOSE
                EORPC.
0F86 3A3702             LDA     HDISK
0F89 FE19               CPI     'Z'-'A'
0F8B CA970F             JZ      EORHC
0F8E CD300D             CALL    SELH
0F91 117A02             LXI     D,HCB
0F94 CDFA0C             CALL    CLOSE
                ;
                EORHC.
0F97 213C10             LXI     H,ENDA
0F9A CD8C0C             CALL    PCON
0F9D C30000             JMP     BOOT
                ;
0FA0 43502F4D20TITL.    DB      'CP/M ASSEMBLER - VER 1.4',CR
0FB9 4E4F20534FERROP.   DB      'NO SOURCE FILE PRESENT',CR
0FD0 4E4F204449ERRMA.   DB      'NO DIRECTORY SPACE',CR
0FE3 534F555243ERRFH.   DB      'SOURCE FILE NAME ERROR',CR
0FFA 534F555243ERRFR.   DB      'SOURCE FILE READ ERROR',CR
1011 4F55545055ERRFW.   DB      'OUTPUT FILE WRITE ERROR',CR
1029 43414E4E4FERRCL.   DB      'CANNOT CLOSE FILES',CR
103C 454E44204FENDA.    DB      'END OF ASSEMBLY',CR
                ;
                DHEX.   ;DATA TO HEX BUFFER (BYTE IN REG-A)
104C C5                 PUSH    B
104D 47                 MOV     B,A     ;HOLD CHARACTER FOR 'Z' TEST
104E 3A3702             LDA     HDISK
1051 FE19               CPI     'Z'-'A'
1053 78                 MOV     A,B     ;RECOVER CHARACTER
1054 CA9810             JZ      DHRET
1057 D5                 PUSH    D       ;ENVIRONMENT SAVED
1058 F5                 PUSH    PSW     ;SAVE DATA BYTE
1059 212302             LXI     H,DBL   ;CURRENT LENGTH
105C 7E                 MOV     A,M     ;TO ACCUM
105D B7                 ORA     A       ;ZERO?
105E CA8410             JZ      DHEX3
```

```
                ;
                ;       LENGTH NOT ZERO, MAY BE FULL BUFFER
1061 FE10               CPI     16
1063 DA6C10             JC      DHEX1   ;BR IF LESS THAN 16 BYTES
                ;       BUFFER FULL, DUMP IT
1066 CDB810             CALL    WHEX    ;DBL = 0 UPON RETURN
1069 C3B410             JMP     DHEX3   ;SET BPC AND DATA BYTE
                ;
                DHEX1.  ;PARTIAL BUFFER IN PROGRESS, CHECK FOR SEQUENTIAL BYTE LOAD
106C 2AD081             LHLD    FPC
106F EB                 XCHG
1070 2A2182             LHLD    BPC     ;BASE PC IN H,L
1073 4F                 MOV     C,A     ;CURRENT LENGTH OF BUFFER
1074 0600               MVI     B,0     ;IS IN B,C
1076 09                 DAD     B       ;BPC+DBL TO H,L
1077 7B                 MOV     A,E     ;READY FOR COMPARE
1078 BD                 CMP     L       ;EQUAL?
1079 C28110             JNZ     DHEX2   ;BR IF NOT
107C 7A                 MOV     A,D     ;CHECK HO BYTE
107D BC                 CMP     H
107E CA8A10             JZ      DHEX4   ;BR IF SAME ADDRESS
                ;
                DHEX2.  ;NON SEQUENTIAL ADDRESS, DUMP AND CHANGE BASE ADDRESS
1081 CDB810             CALL    WHEX
                DHEX3.  ;SET NEW BASE
1084 2AD081             LHLD    FPC
1087 222182             SHLD    BPC
                ;
                DHEX4.  ;STORE DATA BYTE AND INC DBL
108A 212382             LXI     H,DBL
108D 5E                 MOV     E,M     ;LENGTH TO REG-E
108E 34                 INR     M       ;DBL=DBL+1
108F 1600               MVI     D,0     ;HIGH ORDER ZERO FOR DOUBLE ADD
1091 212482             LXI     H,DBUFF
1094 19                 DAD     D       ;DBUFF+DBL TO H,L
1095 F1                 POP     PSW     ;RESTORE DATA BYTE
1096 77                 MOV     M,A     ;INTO DATA BUFFER
1097 D1                 POP     D
1098 C1         DHRET.  POP     B       ;ENVIRONMENT RESTORED
1099 C9                 RET
                ;
                WRC.    ;WRITE CHARACTER WITH CHECK SUM IN D
109A F5                 PUSH    PSW
109B 0F                 RRC
109C 0F                 RRC
109D 0F                 RRC
109E 0F                 RRC
109F E60F               ANI     0FH
10A1 CDAF10             CALL    HEXC    ;OUTPUT HEX CHARACTER
10A4 F1                 POP     PSW     ;RESTORE BYTE
10A5 F5                 PUSH    PSW     ;SAVE A VERSION
10A6 E60F               ANI     0FH
10A8 CDAF10             CALL    HEXC    ;WRITE LOW NIBBLE
10AB F1                 POP     PSW     ;RESTORE BYTE
10AC 82                 ADD     D       ;COMPUTE CHECKSUM
10AD 57                 MOV     D,A     ;SAVE CS
```



```
10AE C9                 RET
                ;
                HEXC.   ;WRITE CHARACTER
10AF C690               ADI     90H
10B1 27                 DAA
10B2 CE40               ACI     40H
10B4 27                 DAA
10B5 C3AA8E             JMP     PNB     ;PUT BYTE
                ;
                WHEX.   ;WRITE CURRENT HEX BUFFER
10B8 3E3A               MVI     A,':'   ;RECORD HEADER
10BA CDAA8E             CALL    PNB     ;PUT BYTE
10BD 212382             LXI     H,DBL   ;RECORD LENGTH ADDRESS
10C0 5E                 MOV     E,M     ;LENGTH TO REG-E
10C1 AF                 XRA     A       ;ZERO TO REG-A
10C2 57                 MOV     D,A     ;CLEAR CHECKSUM
10C3 77                 MOV     M,A     ;LENGTH IS ZEROED FOR NEXT WRITE
10C4 2A2182             LHLD    BPC     ;BASE ADDRESS FOR RECORD
10C7 7B                 MOV     A,E     ;LENGTH TO A
10C8 CD9A10             CALL    WRC     ;WRITE HEX VALUE
10CB 7C                 MOV     A,H     ;HIGH ORDER BASE ADDR
10CC CD9A10             CALL    WRC     ;WRITE HO BYTE
10CF 7D                 MOV     A,L     ;LOW ORDER BASE ADDR
10D0 CD9A10             CALL    WRC     ;WRITE LO BYTE
10D3 AF                 XRA     A       ;ZERO TO A
10D4 CD9A10             CALL    WRC     ;WRITE RECORD TYPE 00
10D7 7B                 MOV     A,E     ;CHECK FOR LENGTH 0
10D8 B7                 ORA     A
10D9 CAE810             JZ      WHEX1
                ;
                ;       NON - ZERO, WRITE DATA BYTES
10DC 212482             LXI     H,DBUFF
10DF 7E         WHEX0.  MOV     A,M     ;GET BYTE
10E0 23                 INX     H
10E1 CD9A10             CALL    WRC     ;WRITE DATA BYTE
10E4 1B                 DCR     E       ;END OF BUFFER?
10E5 C2DF10             JNZ     WHEX0
                ;
                ;       END OF DATA BYTES, WRITE CHECK SUM
10E8 AF         WHEX1.  XRA     A
10E9 92                 SUB     D       ;COMPUTE CHECKSUM
10EA CD9A10             CALL    WRC
                ;
                ;       SEND CRLF AT END OF RECORD
10ED 3E0D               MVI     A,CR
10EF CDAA8E             CALL    PNB
10F2 3E0A               MVI     A,LF
10F4 CDAA8E             CALL    PNB
10F7 C9                 RET
                ;
                ;
                ;
1100 =          ENDMOD  EQU     ($ AND 0FFE0H)+20H
10F8                    END
```

```
CP/M MACRO ASSEM 2.0     0014     ASM IO MODULE

0188 ACCLEN     0189 ACCUM      0040 ACMAX      0235 ADISK      01D2 ASPC
0005 BDOS       0000 BOOT       0221 BPC        0080 BUFF       0234 CDISK
0CFA CLOSE      0010 CLOSF      000D CR         0019 CSEL       0223 DBL
0224 DBUFF      0013 DELEF      0D08 DELETE     104C DHEX       106C DHEX1
1081 DHEX2      1084 DHEX3      108A DHEX4      1098 DHRET      103C ENDA
1100 ENDMOD     001A EOF        0F4F EOPR       0F39 EOR        0F57 EOR0
0F67 EOR1       0F3F EOR2       0F77 EORC       0F97 EORHC      0F86 EORPC
1029 ERRCL      0FE3 ERRFN      0FFA ERRFR      1011 ERRFW      0FD0 ERRMA
0F89 ERROP      01C9 EVALUE     005C FCB        0009 FLN        0CD2 FNAMB
0CCD FNAME      0DBB FNERR      0001 FNM        01D0 FPC        0E2B FRERR
0EA1 FWERR      0DC4 GCOMP      0DCA GNC        0DE7 GNC0       0E0D GNC1
0E19 GNC2       0E12 GNCE       099F HBP        09A1 HBUFF      027A HCB
0237 HDISK      10AF HEXC       0300 HSIZE      0CE0 INIT       000A LF
0016 MAKEF      0D10 MAKE       0DFE MOVB       0006 MHB        0D9E NOHEX
0D63 NOPR       0006 NPB        0D2B NPR        0069 NSB        000F OPENF
0CE9 OPEN       01CF PASS       069D PBP        069F PBUFF      0259 PCB
0EDE PCHAR      0CBC PCON       0236 PDISK      0F2F PERR       0EAA PHB
0EB4 PHBF       0E34 PHC        0E53 PHCF       0E4A PHGO       0E51 PHRET
0300 PSIZE      0070 QBMAX      0104 QBP        010C QBUFF      0001 READC
0014 READF      000B REDYC      0298 SBP        029D SBUFF      0238 SCB
0258 SCBCR      0244 SCBR       0CAE SCNP       0CB0 SCNP0      0D21 SELA
000E SELECT     0CA1 SEL        0D38 SELH       0D31 SELP       0D3F SET0
001A SETDM      0DA1 SETUP      0400 SSIZE      01CD SYMAX      01CB SYTOP
0FA0 TITL       0185 TOKEN      0186 VALUE      0EB5 WBUF0      0E7A WBUFF
108B WHEX       10DF WHEX0      10EB WHEX1      0F86 WOB0       0F27 WOB2
0F15 WOBE       0F00 WOBUFF     0EEB WOCHAR     109A WRC        0002 WRITC
0015 WRITF
```

```
CP/M MACRO ASSEM 2.0     0001     ASM SCANNER MODULE

                         TITLE   'ASM SCANNER MODULE'
 1100                    ORG     1100H
 1100 C34013             JMP     ENDMOD  ;END OF THIS MODULE
 1103 C33211             JMP     INITS   ;INITIALIZE THE SCANNER
 1106 C3C011             JMP     SCAN    ;CALL THE SCANNER
                 ;
                 ;
                 ;       ENTRY POINTS IN I/O MODULE
 0200 =          IOMOD   EQU     200H
 0206 =          GNCF    EQU     IOMOD+6H
 0215 =          WOBUFF  EQU     IOMOD+15H
 0218 =          PERR    EQU     IOMOD+18H
                 ;
 1109            LASTC.  DS      1       ;LAST CHAR SCANNED
 110A            NEXTC.  DS      1       ;LOOK AHEAD CHAR
 110B            STYPE.  DS      1       ;RADIX INDICATOR
                 ;
                 ;       COMMON EQUATES
 0070 =          PBMAX   EQU     120     ;MAX PRINT SIZE
 010C =          PBUFF   EQU     10CH    ;PRINT BUFFER
 0104 =          PBP     EQU     PBUFF+PBMAX     ;PRINT BUFFER POINTER
                 ;
 0185 =          TOKEN   EQU     PBP+1   ;CURRENT TOKEN UDER SCAN
 0186 =          VALUE   EQU     TOKEN+1 ;VALUE OF NUMBER IN BINARY
 0188 =          ACCLEN  EQU     VALUE+2 ;ACCUMULATOR LENGTH
 0040 =          ACMAX   EQU     64      ;MAX ACCUMULATOR LENGTH
 0189 =          ACCUM   EQU     ACCLEN+1
                 ;
 01C9 =          EVALUE  EQU     ACCUM+ACMAX     ;VALUE FROM EXPRESSION ANALYSIS
                 ;
 01CB =          SYTOP   EQU     EVALUE+2        ;CURRENT SYMBOL TOP
 01CD =          SYMAX   EQU     SYTOP+2         ;MAX ADDRESS+1
                 ;
 01CF =          PASS    EQU     SYMAX+2 ;CURRENT PASS NUMBER
 01D0 =          FPC     EQU     PASS+1  ;FILL ADDRESS FOR NEXT HEX BYTE
 01D2 =          ASPC    EQU     FPC+2   ;ASSEMBLER'S PSEUDO PC
                 ;
                 ;       GLOBAL EQUATES
 0001 =          IDEN    EQU     1       ;IDENTIFIER
 0002 =          NUMB    EQU     2       ;NUMBER
 0003 =          STRNG   EQU     3       ;STRING
 0004 =          SPECL   EQU     4       ;SPECIAL CHARACTER
                 ;
 0001 =          PLABT   EQU     00001B  ;PROGRAM LABEL
 0002 =          DLABT   EQU     00010B  ;DATA LABEL
 0004 =          EQUT    EQU     00100B  ;EQUATE
 0005 =          SETT    EQU     00101B  ;SET
 0006 =          MACT    EQU     00110B  ;MACRO
                 ;
 0008 =          EXTT    EQU     01000B  ;EXTERNAL
 0008 =          REFT    EQU     01011B  ;REFER
 000C =          GLBT    EQU     01100B  ;GLOBAL
                 ;
 0002 =          BINV    EQU     2
 0008 =          OCTV    EQU     8
 000A =          DECV    EQU     10
```

```
CP/M MACRO ASSEM 2.0     #002     ASM SCANNER MODULE

 0010 =          MEXV    EQU     16
 000D =          CR      EQU     0DH
 000A =          LF      EQU     0AH
 001A =          EOF     EQU     1AH
 0009 =          TAB     EQU     09H     ;TAB CHARACTER
                 ;
                 ;
                 ;       UTILITY SUBROUTINES
                 GNC.    ;GET NEXT CHARACTER AND ECHO TO PRINT FILE
 110C CD0602             CALL    GNCF
 110F F5                 PUSH    PSW
 1110 FE0D               CPI     CR
 1112 CA3011             JZ      GNC0
 1115 FE0A               CPI     LF      ;IF LF THEN DUMP CURRENT BUFFER
 1117 CA3011             JZ      GNC0
                 ;
                 ;NOT A CR OR LF, PLACE INTO BUFFER IF THERE IS ENOUGH ROOM
 111A 3A8401             LDA     PBP
 111D FE78               CPI     PBMAX
 111F D23011             JNC     GNC0
                 ;       ENOUGH ROOM, PLACE INTO BUFFER
 1122 5F                 MOV     E,A
 1123 1600               MVI     D,0     ;DOUBLE PRECISION PBP IN D,E
 1125 3C                 INR     A
 1126 328401             STA     PBP     ;INCREMENTED PBP IN MEMORY
 1129 210C01             LXI     H,PBUFF
 112C 19                 DAD     D       ;PBUFF(PBP)
 112D F1                 POP     PSW
 112E 77                 MOV     M,A     ;PBUFF(PBP) = CHAR
 112F C9                 RET
                 GNC0.   ;CHAR NOT PLACED INTO BUFFER
 1130 F1                 POP     PSW
 1131 C9                 RET
                 ;
                 INITS.  ;INITIALIZE THE SCANNER
 1132 CD4911             CALL    ZERO
 1135 328A11             STA     NEXTC   ;CLEAR NEXT CHARACTER
 1138 328401             STA     PBP
 113B 3E0A               MVI     A,LF    ;SET LAST CHAR TO LF
 113D 320911             STA     LASTC
 1140 CD1502             CALL    WDBUFF  ;CLEAR BUFFER
 1143 3E10               MVI     A,16    ;START OF PRINT LINE
 1145 328401             STA     PBP
 1148 C9                 RET
                 ;
 1149 AF         ZERO.   XRA     A
 114A 320B01             STA     ACCLEN
 114D 328B11             STA     STYPE
 1150 C9                 RET
                 ;
                 SAVER.  ;STORE THE NEXT CHARACTER INTO THE ACCUMULATOR AND UPDATE AC
 1151 210B01             LXI     H,ACCLEN
 1154 7E                 MOV     A,M
 1155 FE40               CPI     ACMAX
 1157 DA5F11             JC      SAV1    ;JUMP IF NOT UP TO LAST POSITION
 115A 3600               MVI     M,0
```



```
CP/M MACRO ASSEM 2.0     #003     ASM SCANNER MODULE

 115C CD1E13             CALL    ERRO
 115F 5E         SAV1.   MOV     E,M     ;D,E WILL HOLD INDEX
 1160 1600               MVI     D,0
 1162 34                 INR     M       ;ACCLEN INCREMENTED
 1163 23                 INX     H       ;ADDRESS ACCUMULATOR
 1164 19                 DAD     D       ;ADD INDEX TO ACCUMULATOR
 1165 3A8A11             LDA     NEXTC   ;GET CHARACTER
 1168 77                 MOV     M,A     ;INTO ACCUMULATOR
 1169 C9                 RET
                 ;
                 TDOLL.  ;TEST FOR DOLLAR SIGN, ASSUMING H,L ADDRESS NEXTC
 116A 7E                 MOV     A,M
 116B FE24               CPI     '$'
 116D C0                 RNZ
 116E AF                 XRA     A       ;TO GET A ZERO
 116F 77                 MOV     M,A     ;CLEARS NEXTC
 1170 C9                 RET             ;WITH ZERO FLAG SET
                 ;
                 NUMERIC.        ;CHECK NEXTC FOR NUMERIC, RETURN ZERO FLAG IF NOT
 1171 3A8A11             LDA     NEXTC
 1174 D630               SUI     '0'
 1176 FE0A               CPI     10
                 ;       CARRY RESET IF NUMERIC
 1178 17                 RAL
 1179 E601               ANI     1B      ;ZERO IF NOT NUMERIC
 117B C9                 RET
                 ;
                 HEX.    ;RETURN ZERO FLAG IF NEXTC IS NOT HEXADECIMAL
 117C CD7111             CALL    NUMERIC
 117F C0                 RNZ             ;RETURNS IF 0-9
 1180 3A8A11             LDA     NEXTC
 1183 D641               SUI     'A'
 1185 FE06               CPI     6
                 ;       CARRY SET IF OUT OF RANGE
 1187 17                 RAL
 1188 E601               ANI     1B
 118A C9                 RET
                 ;
                 LETTER. ;RETURN ZERO FLAG IF NEXTC IS NOT A LETTER
 118B 3A8A11             LDA     NEXTC
 118E D641               SUI     'A'
 1190 FE1A               CPI     26
 1192 17                 RAL
 1193 E601               ANI     1B
 1195 C9                 RET
                 ;
                 ALNUM.  ;RETURN ZERO FLAG IF NOT ALPHANUMERIC
 1196 CD8B11             CALL    LETTER
 1199 C0                 RNZ
 119A CD7111             CALL    NUMERIC
 119D C9                 RET
                 ;
                 TRANS.  ;TRANSLATE TO UPPER CASE
 119E 3A8A11             LDA     NEXTC
 11A1 FE61               CPI     'A' OR 1100000B ;LOWER CASE A
 11A3 D8                 RC                      ;CARRY IF LESS THAN LOWE
```

```
11A4 FE7B              CPI     ('Z' OR 1100000B)+1     ;LOWER CASE Z
11A6 D0                RNC                             ;NO CARRY IF GREATER THAN L
11A7 E65F              ANI     1011111B                ;CONVERT TO UPPER CASE
11A9 320A11            STA     NEXTC
11AC C9                RET
                ;
                GNCH:   ;GET CHARACTER AND STORE TO NEXTC
11AD CD8C11            CALL    GNC
11B0 320A11            STA     NEXTC
11B3 CD9E11            CALL    TRANS   ;TRANSLATE TO UPPER CASE
11B6 C9                RET
                ;
                EOLT:   ;END OF LINE TEST FOR COMMENT SCAN
11B7 FE0D              CPI     CR
11B9 C8                RZ
11BA FE1A              CPI     EOF
11BC C8                RZ
11BD FE21              CPI     '!'
11BF C9                RET
                ;
                SCAN:   ;FIND NEXT TOKEN IN INPUT STREAM
11C0 AF                XRA     A
11C1 328501            STA     TOKEN
11C4 CD4911            CALL    ZERO
                ;
                ;       DEBLANK
11C7 3A0A11     DEBL:   LDA     NEXTC
11CA FE09              CPI     TAB     ;TAB CHARACTER TREATED AS BLANK OUTSIDE STR
11CC CAF411            JZ      DEB0
11CF FE3B              CPI     ';'     ;MAY BE A COMMENT
11D1 CAE111            JZ      DEB1    ;DEBLANK THROUGH COMMENT
11D4 FE2A              CPI     '*'     ;PROCESSOR TECH COMMENT
11D6 C2ED11            JNZ     DEB2    ;NOT *
11D9 3A0911            LDA     LASTC
11DC FE0A              CPI     LF      ;LAST LINE FEED?
11DE C2ED11            JNZ     DEB2    ;NOT LF*
                ;       COMMENT FOUND, REMOVE IT
11E1 CDAD11     DEB1:   CALL    GNCH
11E4 CDB711            CALL    EOLT    ;CR, EOF, OR !
11E7 CAFA11            JZ      FINDL   ;HANDLE END OF LINE
11EA C3E111            JMP     DEB1    ;OTHERWISE CONTINUE SCAN
11ED F620       DEB2:   ORI     ' '     ;MAY BE ZERO
11EF FE20              CPI     ' '
11F1 C2FA11            JNZ     FINDL
11F4 CDAD11     DEB0:   CALL    GNCH    ;GET NEXT AND STORE TO NEXTC
11F7 C3C711            JMP     DEBL
                ;
                ;       LINE DEBLANKED, FIND TOKEN TYPE
                FINDL:  ;LOOK FOR LETTER, DECIMAL DIGIT, OR STRING QUOTE
11FA CD8B11            CALL    LETTER
11FD CA0512            JZ      FIND0
1200 3E01              MVI     A,IDEN
1202 C33912            JMP     STOKEN
                ;
1205 CD7111     FIND0:  CALL    NUMERIC
1208 CA1012            JZ      FIND1
```



```
120B 3E02              MVI     A,NUMB
120D C33912            JMP     STOKEN
                ;
1210 3A0A11     FIND1:  LDA     NEXTC
1213 FE27              CPI     ''''
1215 C22112            JNZ     FIND2
1218 AF                XRA     A
1219 320A11            STA     NEXTC   ;DON'T STORE THE QUOTE
121C 3E03              MVI     A,STRNG
121E C33912            JMP     STOKEN
                ;
                FIND2:  ;ASSUME IT IS A SPECIAL CHARACTER
1221 FE0A              CPI     LF      ;IF LF THEN DUMP THE BUFFER
1223 C23712            JNZ     FIND3
                ;       LF FOUND
1226 3ACF01            LDA     PASS
1229 B7                ORA     A
122A C41502            CNZ     WOBUFF
122D 21CC01            LXI     H,PBUFF ;CLEAR ERROR CHAR ON BOTH PASSES
1230 3620              MVI     M,' '
1232 3E10              MVI     A,16
1234 32D401            STA     PBP     ;START NEW LINE
1237 3E04       FIND3:  MVI     A,SPECL
                ;
1239 328501     STOKEN: STA     TOKEN
                ;
                ;
                ;       LOOP WHILE CURRENT ITEM IS ACCUMULATING
123C 3A0A11     SCTOK:  LDA     NEXTC
123F 320911            STA     LASTC   ;SAVE LAST CHARACTER
1242 B7                ORA     A
1243 C45111            CNZ     SAVER   ;STORE CHARACTER INTO ACCUM IF NOT ZERO
1246 CDAD11            CALL    GNCH    ;GET NEXT TO NEXTC
1249 3A8501            LDA     TOKEN
124C FE04              CPI     SPECL
124E C8                RZ              ;RETURN IF SPECIAL CHARACTER
124F FE03              CPI     STRNG
1251 C49E11            CNZ     TRANS   ;TRANSLATE TO UPPER CASE IF NOT IN STRING
1254 210A11            LXI     H,NEXTC
1257 3A8501            LDA     TOKEN
                ;
125A FE01              CPI     IDEN
125C C26C12            JNZ     SCT2
                ;
                ;       ACCUMULATING AN IDENTIFIER
125F CD6A11            CALL    TDOLL   ;$?
1262 CA3C12            JZ      SCTOK   ;IF SO, SKIP T
1265 CD9611            CALL    ALNUM   ;ALPHA NUMERIC?
1268 C8                RZ              ;RETURN IF EN
                ;       NOT END OF THE IDENTIFIER
1269 C33C12            JMP     SCTOK
                ;
                SCT2:   ;NOT SPECIAL OR IDENT, CHECK NUMBER
126C FE02              CPI     NUMB
126E C28213            JNZ     SCT3
                ;
```

```
CP/M MACRO ASSEM 2.0     #006     ASM SCANNER MODULE

                ;       ACCUMULATING A NUMBER, CHECK FOR $
1271 CD6A11             CALL    TDOLL
1274 CA3C12             JZ      SCTOK   ;SKIP IF FOUND
1277 CD7C11             CALL    HEX     ;HEX CHARACTER?
127A C23C12             JNZ     SCTOK   ;STORE IT IF FOUND
                ;       END OF NUMBER, LOOK FOR RADIX INDICATOR
                ;
127D 3A8A11             LDA     NEXTC
1280 FE4F               CPI     'O'     ;OCTAL INDICATOR
1282 CA8A12             JZ      NOCT
1285 FE51               CPI     'Q'     ;OCTAL IND[ICATOR
1287 C28F12             JNZ     NUM2
                ;
                NOCT:   ;OCTAL
128A 3E08               MVI     A,OCTV
128C C39612             JMP     SSTYP
                ;
128F FE48       NUM2:   CPI     'H'
1291 C2A012             JNZ     NUM3
1294 3E10               MVI     A,HEXV
1296 320B11     SSTYP:  STA     STYPE
1299 AF                 XRA     A
129A 320A11             STA     NEXTC   ;CLEARS THE LOOKAHEAD CHARACTER
129D C3B812             JMP     NCON
                ;
                ;       RADIX MUST COME FROM ACCUM
12A0 3A0911     NUM3:   LDA     LASTC
12A3 FE42               CPI     'B'
12A5 C2AD12             JNZ     NUM4
12A8 3E02               MVI     A,BINV
12AA C3B412             JMP     SSTY1
                ;
12AD FE44       NUM4:   CPI     'D'
12AF 3E0A               MVI     A,DECV
12B1 C2B812             JNZ     SSTY2
12B4 218801     SSTY1:  LXI     H,ACCLEN
12B7 35                 DCR     M       ;ACCLEN DECREMENTED TO REMOVE RADIX INDICAT
12B8 320B11     SSTY2:  STA     STYPE
                ;
                NCON:   ;NUMERIC CONVERSION OCCURS HERE
12BB 210000             LXI     H,0
12BE 228601             SHLD    VALUE   ;VALUE ACCUMULATES BINARY EQUIVALENT
12C1 218801             LXI     H,ACCLEN
12C4 4E                 MOV     C,M     ;C=ACCLEN
12C5 23                 INX     H       ;ADDRESSES ACCUM
                CLOP:   ;NEXT DIGIT IS PROCESSED HERE
12C6 7E                 MOV     A,M
12C7 23                 INX     H       ;READY FOR NEXT LOOP
12C8 FE41               CPI     'A'
12CA D2D212             JNC     CLOP1   ;NOT HEX A-F
12CD D630               SUI     '0'     ;NORMALIZE
12CF C3D412             JMP     CLOP2
                ;
                CLOP1:  ;HEX A-F
12D2 D637               SUI     'A'-10
                CLOP2:  ;CHECK SIZE AGAINST RADIX
```



```
CP/M MACRO ASSEM 2.0     #007     ASM SCANNER MODULE

12D4 E5                 PUSH    H       ;SAVE ACCUM ADDR
12D5 C5                 PUSH    B       ;SAVE CURRENT POSITION
12D6 4F                 MOV     C,A
12D7 210B11             LXI     H,STYPE
12DA BE                 CMP     M
12DB D41013             CNC     ERRV    ;VALUE ERROR IF DIGIT>=RADIX
12DE 0600               MVI     B,0     ;DOUBLE PRECISION DIGIT
12E0 7E                 MOV     A,M     ;RADIX TO ACCUMULATOR
12E1 2A8601             LHLD    VALUE
12E4 EB                 XCHG            ;VALUE TO D,E - ACCUMULATE RESULT IN H,L
12E5 210000             LXI     H,0     ;ZERO ACCUMULATOR
                CLOP3:  ;LOOP UNTIL RADIX GOES TO ZERO
12E8 B7                 ORA     A
12E9 CAF712             JZ      CLOP4
12EC 1F                 RAR             ;TEST LSB
12ED D2F112             JNC     TTWO    ;SKIP SUMMING OPERATION IF LSB=0
12F0 19                 DAD     D       ;ADD IN VALUE
                TTWO:   ;MULTIPLY VALUE * 2 FOR SHL OPERATION
12F1 EB                 XCHG
12F2 29                 DAD     H
12F3 EB                 XCHG
12F4 C3E812             JMP     CLOP3
                ;
                ;
                CLOP4:  ;END OF NUMBER CONVERSION
12F7 09                 DAD     B       ;DIGIT ADDED IN
12F8 228601             SHLD    VALUE
12FB C1                 POP     B
12FC E1                 POP     H
12FD 0D                 DCR     C       ;MORE DIGITS?
12FE C2C612             JNZ     CLOP
1301 C9                 RET             ;DONE WITH THE NUMBER
                ;
                SCT3:   ;MUST BE A STRING
1302 3A8A11             LDA     NEXTC
1305 FE0D               CPI     CR      ;END OF LINE?
1307 CA1E13             JZ      ERR0    ;AND RETURN
130A FE27               CPI     ''''
130C C23C12             JNZ     SCTOK
130F CDAD11             CALL    GNCH
1312 FE27               CPI     ''''
1314 C0                 RNZ             ;RETURN IF SINGLE QUOTE ENCOUNTERED
1315 C33C12             JMP     SCTOK   ;OTHERWISE TREAT AS ONE QUOTE
                ;
                ;       END OF SCANNER
                ;
                ;       ERROR MESSAGE ROUTINES
                ERRV:   ;'V' VALUE ERROR
1318 F5                 PUSH    PSW
1319 3E56               MVI     A,'V'
131B C32413             JMP     ERR
                ;
                ERR0:   ;'0' OVERFLOW ERROR
131E F5                 PUSH    PSW
131F 3E4F               MVI     A,'O'
1321 C32413             JMP     ERR
```

```
CP/M MACRO ASSEM 2.0     0000     ASM SCANNER MODULE

                ;
                ERR.    ;PRINT ERROR MESSAGE
1324 C5                 PUSH    B
1325 E5                 PUSH    H
1326 CD1802             CALL    PERR
1329 E1                 POP     H
132A C1                 POP     B
132B F1                 POP     PSW
132C C9                 RET
                ;
1340 =          ENDMOD  EQU     ($ AND 0FFE0H) + 20H
132D                    END
```



```
CP/M MACRO ASSEM 2.0     0009     ASM SCANNER MODULE

0188 ACCLEN     0189 ACCUM      0048 ACMAX      1196 ALNUM      01D2 ASPC
0002 BINV       12C6 CLOP       12D2 CLOP1      12D4 CLOP2      12E8 CLOP3
12F7 CLOP4      000D CR         11F4 DEB0       11E1 DEB1       11ED DEB2
11C7 DEBL       000A DECV       0002 DLABT      1340 ENDMOD     001A EOF
11B7 EOLT       0004 EQUT       131E ERR0       1324 ERR        1310 ERRV
01C9 EVALUE     0008 EXTT       1205 FIND0      1218 FIND1      1221 FIND2
1237 FIND3      11FA FINDL      01D8 FPC        000C GLBT       1136 GNC0
0206 GNCF       110C GNC        11AB GNCN       117C HEX        0010 HEXV
0001 IDEN       1132 INITS      0200 IOMOD      1109 LASTC      11BB LETTER
000A LF         0006 MACT       12BB NCON       110A NEXTC      120A NOCT
128F NUM2       12A0 NUM3       12AD NUM4       0002 NUMB       1171 NUMERIC
000B OCTV       01CF PASS       0078 PBMAX      01B4 PBP        010C PBUFF
0218 PERR       0001 PLABT      0008 REFT       115F SAVI       1151 SAVER
11C0 SCAN       126C SCT2       1302 SCT3       123C SCTOK      0005 SETT
0004 SPECL      12B4 SSTYI      12B0 SSTY2      1296 SSTYP      1239 STOKEN
0003 STRNG      110B STYPE      01CD SYMAX      01CB SYTOP      0009 TAB
116A TDOLL      0185 TOKEN      119E TRANS      12F1 TTWO       0186 VALUE
0215 WOBUFF     1149 ZERO
```

```
CP/M MACRO ASSEM 2.0     0001     ASM SYMBOL TABLE MODULE

                        TITLE   'ASM SYMBOL TABLE MODULE'
                ;       SYMBOL TABLE MANIPULATION MODULE
                ;
1340                    ORG     1340H
0200 =          IOMOD   EQU     200H    ;IO MODULE ENTRY POINT
0212 =          PCON    EQU     IOMOD+12H
021E =          EOR     EQU     IOMOD+1EH
                ;
                ;
                ;       ENTRY POINTS TO SYMBOL TABLE MODULE
1340 C3A015             JMP     ENDMOD
1343 C35C14             JMP     INISY
1346 C39E14             JMP     LOOKUP
1349 C39814             JMP     FOUND
134C C3EB14             JMP     ENTER
134F C36015             JMP     SETTY
1352 C37215             JMP     GETTY
1355 C38D15             JMP     SETVAL
1358 C39615             JMP     GETVAL
                ;
                ;       COMMON EQUATES
0078 =          PBMAX   EQU     120     ;MAX PRINT SIZE
010C =          PBUFF   EQU     10CH    ;PRINT BUFFER
0184 =          PBP     EQU     PBUFF+PBMAX     ;PRINT BUFFER POINTER
                ;
0185 =          TOKEN   EQU     PBP+1   ;CURRENT TOKEN UDER SCAN
0186 =          VALUE   EQU     TOKEN+1 ;VALUE OF NUMBER IN BINARY
0188 =          ACCLEN  EQU     VALUE+2 ;ACCUMULATOR LENGTH
0040 =          ACMAX   EQU     64      ;MAX ACCUMULATOR LENGTH
0189 =          ACCUM   EQU     ACCLEN+1
                ;
01C9 =          EVALUE  EQU     ACCUM+ACMAX     ;VALUE FROM EXPRESSION ANALYSIS
                ;
01CB =          SYTOP   EQU     EVALUE+2        ;CURRENT SYMBOL TOP
01CD =          SYMAX   EQU     SYTOP+2         ;MAX ADDRESS+1
                ;
01CF =          PASS    EQU     SYMAX+2 ;CURRENT PASS NUMBER
01D0 =          FPC     EQU     PASS+1  ;FILL ADDRESS FOR NEXT HEX BYTE
01D2 =          ASPC    EQU     FPC+2   ;ASSEMBLER'S PSEUDO PC
01D4 =          SYBAS   EQU     ASPC+2  ;BASE OF SYMBOL TABLE
01D6 =          SYADR   EQU     SYBAS+2 ;CURRENT SYMBOL BEING ACCESSED
                ;
                ;       GLOBAL EQUATES
0001 =          IDEN    EQU     1       ;IDENTIFIER
0002 =          NUMB    EQU     2       ;NUMBER
0003 =          STRNG   EQU     3       ;STRING
0004 =          SPECL   EQU     4       ;SPECIAL CHARACTER
                ;
0001 =          PLABT   EQU     0001B   ;PROGRAM LABEL
0002 =          DLABT   EQU     0010B   ;DATA LABEL
0004 =          EQUT    EQU     0100B   ;EQUATE
0005 =          SETT    EQU     0101B   ;SET
0006 =          MACT    EQU     0110B   ;MACRO
                ;
0008 =          EXTT    EQU     1000B   ;EXTERNAL
000B =          REFT    EQU     1011B   ;REFER
```

```
CP/M MACRO ASSEM 2.0    0002    ASM SYMBOL TABLE MODULE

000C =          GLBT    EQU     11000B  ;GLOBAL
                ;
                ;
000D =          CR      EQU     0DH
                ;
                ;       DATA AREAS
                ;       SYMBOL TABLE BEGINS AT THE END OF THIS MODULE
0005 =          FIXD    EQU     5       ;5 BYTES OVERHEAD WITH EACH SYMBOL ENTRY
                ;                       2BY COLLISION, 1BY TYPE/LEN, 2BY VALUE
0080 =          HSIZE   EQU     128     ;HASH TABLE SIZE
007F =          HMASK   EQU     HSIZE-1 ;HASH MASK FOR CODING
135B            HASHT:  DS      HSIZE*2 ;HASH TABLE
145B            HASHC:  DS      1       ;HASH CODE AFTER CALL ON LOOKUP
                ;
                ;       SYMBOL TABLE ENTRY FORMAT IS
                ;               -----------------
                ;               | HIGH VAL BYTE |
                ;               -----------------
                ;               | LOW  VAL BYTE |
                ;               -----------------
                ;               | CHARACTER N   |
                ;               -----------------
                ;               |     ...       |
                ;               -----------------
                ;               | CHARACTER 1   |
                ;               -----------------
                ;               | TYPE  | LENG  |
                ;               -----------------
                ;               | HIGH COLLISION|
                ;               -----------------
                ;       SYADR=  | LOW COLLISION |
                ;               -----------------
                ;
                ;       WHERE THE LOW/HIGH COLLISION FIELD ADDRESSES ANOTHER ENTRY
                ;       THE SAME HASH CODE (OR ZERO IF THE END OF CHAIN), TYPE DES
                ;       THE ENTRY TYPE (GIVEN BELOW), LENG IS THE NUMBER OF CHARAC
                ;       THE SYMBOL PRINTNAME -1 (I.E., LENG=0 IS A SINGLE CHARACTE
                ;       NAME, WHILE LENG=15 INDICATES A 16 CHARACTER NAME).  CHARA
                ;       THROUGH N GIVE THE PRINTNAME CHARACTERS IN ASCII UPPER CAS
                ;       LOWER CASE NAMES ARE TRANSLATED ON INPUT), AND THE LOW/HIG
                ;       GIVE THE PARTICULAR ADDRESS OR CONSTANT VALUE ASSOCIATED W
                ;       NAME.  THE REPRESENTATION OF MACROS DIFFERS IN THE FIELDS
                ;       FOLLOW THE VALUE FIELD (MACROS ARE NOT CURRENTLY IMPLEMENT
                ;
                ;       THE TYPE FIELD CONSISTS OF FOUR BITS WHICH ARE ASSIGNED AS
                ;       FOLLOWS.
                ;
                ;               0000    UNDEFINED SYMBOL
                ;               0001    LOCAL   LABELLED PROGRAM
                ;               0010    LOCAL   LABELLED DATA
                ;               0011    (UNUSED)
                ;               0100            EQUATE
                ;               0101            SET
                ;               0110            MACRO
                ;               0111            (UNUSED)
                ;
```



```
CP/M MACRO ASSEM 2.0    0003    ASM SYMBOL TABLE MODULE

                ;               1000            (UNUSED)
                ;               1001    EXTERN  LABELLED PROGRAM
                ;               1010    EXTERN  LABELLED DATA
                ;               1011            REFERENCE TO MODULE
                ;               1100            (UNUSED)
                ;               1101    GLOBAL  UNDEFINED SYMBOL
                ;               1110    GLOBAL  LABELLED PROGRAM
                ;               1111            (UNUSED)
                ;
                ;       TYPE DEFINITIONS
                ;
0001 =          PLABT   EQU     0001B   ;PROGRAM LABEL
0002 =          DLABT   EQU     0010B   ;DATA LABEL
0004 =          EQUT    EQU     0100B   ;EQUATE
0005 =          SETT    EQU     0101B   ;SET
0006 =          MACT    EQU     0110B   ;MACRO
                ;
0008 =          EXTT    EQU     1000B   ;EXTERNAL ATTRIBUTE
0009 =          REFT    EQU     1001B   ;REFER
000C =          GLBT    EQU     1100B   ;GLOBAL ATTRIBUTE
                ;
                ;
                INISY:  ;INITIALIZE THE SYMBOL TABLE
145C 215B13             LXI     H,HASHT ;ZERO THE HASH TABLE
145F 0680               MVI     B,HSIZE
1461 AF                 XRA     A       ;CLEAR ACCUM
                INI0:
1462 77                 MOV     M,A
1463 23                 INX     H
1464 77                 MOV     M,A     ;CLEAR DOUBLE WORD
1465 23                 INX     H
1466 05                 DCR     B
1467 C26214             JNZ     INI0
                ;
                ;       SET SYMBOL TABLE POINTERS
146A 210000             LXI     H,0
146D 22D601             SHLD    SYADR
                ;
1470 C9                 RET
                ;
                CHASH:  ;COMPUTE HASH CODE FOR CURRENT ACCUMULATOR
1471 21B801             LXI     H,ACCLEN
1474 46                 MOV     B,M     ;GET ACCUM LENGTH
1475 AF                 XRA     A       ;CLEAR ACCUMULATOR
1476 23         CH0:    INX     H       ;MOVE TO FIRST/NEXT CHARACTER POSITION
1477 86                 ADD     M       ;ADD WITH OVERFLOW
1478 05                 DCR     B
1479 C27614             JNZ     CH0
147C E67F               ANI     HMASK   ;MASK BITS FOR MODULO HZISE
147E 325B14             STA     HASHC   ;FILL HASHC WITH RESULT
1481 C9                 RET
                ;
                SETLN:  ;SET THE LENGTH FIELD OF THE CURRENT SYMBOL
1482 47                 MOV     B,A     ;SAVE LENGTH IN B
1483 2AD601             LHLD    SYADR
1486 23                 INX     H
```

```
1487 23              INX     H
1488 7E              MOV     A,M     ;GET TYPE/LENGTH FIELD
1489 E6F0            ANI     0F0H    ;MASK OUT TYPE FIELD
148B B0              ORA     B       ;MASK IN LENGTH
148C 77              MOV     M,A
148D C9              RET
                     ;
                     GETLN.  ;GET THE LENGTH FIELD TO REG-A
148E 2AD601          LHLD    SYADR
1491 23              INX     H
1492 23              INX     H
1493 7E              MOV     A,M
1494 E60F            ANI     0FH
1496 3C              INR     A       ;LENGTH IS STORED AS VALUE - 1
1497 C9              RET
                     ;
                     FOUND.  ;FOUND RETURNS TRUE IF SYADR IS NOT ZERO (TRUE IS NZ FLAG
1498 2AD601          LHLD    SYADR
149B 7D              MOV     A,L
149C B4              ORA     H
149D C9              RET
                     ;
                     LOOKUP. ;LOOK FOR SYMBOL IN ACCUMULATOR
149E CD7114          CALL    CHASH   ;COMPUTE HASH CODE
                     ;       NORMALIZE IDENTIFIER TO 16 CHARACTERS
14A1 21B0B1          LXI     H,ACCLEN
14A4 7E              MOV     A,M
14A5 FE11            CPI     17
14A7 DAAC14          JC      LENOK
14AA 3610            MVI     M,16
                     LENOK.
                     ;       LOOK FOR SYMBOL THROUGH HASH TABLE
14AC 21B814          LXI     H,HASHC
14AF 5E              MOV     E,M
14B0 1600            MVI     D,0     ;DOUBLE HASH CODE IN D,E
14B2 21B813          LXI     H,HASHT ;BASE OF HASH TABLE
14B5 19              DAD     D
14B6 19              DAD     D       ;HASHT(HASHC)
14B7 5E              MOV     E,M     ;LOW ORDER ADDRESS
14B8 23              INX     H
14B9 66              MOV     H,M
14BA 6B              MOV     L,E     ;HEADER TO LIST OF SYMBOLS IS IN H,L
14BB 22D601  LOOK0.  SHLD    SYADR
14BE CD9814          CALL    FOUND
14C1 C8              RZ              ;RETURN IF SYADR BECOMES ZERO
                     ;
                     ;       OTHERWISE EXAMINE CHARACTER STRING FOR MATCH
14C2 CDBE14          CALL    GETLN   ;GET LENGTH TO REG-A
14C5 21B0B1          LXI     H,ACCLEN
14C8 BE              CMP     M
14C9 C2E114          JNZ     LCOMP
                     ;
                     ;       LENGTH MATCH, TRY TO MATCH CHARACTERS
14CC 47              MOV     B,A     ;STRING LENGTH IN B
14CD 23              INX     H       ;HL ADDRESSES ACCUM
14CE EB              XCHG            ;TO D,E
```



```
14CF 2AD601          LHLD    SYADR
14D2 23              INX     H
14D3 23              INX     H
14D4 23              INX     H       ;ADDRESSES CHARACTERS
14D5 1A      LOOK1.  LDAX    D       ;NEXT CHARACTER FROM ACCUM
14D6 BE              CMP     M       ;NEXT CHARACTER IN SYMBOL TABLE
14D7 C2E114          JNZ     LCOMP
                     ;       CHARACTER MATCHED, INCREMENT TO NEXT
14DA 13              INX     D
14DB 23              INX     H
14DC 05              DCR     B
14DD C2D514          JNZ     LOOK1
                     ;
                     ;       COMPLETE MATCH AT CURRENT SYMBOL, SYADR IS SET
14E0 C9              RET
                     ;
                     LCOMP.  ;NOT FOUND, MOVE SYADR DOWN ONE COLLISION ADDRESS
14E1 2AD601          LHLD    SYADR
14E4 5E              MOV     E,M
14E5 23              INX     H
14E6 56              MOV     D,M     ;COLLISION ADDRESS IN D,E
14E7 EB              XCHG
14E8 C3BB14          JMP     LOOK0
                     ;
                     ;
                     ENTER.  ;ENTER SYMBOL IN ACCUMULATOR
                     ;       ENSURE THERE IS ENOUGH SPACE IN THE TABLE
14EB 21B0B1          LXI     H,ACCLEN
14EE 5E              MOV     E,M
14EF 1600            MVI     D,0     ;DOUBLE PRECISION ACCLEN IN D,E
14F1 2ACB01          LHLD    SYTOP
14F4 22D601          SHLD    SYADR   ;NEXT SYMBOL LOCATION
14F7 19              DAD     D       ;SYTOP+ACCLEN
14F8 11B5B0          LXI     D,FIXD  ;FIXED DATA/SYMBOL
14FB 19              DAD     D       ;HL HAS NEXT TABLE LOCATION FOR SYMBOL
14FC EB              XCHG            ;NEW SYTOP IN D,E
14FD 2ACD01          LHLD    SYMAX   ;MAXIMUM SYMTOP VALUE
1500 7B              MOV     A,E
1501 95              SUB     L       ;COMPUTE 16-BIT DIFFERENCE
1502 7A              MOV     A,D
1503 9C              SBB     H
1504 EB              XCHG            ;NEW SYTOP IN H,L
1505 D21115          JNC     OVERER  ;OVERFLOW IN TABLE
                     ;
                     ;       OTHERWISE NO ERROR
1508 22CB01          SHLD    SYTOP   ;SET NEW TABLE TOP
150B 2AD601          LHLD    SYADR   ;SET COLLISION FIELD
150E EB              XCHG            ;CURRENT SYMBOL ADDRESS TO D,E
150F 21B814          LXI     H,HASHC ;HASH CODE FOR CURRENT SYMBOL TO H,L
1512 4E              MOV     C,M     ;LOW BYTE
1513 0600            MVI     B,0     ;DOUBLE PRECISION VALUE IN B,C
1515 21B813          LXI     H,HASHT ;BASE OF HASH TABLE
1518 09              DAD     B
1519 09              DAD     B       ;HASHT(HASHC) IN H,L
                     ;       D,E ADDRESSES CURRENT SYMBOL - CHANGE LINKS
151A 4E              MOV     C,M     ;LOW ORDER OLD HEADER
```

```
151B 23                 INX     H
151C 46                 MOV     B,M     ;HIGH ORDER OLD HEADER
151D 72                 MOV     M,D     ;HIGH ORDER NEW HEADER TO HASH TABLE
151E 2B                 DCX     H
151F 73                 MOV     M,E     ;LOW ORDER NEW HEADER TO HASH TABLE
1520 EB                 XCHG            ;H,L HOLDS SYMBOL TABLE ADDRESS
1521 71                 MOV     M,C     ;LOW ORDER OLD HEADER TO COLLISION FIELD
1522 23                 INX     H
1523 70                 MOV     M,B     ;HIGH ORDER OLD HEADER TO COLLISION FIELD
                ;
                ;       HASH CHAIN NOW REPAIRED FOR THIS ENTRY, COPY THE PRINTNAME
1524 118001             LXI     D,ACCLEN
1527 1A                 LDAX    D       ;GET SYMBOL LENGTH
1528 FE11               CPI     17      ;LARGER THAN 16 SYMBOLS?
152A DA2F15             JC      ENT1
152D 3E10               MVI     A,16    ;TRUNCATE TO 16 CHARACTERS
                ;       COPY LENGTH FIELD, FOLLOWED BY PRINTNAME CHARACTERS
152F 47         ENT1:   MOV     B,A     ;COPY LENGTH TO B
1530 3D                 DCR     A       ;1-16 CHANGED TO 0-15
1531 23                 INX     H       ;FOLLOWING COLLISION FIELD
1532 77                 MOV     M,A     ;STORE LENGTH WITH UNDEFINED TYPE (0000)
1533 23         ENT2:   INX     H
1534 13                 INX     D
1535 1A                 LDAX    D
1536 77                 MOV     M,A     ;STORE NEXT CHARACTER OF PRINTNAME
1537 05                 DCR     B       ;LENGTH=LENGTH-1
1538 C23315             JNZ     ENT2    ;FOR ANOTHER CHARACTER
                ;
                ;       PRINTNAME COPIED, ZERO THE VALUE FIELD
153B AF                 XRA     A       ;ZERO A
153C 23                 INX     H       ;LOW ORDER VALUE
153D 77                 MOV     M,A
153E 23                 INX     H
153F 77                 MOV     M,A     ;HIGH ORDER VALUE
1540 C9                 RET
                ;
                OVERER: ;OVERFLOW IN SYMBOL TABLE
1541 214A15             LXI     H,ERRO
1544 CD1202             CALL    PCON
1547 C31E02             JMP     EOR     ;END OF EXECUTION
154A 53594D424FERRO:    DB      'SYMBOL TABLE OVERFLOW',CR
                ;
                SETTY:  ;SET CURRENT SYMBOL TYPE TO VALUE IN REG-A
1560 17                 RAL
1561 17                 RAL
1562 17                 RAL
1563 17                 RAL
1564 E6F0               ANI     0F0H    ;TYPE MOVED TO HIGH ORDER 4-BITS
1566 47                 MOV     B,A     ;SAVE IT IN B
1567 2AD601             LHLD    SYADR   ;BASE OF SYMBOL TO ACCESS
156A 23                 INX     H
156B 23                 INX     H       ;ADDRESS OF TYPE/LENGTH FIELD
156C 7E                 MOV     A,M     ;GET IT AND MASK
156D E60F               ANI     0FH     ;LEAVE LENGTH
156F B0                 ORA     B       ;MASK IN TYPE
1570 77                 MOV     M,A     ;STORE IT
```



```
1571 C9                 RET
                ;
                GETTY:  ;RETURN THE TYPE OF THE VALUE IN CURRENT SYMBOL
1572 2AD601             LHLD    SYADR
1575 23                 INX     H
1576 23                 INX     H
1577 7E                 MOV     A,M
1578 1F                 RAR
1579 1F                 RAR
157A 1F                 RAR
157B 1F                 RAR
157C E60F               ANI     0FH     ;TYPE MOVED TO LOW 4-BITS OF REG-A
157E C9                 RET
                ;
                VALADR: ;GET VALUE FIELD ADDRESS FOR CURRENT SYMBOL
157F CD8E14             CALL    GETLN   ;PRINTNAME LENGTH TO ACCUM
1582 2AD601             LHLD    SYADR   ;BASE ADDRESS
1585 5F                 MOV     E,A
1586 1600               MVI     D,0
1588 19                 DAD     D       ;BASE(LEN)
1589 23                 INX     H
158A 23                 INX     H       ;FOR COLLISION FIELD
158B 23                 INX     H       ;FOR TYPE/LEN FIELD
158C C9                 RET             ;WITH H,L ADDRESSING VALUE FIELD
                ;
                SETVAL: ;SET THE VALUE FIELD OF THE CURRENT SYMBOL
                ;       VALUE IS SENT IN H,L
158D E5                 PUSH    H       ;SAVE VALUE TO SET
158E CD7F15             CALL    VALADR
1591 D1                 POP     D       ;POP VALUE TO SET, HL HAS ADDRESS TO FIL
1592 73                 MOV     M,E
1593 23                 INX     H
1594 72                 MOV     M,D     ;FIELD SET
1595 C9                 RET
                ;
                GETVAL: ;GET THE VALUE FIELD OF THE CURRENT SYMBOL TO H,L
1596 CD7F15             CALL    VALADR  ;ADDRESS OF VALUE FIELD TO H,L
1599 5E                 MOV     E,M
159A 23                 INX     H
159B 56                 MOV     D,M
159C EB                 XCHG
159D C9                 RET
                ;
15A0 =          ENDMOD  EQU     (* AND 0FFE0H) + 20H
159E                    END
```

```
CP/M MACRO ASSEM 2.0     0000     ASM SYMBOL TABLE MODULE

0188 ACCLEN      0189 ACCUM       0040 ACMAX       01D2 ASPC        1476 CH0
1471 CHASH       000D CR          0002 DLABT       15A0 ENDMOD      152F ENT1
1533 ENT2        14EB ENTER       021E EOR         0004 EQUT        154A ERRO
01C9 EVALUE      0008 EXTT        0005 FIXD        1498 FOUND       01D0 FPC
140E GETLN       1572 GETTY       1596 GETVAL      000C GLBT        145B HASHC
1358 HASHT       007F HMASK       0080 HSIZE       0001 IDEN        1462 INI0
145C INISY       0200 IOMOD       14E1 LCOMP       14AC LENOK       1488 LOOK0
14D5 LOOK1       149E LOOKUP      0006 MACT        0002 NUMB        1541 OVERER
01CF PASS        007B PBMAX       0184 PBP         018C PBUFF       0212 PCON
0001 PLABT       000B REFT        1402 SETLN       0005 SETT        1560 SETTY
150D SETVAL      0004 SPECL       0003 STRNG       01D6 SYADR       01D4 SYBAS
01CD SYMAX       01CB SYTOP       0185 TOKEN       157F VALADR      0186 VALUE
```

```
CP/M MACRO ASSEM 2.0     0001     ASM TABLE SEARCH MODULE

                                  TITLE   'ASM TABLE SEARCH MODULE'
 15A0                             ORG     15A0H
 15A0 C36018                      JMP     ENDMOD  ;TO NEXT MODULE
 15A3 C38317                      JMP     BSEAR
 15A6 C31018                      JMP     BGET
                  ;
                  ;       COMMON EQUATES
 0078 =           PBMAX   EQU     120     ;MAX PRINT SIZE
 018C =           PBUFF   EQU     18CH    ;PRINT BUFFER
 0184 =           PBP     EQU     PBUFF+PBMAX     ;PRINT BUFFER POINTER
                  ;
 0185 =           TOKEN   EQU     PBP+1   ;CURRENT TOKEN UDER SCAN
 0186 =           VALUE   EQU     TOKEN+1 ;VALUE OF NUMBER IN BINARY
 0188 =           ACCLEN  EQU     VALUE+2 ;ACCUMULATOR LENGTH
 0040 =           ACMAX   EQU     64      ;MAX ACCUMULATOR LENGTH
 0189 =           ACCUM   EQU     ACCLEN+1
                  ;
 01C9 =           EVALUE  EQU     ACCUM+ACMAX     ;VALUE FROM EXPRESSION ANALYSIS
                  ;
 01CB =           SYTOP   EQU     EVALUE+2        ;CURRENT SYMBOL TOP
 01CD =           SYMAX   EQU     SYTOP+2         ;MAX ADDRESS+1
                  ;
 01CF =           PASS    EQU     SYMAX+2 ;CURRENT PASS NUMBER
 01D0 =           FPC     EQU     PASS+1  ;FILL ADDRESS FOR NEXT HEX BYTE
 01D2 =           ASPC    EQU     FPC+2   ;ASSEMBLER'S PSEUDO PC
                  ;
                  ;       GLOBAL EQUATES
 0001 =           IDEN    EQU     1       ;IDENTIFIER
 0002 =           NUMB    EQU     2       ;NUMBER
 0003 =           STRNG   EQU     3       ;STRING
 0004 =           SPECL   EQU     4       ;SPECIAL CHARACTER
                  ;
 0001 =           PLABT   EQU     0001B   ;PROGRAM LABEL
 0002 =           DLABT   EQU     0010B   ;DATA LABEL
 0004 =           EQUT    EQU     0100B   ;EQUATE
 0005 =           SETT    EQU     0101B   ;SET
 0006 =           MACT    EQU     0110B   ;MACRO
                  ;
 0008 =           EXTT    EQU     1000B   ;EXTERNAL
 000B =           REFT    EQU     1011B   ;REFER
 000C =           GLBT    EQU     1100B   ;GLOBAL
                  ;
                  ;
 000D =           CR      EQU     0DH     ;CARRIAGE RETURN
                  ;
                  ;
                  ;       TABLE DEFINITIONS
                  ;
                  ;       TYPES
 0000 =           XBASE   EQU     0       ;START OF OPERATORS
                  ;       01 THROUGH 015 DENOTE OPERATIONS
 0010 =           RT      EQU     16
 0011 =           PT      EQU     RT+1    ;RT IS REGISTER TYPE, PT IS PSEUDO OPERA
 0012 =           OBASE   EQU     PT+1
 0013 =           01      EQU     OBASE+1 ;SIMPLE
 0014 =           02      EQU     OBASE+2 ;LXI
```

```
CP/M MACRO ASSEM 2.0    #002    ASM TABLE SEARCH MODULE

 0015 =         03      EQU     OBASE+3 ;DAD
 0016 =         04      EQU     OBASE+4 ;PUSH/POP
 0017 =         05      EQU     OBASE+5 ;JMP/CALL
 0018 =         06      EQU     OBASE+6 ;MOV
 0019 =         07      EQU     OBASE+7 ;MVI
 001A =         08      EQU     OBASE+8 ;ACC IMMEDIATE
 001B =         09      EQU     OBASE+9 ;LDAX/STAX
 001C =         010     EQU     OBASE+10        ;LHLD/SHLD/LDA/STA
 001D =         011     EQU     OBASE+11        ;ACCUM REGISTER
 001E =         012     EQU     OBASE+12        ;INC/DEC
 001F =         013     EQU     OBASE+13        ;INX/DCX
 0020 =         014     EQU     OBASE+14        ;RST
 0021 =         015     EQU     OBASE+15        ;IN/OUT
                ;
                ;       X1 THROUGH X15 DENOTE OPERATORS
 0080 =         X1      EQU     XBASE   ;*
 0081 =         X2      EQU     XBASE+1 ;/
 0082 =         X3      EQU     XBASE+2 ;MOD
 0083 =         X4      EQU     XBASE+3 ;SHL
 0084 =         X5      EQU     XBASE+4 ;SHR
 0085 =         X6      EQU     XBASE+5 ;+
 0086 =         X7      EQU     XBASE+6 ;-
 0087 =         X8      EQU     XBASE+7 ;UNARY -
 0088 =         X9      EQU     XBASE+8 ;NOT
 0089 =         X10     EQU     XBASE+9 ;AND
 008A =         X11     EQU     XBASE+10;OR
 008B =         X12     EQU     XBASE+11;XOR
 008C =         X13     EQU     XBASE+12;(
 008D =         X14     EQU     XBASE+13;)
 008E =         X15     EQU     XBASE+14;,
 008F =         X16     EQU     XBASE+15;CR
                ;
                ;
                ;
                ;
                ;       RESERVED WORD TABLES
                ;
                ;       BASE ADDRESS VECTOR FOR CHARACTERS
 15A9 C415      CINX:   DW      CHAR1   ;LENGTH 1 BASE
 15AB D415              DW      CHAR2   ;LENGTH 2 BASE
 15AD E615              DW      CHAR3   ;LENGTH 3 BASE
 15AF 0216              DW      CHAR4   ;LENGTH 4 BASE
 15B1 AE16              DW      CHAR5   ;LENGTH 5 BASE
 15B3 BD16              DW      CHAR6   ;LENGTH 6 BASE
                ;
 0005 =         CMAX    EQU     ($-CINX)/2-1    ;LARGEST STRING TO MATCH
                ;
                CLEN:   ;LENGTH VECTOR GIVES THE NUMBER OF ITEMS IN EACH TABLE
 15B5 10                DB      CHAR2-CHAR1
 15B6 09                DB      (CHAR3-CHAR2)/2
 15B7 34                DB      (CHAR4-CHAR3)/3
 15B8 0B                DB      (CHAR5-CHAR4)/4
 15B9 03                DB      (CHAR6-CHAR5)/5
                ;
                TVINX:  ;TABLE OF TYPE,VALUE PAIRS FOR EACH RESERVED SYMBOL
 15BA BD16              DW      TV1
```



```
CP/M MACRO ASSEM 2.0    #003    ASM TABLE SEARCH MODULE

 15BC DD16              DW      TV2
 15BE EF16              DW      TV3
 15C0 5717              DW      TV4
 15C2 6D17              DW      TV5
                ;
                ;       CHARACTER VECTORS FOR 1,2,3,4, AND 5 CHARACTER NAMES
 15C4 0D20292A  CHAR1:  DB      CR,'()*'
 15C8 2B                DB      '+'
 15C9 2C2D2F41          DB      ',-/A'
 15CD 42434445          DB      'BCDE'
 15D1 484C4D            DB      'HLM'
                ;
 15D4 4442444944CHAR2:  DB      'DBDIDSDW'
 15DC 4549494649        DB      'EIIFINOR'
 15E4 5350              DB      'SP'
                ;
 15E6 4143494144CHAR3:  DB      'ACIADCADDADI'
 15F2 414E41414E        DB      'ANAANDANICMA'
 15FE 434D43434D        DB      'CMCCMPCPIDAA'
 160A 4441444443        DB      'DADCRDCXEND'
 1616 455155484C        DB      'EQUHLTINRINX'
 1622 4A4D504C44        DB      'JMPLDALXIMOD'
 162E 4D4F564D56        DB      'MOVMVINOPNOT'
 163A 4F52414F52        DB      'ORAORGORIOUT'
 1646 504F505053        DB      'POPPSWRALRAR'
 1652 5245544524C       DB      'RETRLCRRCRST'
 165E 5342425342        DB      'SBBSBISETSHL'
 166A 5348525354        DB      'SHRSTASTCSUB'
 1676 535549584F        DB      'SUIXORXRAXRI'
                ;
 1682 43414C4C45CHAR4:  DB      'CALLENDMLDAXLHLDPCHL'
 1696 5055534853        DB      'PUSHSHLDSPHLSTAX'
 16A6 5043484758        DB      'XCHGXTHL'
                ;
 16AE 454E444946CHAR5:  DB      'ENDIFMACROTITLE'
                ;
                CHAR6:  ;END OF CHARACTER VECTOR
                ;
                TV1:    ;TYPE,VALUE PAIRS FOR CHAR1 VECTOR
 16BD 0F0ABC14          DB      X16,10,         X13,20          ;CR (
 16C1 0D1E0850          DB      X14,30,         X1,80           ;) *
 16C5 0546              DB      X6,70                           ;+
 16C7 0E0A0646          DB      X15,10,         X7,70           ;, -
 16CB 01501007          DB      X2,80,          RT,7            ;/ A
 16CF 10001001          DB      RT,0,           RT,1            ;B C
 16D3 10021003          DB      RT,2,           RT,3            ;D E
 16D7 10041005          DB      RT,4,           RT,5            ;H L
 16DB 1006              DB      RT,6                            ;M
                ;
                TV2:    ;TYPE,VALUE PAIRS FOR CHAR2 VECTOR
 16DD 110113F3          DB      PT,1,           01,0F3H         ;DB DI
 16E1 11021103          DB      PT,2,           PT,3            ;DS DW
 16E5 13FB1108          DB      01,0FBH,        PT,0            ;EI IF
 16E9 21DB0A28          DB      015,0DBH,       X11,40          ;IN OR
 16ED 1006              DB      RT,6                            ;SP
                ;
```

```
                ;
                TV3:    ;TYPE,VALUE PAIRS FOR CHAR3 VECTOR
16EF 1ACE1D88           DB      08,0CEH,        011,88H         ;ADC ADC
16F3 1D801AC6           DB      011,80H,        08,0C6H         ;ADD ADI
16F7 1DA08932           DB      011,0A0H,       X10,50          ;ANA ANI
16FB 1AE6132F           DB      08,0E6H,        01,2FH          ;ANI CMA
16FF 133F1DB8           DB      01,3FH,         011,0B8H        ;CMC CMP
1703 1AFE1327           DB      08,0FEH,        01,27H          ;CPI DAA
1707 15091E05           DB      03,89H,         012,05H         ;DAD DCR
170B 1F0B1104           DB      013,0BH,        PT,4            ;DCX END
170F 11071376           DB      PT,7,           01,76H          ;EQU HLT
1713 1E041F03           DB      012,04H,        013,03H         ;INR INX
1717 17C31C3A           DB      05,0C3H,        010,3AH         ;JMP LDA
171B 14010250           DB      02,01H,         X3,80           ;LXI MOD
171F 18401906           DB      06,40H,         07,06H          ;MOV MVI
1723 1300083C           DB      01,00H,         X9,60           ;NOP NOT
1727 1DB0110A           DB      011,0B0H,       PT,10           ;ORA ORG
172B 1AF621D3           DB      08,0F6H,        015,0D3H        ;ORI OUT
172F 16C11006           DB      04,0C1H,        RT,6            ;POP PSW
1733 1317131F           DB      01,17H,         01,1FH          ;RAL RAR
1737 13C91307           DB      01,0C9H,        01,07H          ;RET RLC
173B 130F20C7           DB      01,0FH,         014,0C7H        ;RRC RST
173F 1D981ADE           DB      011,098H,       00,0DEH         ;SBB SBI
1743 11080350           DB      PT,11,          X4,80           ;SET SHL
1747 04501C32           DB      X5,80,          010,32H         ;STA STC
174B 13371D90           DB      01,37H,         011,90H         ;STC SUB
174F 1AD60028           DB      08,0D6H,        X12,40          ;SUI XOR
1753 1DA01AEE           DB      011,0A0H,       00,0EEH         ;XRA XRI
                ;
                ;
                TV4:    ;TYPE,VALUE PAIRS FOR CHAR4 VECTOR
1757 17CD               DB      05,0CDH                         ;CALL
1759 1106100A           DB      PT,6,           09,0AH          ;ENDM LDAX
175D 1C2A13E9           DB      010,02AH,       01,0E9H         ;LHLD PCHL
1761 16C51C22           DB      04,0C5H,        010,22H         ;PUSH SHLD
1765 13F91B02           DB      01,0F9H,        09,02H          ;SPHL STAX
1769 13EB13E3           DB      01,0EBH,        01,0E3H         ;XCHG XTHL
                ;
                TV5:    ;TYPE,VALUE PAIRS FOR CHAR5 VECTOR
176D 11051109           DB      PT,5,           PT,9            ;ENDIF MACRO
1771 110C               DB      PT,12                           ;TITLE
                ;
                SUFTAB: ;TABLE OF SUFFIXES FOR J C AND R OPERATIONS
1773 4E5A5A284E         DB      'NZZ NCC POPEP M '
                ;
                BSEAR:  ;BINARY SEARCH MNEMONIC TABLE
                ;       INPUT:  UR = UPPER BOUND OF TABLE (I.E., TABLE LENGTH-1)
                ;               SR = SIZE OF EACH TABLE ELEMENT
                ;               H,L ADDRESS BASE OF TABLE TO SEARCH
                ;       OUTPUT: ZERO FLAG INDICATES MATCH WAS FOUND, IN WHICH CASE
                ;               THE ACCUMULATOR CONTAINS AN INDEX TO THE ELEMENT
                ;               NOT ZERO FLAG INDICATES NO MATCH FOUND IN TABLE
                ;
0000 =          UR      EQU     B       ;UPPER BOUND REGISTER
0001 =          LR      EQU     C       ;LOWER BOUND REGISTER
0002 =          SR      EQU     D       ;SIZE REGISTER
```



```
0003 =          MR      EQU     E       ;MIDDLE POINTER REGISTER
0000 =          SP1     EQU     B       ;SIZE PRIME, USED IN COMPUTING MIDDLE POS
0001 =          SP1P    EQU     C       ;ANOTHER COPY OF SIZE PRIME
0004 =          KR      EQU     H       ;K
                ;
1783 1EFF               MVI     MR,255  ;MARK M <> OLD M
1785 04                 INR     UR      ;U=U+1
1786 0E00               MVI     LR,0    ;L = 0
                ;
                ;       COMPUTE M' = (U+L)/2
1788 AF         NEXT:   XRA     A
1789 78                 MOV     A,UR    ;CY=0, A=U
178A 81                 ADD     LR      ;(U+L)
178B 1F                 RAR             ;(U+L)/2
178C BB                 CMP     MR      ;SAME AS LAST TIME THROUGH?
178D CAC417             JZ      NMATCH  ;JUMP IF = TO NO MATCH
                ;
                ;       MORE ELEMENTS TO SCAN
1790 5F                 MOV     MR,A    ;NEW MIDDLE VALUE
1791 E5                 PUSH    H       ;SAVE A COPY OF THE BASE ADDRESS
1792 D5                 PUSH    D       ;SAVE S,M
1793 C5                 PUSH    B       ;SAVE U,L
1794 E5                 PUSH    H       ;SAVE ANOTHER COPY OF THE BASE ADDRESS
1795 42                 MOV     SP1,SR  ;S' = S
1796 48                 MOV     SP1P,SP1        ;S'' = S'
1797 1600               MVI     SR,0    ;FOR DOUBLE ADD OPERATION BELOW (DOUBLE M
                ;
1799 210000             LXI     KR,0    ;K=0
179C 19         SUMK:   DAD     D       ;K = K + M
179D 05                 DCR     SP1     ;S' = S' - 1
179E C29C17             JNZ     SUMK    ;DECREMENT IF SP1 <> 0
                ;
                ;       K IS NOW RELATIVE BYTE POSITION
17A1 D1                 POP     D       ;TABLE BASE ADDRESS
17A2 19                 DAD     D       ;H,L CONTAINS ABSOLUTE ADDRESS OF BYTE TO
17A3 118901             LXI     D,ACCUM ;D,E ADDRESS CHARACTERS TO COMPARE
                ;
                COMK:   ;COMPARE NEXT CHARACTER
17A6 1A                 LDAX    D       ;ACCUM CHARACTER TO REG A
17A7 BE                 CMP     M       ;SAME AS TABLE ENTRY?
17A8 13                 INX     D
17A9 23                 INX     H       ;TO NEXT POSITIONS
17AA C2B617             JNZ     NCOM    ;JUMP IF NOT THE SAME
17AD 0D                 DCR     SP1P    ;MORE CHARACTERS?
17AE C2A617             JNZ     COMK
                ;
                ;       COMPLETE MATCH AT M
17B1 C1                 POP     B
17B2 D1                 POP     D       ;M RESTORED
17B3 E1                 POP     H
17B4 78                 MOV     A,MR    ;VALUE OF M COPIED IN A
17B5 C9                 RET             ;WITH ZERO FLAG SET
                ;
                NCOM:   ;NO MATCH, DETERMINE IF LESS OR GREATER
17B6 C1                 POP     B       ;U,L
17B7 D1                 POP     D       ;S,M
```

```
17B8 E1             POP     H       ;TABLE ADDRESS
17B9 DAC017         JC      NCOML
                    ;       ACCUM IS HIGHER
17BC 4B             MOV     LR,MR   ;L = M
17BD C38017         JMP     NEXT
                    ;
                    NCOML.  ;ACCUMULATOR IS LOW
17C0 43             MOV     UR,MR   ;U = M
17C1 C38017         JMP     NEXT
                    ;
                    NMATCH. ;NO MATCH
17C4 AF             XRA     A
17C5 3C             INR     A       ;SETS NOT ZERO FLAG
17C6 C9             RET
                    ;
                    PREFIX. ;J C OR R PREFIX?
17C7 3A8901         LDA     ACCUM
17CA 0117C2         LXI     B,(0C2H SHL 8) OR 05    ;JNZ OPCODE TO B, TYPE TO C
17CD FE4A           CPI     'J'
17CF C8             RZ              ;RETURN WITH ZERO FLAG SET IF J
17D0 06C4           MVI     B,0C4H          ;CNZ OPCODE TO B, TYPE IS IN C
17D2 FE43           CPI     'C'
17D4 C8             RZ
17D5 0113C8         LXI     B,(0C8H SHL 8) OR 01    ;RNZ OPCODE
17D8 FE52           CPI     'R'
17DA C9             RET
                    ;
                    SUFFIX. ;J R OR C RECOGNIZED, LOOK FOR SUFFIX
17DB 3A8801         LDA     ACCLEN
17DE FE04           CPI     4       ;CHECK LENGTH
17E0 D20D18         JNC     NSUFF   ;CARRY IF 0,1,2,3 IN LENGTH
17E3 FE03           CPI     3
17E5 CAF217         JZ      SUF0    ;ASSUME 1 OR 2 IF NO BRANCH
17E8 FE02           CPI     2
17EA C20D18         JNZ     NSUFF   ;RETURNS IF 0 OR 1
17ED 218B01         LXI     H,ACCUM+2
17F0 3620           MVI     M,' '   ;BLANK-OUT FOR MATCH ATTEMPT
                    SUF0.   ;SEARCH 'TIL END OF TABLE
17F2 010800         LXI     B,0     ;B=0, C=0 COUNTS TABLE DOWN TO ZERO OR MATCH
17F5 117317         LXI     D,SUFTAB
                    NEXTS.  ;LOOK AT NEXT SUFFIX
17F8 218A01         LXI     H,ACCUM+1       ;SUFFIX POSITION
17FB 1A             LDAX    D               ;CHARACTER TO ACCUM
17FC BE             CMP     M
17FD 13             INX     D               ;READY FOR NEXT CHARACTER
17FE C20518         JNZ     NEXT0           ;JMP IF NO MATCH
1801 1A             LDAX    D               ;GET NEXT CHARACTER
1802 23             INX     H               ;READY FOR COMPARE WITH ACCUM
1803 BE             CMP     M               ;SAME?
1804 C8             RZ                      ;RETURN WITH ZERO FLAG SET, B IS SUF
1805 13             NEXT0.  INX     D       ;MOVE TO NEXT CHARACTER
1806 04             INR     B               ;COUNT SUFFIX UP
1807 0D             DCR     C               ;COUNT TABLE LENGTH DOWN
1808 C2F817         JNZ     NEXTS
                    ;       END OF TABLE, MARK WITH NON ZERO FLAG
180B 0C             INR     C
```



```
180C C9             RET
                    ;
                    NSUFF.  ;NOT PROPER SUFFIX - SET NON ZERO FLAG
180D AF             XRA     A
180E 3C             INR     A
180F C9             RET
                    ;
                    BGET.   ;PERFORM BINARY SEARCH, AND EXTRACT TYPE AND VAL FIELDS F
                    ;       THE ITEM.  ZERO FLAG INDICATES MATCH WAS FOUND, WITH TYPE
                    ;       IN THE ACCUMULATOR, AND VAL IN REGISTER B.  THE SEARCH IS
                    ;       UPON THE LENGTH OF THE ACCUMULATOR
1810 3A8801         LDA     ACCLEN  ;ITEM LENGTH
1813 4F             MOV     C,A     ;SAVE A COPY
1814 3D             DCR     A       ;ACCLEN-1
1815 5F             MOV     E,A
1816 1600           MVI     D,0     ;DOUBLE ACCLEN-1 TO D,E
1818 D5             PUSH    D       ;SAVE A COPY FOR LATER
1819 FE05           CPI     CMAX    ;TOO LONG?
181B D25A18         JNC     NGET    ;NOT IN RANGE IF CARRY
181E 218515         LXI     H,CLEN  ;LENGTH VECTOR
1821 19             DAD     D
1822 46             MOV     UR,M    ;FILL UPPER BOUND FROM MEMORY
1823 21A915         LXI     H,CINX
1826 19             DAD     D
1827 19             DAD     D       ;BASE ADDRESS TO H,L
1828 56             MOV     D,M
1829 23             INX     H
182A 66             MOV     H,M
182B 6A             MOV     L,D     ;NOW IN H,L
182C 51             MOV     SR,C    ;FILL THE SIZE REGISTER
182D CD0317         CALL    BSEAR   ;PERFORM THE BINARY SEARCH
1830 C24518         JNZ     SCASE   ;ZERO IF FOUND
1833 D1             POP     D       ;RESTORE INDEX
1834 21BA15         LXI     H,TVINX
1837 19             DAD     D
1838 19             DAD     D       ;ADDRESSING PROPER TV ELEMENT
1839 5E             MOV     E,M
183A 23             INX     H
183B 56             MOV     D,M
                    ;       D,E IS BASE ADDRESS OF TYPE/VALUE VECTOR, ADD DISPLACEMENT
183C 6F             MOV     L,A
183D 2600           MVI     H,0
183F 29             DAD     H       ;DOUBLED
1840 19             DAD     D       ;INDEXED
1841 7E             MOV     A,M     ;TYPE TO ACC
1842 23             INX     H
1843 46             MOV     B,M     ;VALUE TO B
1844 C9             RET             ;TYPE IN ACC, VALUE IN B
                    ;
                    SCASE.  ;NAME NOT TOO LONG, BUT NOT FOUND IN TABLES, MAY BE J C OR
1845 D1             POP     D       ;RESTORE INDEX
1846 CDC717         CALL    PREFIX
1849 C0             RNZ             ;NOT FOUND AS PREFIX J C OR R IF NOT ZERO
184A C5             PUSH    B       ;SAVE VALUE AND TYPE
184B CDDB17         CALL    SUFFIX  ;ZERO IF SUFFIX MATCHED
184E 78             MOV     A,B     ;READY FOR MASK IF ZERO FLAG
```

```
CP/M MACRO ASSEM 2.0     #008     ASM TABLE SEARCH MODULE

 184F C1                 POP      B        ;RECALL VALUE AND TYPE
 1850 C0                 RNZ              ;RETURN IF NOT ZERO FLAG SET
                 ;       MASK IN THE PROPER BITS AND RETURN
 1851 B7                 ORA      A        ;CLEAR CARRY
 1852 17                 RAL
 1853 17                 RAL
 1854 17                 RAL
 1855 B0                 ORA      B        ;VALUE SET TO JNZ ...
 1856 47                 MOV      B,A      ;REPLACE
 1857 79                 MOV      A,C      ;RETURN WITH TYPE IN REGISTER A
 1858 BF                 CMP      A        ;CLEAR THE ZERO FLAG
 1859 C9                 RET
                 ;
                 NGET:   ;CAN'T FIND THE ENTRY, RETURN WITH ZERO FLAG RESET
 185A D1                 POP      D        ;GET THE ELEMENT BACK
 185B AF                 XRA      A        ;CLEAR
 185C 3C                 INR      A        ;ZERO FLAG RESET
 185D C9                 RET
                 ;
                 ;
 1860 =          ENDMOD  EQU      ($ AND 0FFE0H) + 20H     ;NEXT MODULE ADDRESS
 185E                    END
```



```
CP/M MACRO ASSEM 2.0     #009     ASM TABLE SEARCH MODULE

0188 ACCLEN     0189 ACCUM      0040 ACMAX      01D2 ASPC       1810 BGET
1783 BSEAR      15C4 CHAR1      15D4 CHAR2      15E6 CHAR3      1682 CHAR4
16AE CHAR5      168D CHAR6      15A9 CIHX       1585 CLEN       0005 CMAX
17A6 COMK       000D CR         0002 DLABT      1860 ENDMOD     0004 EQUT
01C9 EVALUE     0008 EXIT       01D8 FPC        000C GLBT       0001 IDEN
0004 KR         0001 LR         0006 MACT       0003 MR         17B6 MCOM
17C8 MCOML      1788 NEXT       1805 NEXT0      17F0 NEXTS      185A NGET
17C4 NMATCH     180D NSUFF      0002 NUM8       0013 O1         001C O10
001D O11        001E O12        001F O13        0020 O14        0021 O15
0014 O2         0015 O3         0016 O4         0017 O5         0018 O6
0019 O7         001A O8         001B O9         0012 OBASE      01CF PASS
0078 PBMAX      0184 PBP        010C PBUFF      0001 PLABT      17C7 PREFIX
0011 PT         000B REFT       0010 RT         1845 SCASE      0005 SETT
0000 SP1        0001 SP1P       0004 SPECL      0002 SR         0003 STRNG
17F2 SUF0       17DB SUFFIX     1773 SUFTAB     179C SUMK       01CD SYMAX
01CB SYTOP      0185 TOKEN      16BD TV1        16DD TV2        16EF TV3
1757 TV4        176D TV5        15BA TVINX      0000 UR         0186 VALUE
0000 X1         0009 X10        000A X11        000B X12        000C X13
000D X14        000E X15        000F X16        0001 X2         0002 X3
0003 X4         0004 X5         0005 X6         0006 X7         0007 X8
0008 X9         0000 XBASE
```

```
CP/M MACRO ASSEM 2.0     #001     ASM OPERAND SCAN MODULE

                         TITLE    'ASM OPERAND SCAN MODULE'
                 ;       OPERAND SCAN MODULE
 1860                    ORG      1860H
                 ;
                 ;       EXTERNALS
 0200 =          IOMOD   EQU      200H     ;I/O MODULE
 1100 =          SCMOD   EQU      1100H    ;SCANNER MODULE
 1340 =          SYMOD   EQU      1340H    ;SYMBOL TABLE MODULE
 15A0 =          BMOD    EQU      15A0H    ;BINARY SEARCH MODULE
                 ;
                 ;
 0210 =          PERR    EQU      IOMOD+10H
 1106 =          SCAN    EQU      SCMOD+6H         ;SCANNER ENTRY POINT
 000D =          CR      EQU      0DH      ;CARRIAGE RETURN
                 ;
 1346 =          LOOKUP  EQU      SYMOD+6H         ;LOOKUP
 1349 =          FOUND   EQU      LOOKUP+3         ;FOUND SYMBOL IF ZERO FLAG NOT SET
 134C =          ENTER   EQU      FOUND+3          ;ENTER SYMBOL
 134F =          SETTY   EQU      ENTER+3          ;SET TYPE FIELD
 1352 =          GETTY   EQU      SETTY+3          ;SET TYPE FIELD
 1355 =          SETVAL  EQU      GETTY+3          ;SET VALUE FIELD
 1358 =          GETVAL  EQU      SETVAL+3         ;GET VALUE FIELD
                 ;
 15A3 =          BSEAR   EQU      BMOD+3   ;BINARY SEARCH ROUTINE
 15A6 =          BGET    EQU      BSEAR+3  ;GET VALUES WITH SEARCH
                 ;
                 ;       COMMON EQUATES
 0078 =          PBMAX   EQU      120      ;MAX PRINT SIZE
 010C =          PBUFF   EQU      10CH     ;PRINT BUFFER
 0184 =          PBP     EQU      PBUFF+PBMAX      ;PRINT BUFFER POINTER
                 ;
 0185 =          TOKEN   EQU      PBP+1    ;CURRENT TOKEN UDER SCAN
 0186 =          VALUE   EQU      TOKEN+1  ;VALUE OF NUMBER IN BINARY
 0188 =          ACCLEN  EQU      VALUE+2  ;ACCUMULATOR LENGTH
 0040 =          ACMAX   EQU      64       ;MAX ACCUMULATOR LENGTH
 0189 =          ACCUM   EQU      ACCLEN+1
                 ;
 01C9 =          EVALUE  EQU      ACCUM+ACMAX      ;VALUE FROM EXPRESSION ANALYSIS
                 ;
 01CB =          SYTOP   EQU      EVALUE+2         ;CURRENT SYMBOL TOP
 01CD =          SYMAX   EQU      SYTOP+2          ;MAX ADDRESS+1
                 ;
 01CF =          PASS    EQU      SYMAX+2  ;CURRENT PASS NUMBER
 01D8 =          FPC     EQU      PASS+1   ;FILL ADDRESS FOR NEXT HEX BYTE
 01D2 =          ASPC    EQU      FPC+2    ;ASSEMBLER'S PSEUDO PC
                 ;
                 ;       GLOBAL EQUATES
 0001 =          IDEN    EQU      1        ;IDENTIFIER
 0002 =          NUMB    EQU      2        ;NUMBER
 0003 =          STRNG   EQU      3        ;STRING
 0004 =          SPECL   EQU      4        ;SPECIAL CHARACTER
                 ;
 0001 =          PLABT   EQU      0001B    ;PROGRAM LABEL
 0002 =          DLABT   EQU      0010B    ;DATA LABEL
 0004 =          EQUT    EQU      0100B    ;EQUATE
 0005 =          SETT    EQU      0101B    ;SET
```

```
CP/M MACRO ASSEM 2.0     #002     ASM OPERAND SCAN MODULE

0006 =          MACT    EQU     0110B   ;MACRO
                ;
0008 =          EXTT    EQU     1000B   ;EXTERNAL
0009 =          REFT    EQU     1001B   ;REFER
000C =          GLBT    EQU     1100B   ;GLOBAL
                ;
                ;
                ;       TABLE DEFINITIONS
0000 =          XBASE   EQU     0       ;START OF OPERATORS
000F =          OPER    EQU     15      ;LAST OPERATOR
0010 =          RT      EQU     16
0011 =          PT      EQU     RT+1    ;RT IS REGISTER TYPE, PT IS PSEUDO OPERATION
0012 =          OBASE   EQU     PT+1
                ;
0005 =          PLUS    EQU     5
0006 =          MINUS   EQU     6
0008 =          NOTF    EQU     8       ;NOT
000C =          LPAR    EQU     12
000D =          RPAR    EQU     13
000A =          OSMAX   EQU     10
0010 =          VSMAX   EQU     8*2
                ;
                ;
                ;       BEGINNING OF MODULE
1860 C3A01B             JMP     ENDMOD  ;PAST THIS MODULE
1863 C3191A             JMP     OPAND   ;SCAN OPERAND FIELD
1866 C36E19             JMP     MULF    ;MULTIPLY FUNCTION
1869 C33819             JMP     DIVE    ;DIVIDE FUNCTION
186C            UNARY:  DS      1       ;TRUE IF NEXT OPERATOR IS UNARY
186D            OPERV:  DS      OSMAX   ;OPERATOR STACK
1877            HIERV:  DS      OSMAX   ;OPERATOR PRIORITY
1881            VSTACK: DS      VSMAX   ;VALUE STACK
1891            OSP:    DS      1       ;OPERATOR STACK POINTER
1892            VSP:    DS      1       ;VALUE STACK POINTER
                ;
                ;
                ;
                STKV:   ;PLACE CURRENT H,L VALUE AT TOP OF VSTACK
1893 EB                 XCHG    ;HOLD VALUE IN D,E
1894 219218             LXI     H,VSP
1897 7E                 MOV     A,M
1898 FE10               CPI     VSMAX
189A DAA218             JC      STKV0
189D CD851B             CALL    ERREX   ;OVERFLOW IN EXPRESSION
18A0 3600               MVI     M,0     ;VSP=0
18A2 7E         STKV0:  MOV     A,M     ;GET VSP
18A3 34                 INR     M       ;VSP=VSP+1
18A4 34                 INR     M       ;VSP=VSP+2
18A5 4F                 MOV     C,A     ;SAVE VSP
18A6 0600               MVI     B,0     ;DOUBLE VSP
18A8 218118             LXI     H,VSTACK
18AB 09                 DAD     B
18AC 73                 MOV     M,E     ;LOW BYTE
18AD 23                 INX     H
18AE 72                 MOV     M,D     ;HIGH BYTE
18AF C9                 RET
```



```
CP/M MACRO ASSEM 2.0     #003     ASM OPERAND SCAN MODULE

                ;
                STKO:   ;STACK OPERATOR (REG-A) AND PRIORITY (REG-B)
18B0 F5                 PUSH    PSW     ;SAVE IT
18B1 219118             LXI     H,OSP
18B4 7E                 MOV     A,M
18B5 FE0A               CPI     OSMAX
18B7 DABF18             JC      STKO1
18BA 3600               MVI     M,0
18BC CD851B             CALL    ERREX   ;OPERATOR STACK OVERFLOW
18BF 5E         STKO1:  MOV     E,M     ;GET OSP
18C0 1600               MVI     D,0
18C2 34                 INR     M       ;OSP=OSP+1
18C3 F1                 POP     PSW     ;RECALL OPERATOR
18C4 216D18             LXI     H,OPERV
18C7 19                 DAD     D       ;OPERV(OSP)
18C8 77                 MOV     M,A     ;OPERV(OSP)=OPERATOR
18C9 217718             LXI     H,HIERV
18CC 19                 DAD     D
18CD 70                 MOV     M,B     ;HIERV(OSP)=PRIORITY
18CE C9                 RET
                ;
                LODV1:  ;LOAD TOP ELEMENT FROM VSTACK TO H,L
18CF 219218             LXI     H,VSP
18D2 7E                 MOV     A,M
18D3 B7                 ORA     A
18D4 C2DE18             JNZ     LODOK
18D7 CD851B             CALL    ERREX   ;UNDERFLOW
18DA 210000             LXI     H,0
18DD C9                 RET
                ;
18DE 35         LODOK:  DCR     M
18DF 35                 DCR     M       ;VSP=VSP-2
18E0 4E                 MOV     C,M     ;LOW BYTE
18E1 0600               MVI     B,0
18E3 218118             LXI     H,VSTACK
18E6 09                 DAD     B       ;VSTACK(VSP)
18E7 4E                 MOV     C,M     ;GET LOW BYTE
18E8 23                 INX     H
18E9 66                 MOV     H,M
18EA 69                 MOV     L,C
18EB C9                 RET
                ;
                LODV2:  ;LOAD TOP TWO ELEMENTS DE HOLDS TOP, HL HOLDS TOP-1
18EC CDCF18             CALL    LODV1
18EF EB                 XCHG
18F0 CDCF18             CALL    LODV1
18F3 C9                 RET
                ;
                APPLY:  ;APPLY OPERATOR IN REG-A TO TOP OF STACK
18F4 6F                 MOV     L,A
18F5 2600               MVI     H,0
18F7 29                 DAD     H       ;OPERATOR NUMBER*2
18F8 110119             LXI     D,OPTAB
18FB 19                 DAD     D       ;INDEXED OPTAB
18FC 5E                 MOV     E,M     ;LOW ADDRESS
18FD 23                 INX     H
```

```
18FE 66                 MOV     H,M     ;HIGH ADDRESS
18FF 6B                 MOV     L,E
1900 E9                 PCHL            ;SET PC AND GO TO SUBROUTINE
                ;
1901 0919       OPTAB.  DW      MULOP
1903 9219               DW      DIVOP
1905 9919               DW      MODOP
1907 9F19               DW      SHLOP
1909 AB19               DW      SHROP
190B BF19               DW      ADDOP
190D C619               DW      SUBOP
190F D019               DW      NEGOP
1911 D919               DW      NOTOP
1913 E019               DW      ANDOP
1915 EC19               DW      OROP
1917 F019               DW      XOROP
1919 8518               DW      ERREX   ;(
                ;
                ;       SPECIFIC HANDLERS FOLLOW
                SHFT.   ;SET UP OPERANDS FOR SHIFT L AND R
191B CDEC18             CALL    LODV2
191E 7A                 MOV     A,D     ;ENSURE 0-15
191F B7                 ORA     A
1920 C22719             JNZ     SHERR
1923 7B                 MOV     A,E
1924 FE11               CPI     17
1926 D8                 RC              ;RETURN IF 0-16 SHIFT
1927 CD8518     SHERR.  CALL    ERREX
192A 3E10               MVI     A,16
192C C9                 RET
                ;
                NEGF.   ;COMPUTE 0-H,L TO H,L
192D AF                 XRA     A
192E 95                 SUB     L
192F 6F                 MOV     L,A
1930 3E00               MVI     A,0
1932 9C                 SBB     H
1933 67                 MOV     H,A
1934 C9                 RET
                ;
1935 CDEC18     DIVF.   CALL    LODV2
                DIVE.   ;(EXTERNAL ENTRY FROM MAIN PROGRAM)
1938 EB                 XCHG            ;SWAP D,E WITH H,L FOR DIVIDE FUNCTION
                ;       COMPUTE X/Y WHERE X IS IN D,E AND Y IS IN H,L
                ;       THE VALUE OF X/Y APPEARS IN D,E AND X MOD Y IS IN H,L
                ;
1939 226B19             SHLD    DTEMP   ;SAVE X IN TEMPORARY
193C 216D19             LXI     H,BNUM  ;STORE BIT COUNT
193F 3611               MVI     M,11H
1941 010000             LXI     B,0     ;INTIALIZE RESULT
1944 C5                 PUSH    B
1945 AF                 XRA     A       ;CLEAR FLAGS
                DLOOP.
1946 7B                 MOV     A,E     ;GET LOW Y BYTE
1947 17                 RAL
1948 5F                 MOV     E,A
```



```
1949 7A                 MOV     A,D
194A 17                 RAL
194B 57                 MOV     D,A
194C 35                 DCR     M       ;DECREMENT BIT COUNT
194D E1                 POP     H       ;RESTORE TEMP RESULT
194E C8                 RZ              ;ZERO BIT COUNT MEANS ALL DONE
194F 3E00               MVI     A,0     ;ADD IN CARRY
1951 CE00               ACI     0       ;CARRY
1953 29                 DAD     H       ;SHIFT TEMP RESULT LEFT ONE BIT
1954 44                 MOV     B,H     ;COPY HA AND L TO A A ND C
1955 05                 ADD     L
1956 2A6B19             LHLD    DTEMP   ;GET ADDRESS OF X
1959 95                 SUB     L       ;SUBTRACT FROM TEMPORARY RESULT
195A 4F                 MOV     C,A
195B 78                 MOV     A,B
195C 9C                 SBB     H
195D 47                 MOV     B,A
195E C5                 PUSH    B       ;SAVE TEMP RESULT IN STACK
195F D26419             JNC     DSKIP   ;NO BORROW FROM SUBTRACT
1962 09                 DAD     B       ;ADD X BACK IN
1963 E3                 XTHL    ;REPLACE TEMP RESULT ON STACK
1964 216D19     DSKIP.  LXI     H,BNUM  ;RESTORE H,L
1967 3F                 CMC
1968 C34619             JMP     DLOOP   ;REPEAT LOOP STEPS
                ;
196B            DTEMP.  DS      2
196D            BNUM.   DS      1
                ;
                MULF.   ;MULTIPLY D,E BY H,L AND REPLACE H,L WITH RESULT
196E 44                 MOV     B,H
196F 4D                 MOV     C,L     ;COPY OF 1ST VALUE TO B,C FOR SHIFT AND AD
1970 210000             LXI     H,0     ;H,L IS THE ACCUMULATOR
1973 AF         MUL0.   XRA     A
1974 78                 MOV     A,B     ;CARRY IS CLEARED
1975 1F                 RAR
1976 47                 MOV     B,A
1977 79                 MOV     A,C
1978 1F                 RAR
1979 4F                 MOV     C,A
197A DA8219             JC      MUL1    ;SKIP THIS ADD IF LSB IS ZERO
197D B0                 ORA     B
197E C8                 RZ              ;RETURN WITH H,L
197F C38319             JMP     MUL2    ;SKIP ADD
1982 19         MUL1.   DAD     D       ;ADD CURRENT VALUE OF D
1983 EB         MUL2.   XCHG    ;READY FOR *2
1984 29                 DAD     H
1985 EB                 XCHG
1986 C37319             JMP     MUL0
                ;
                MULOP.  ;MULTIPLY D,E BY H,L
1989 CDEC18             CALL    LODV2
198C CD6E19             CALL    MULF
198F C3011A             JMP     ENDOP
                ;
                DIVOP.  ;DIVIDE H,L BY D,E
1992 CD3519             CALL    DIVF
```

```
CP/M MACRO ASSEM 2.0     0006    ASM OPERAND SCAN MODULE

 1995 EB                 XCHG            ;RESULT TO H,L
 1996 C3011A             JMP     ENDOP
                 ;
 1999 CD3519     MODOP:  CALL    DIVF
 199C C3011A             JMP     ENDOP
                 ;
 199F CD1B19     SHLOP:  CALL    SHFT    ;CHECK VALUES
 19A2 B7         SHL0:   ORA     A       ;DONE?
 19A3 CA011A             JZ      ENDOP
 19A6 29                 DAD     H       ;HL=HL*2
 19A7 3D                 DCR     A
 19A8 C3A219             JMP     SHL0
                 ;
 19AB CD1B19     SHROP:  CALL    SHFT
 19AE B7         SHR0:   ORA     A       ;DONE?
 19AF CA011A             JZ      ENDOP
 19B2 F5                 PUSH    PSW     ;SAVE CURRENT COUNT
 19B3 AF                 XRA     A
 19B4 7C                 MOV     A,H
 19B5 1F                 RAR
 19B6 67                 MOV     H,A
 19B7 7D                 MOV     A,L
 19B8 1F                 RAR
 19B9 6F                 MOV     L,A
 19BA F1                 POP     PSW
 19BB 3D                 DCR     A
 19BC C3AE19             JMP     SHR0
                 ;
 19BF CDEC18     ADDOP:  CALL    LODV2
 19C2 19         ADD0:   DAD     D
 19C3 C3011A             JMP     ENDOP
                 ;
 19C6 CDEC18     SUBOP:  CALL    LODV2
 19C9 EB                 XCHG            ;TREAT AS HL+(-DE)
 19CA CD2D19             CALL    NEGF    ;0-HL
 19CD C3C219             JMP     ADD0
                 ;
 19D0 CDCF18     NEGOP:  CALL    LODV1
 19D3 CD2D19     NEG0:   CALL    NEGF    ;COMPUTE 0-HL
 19D6 C3011A             JMP     ENDOP
                 ;
 19D9 CDCF18     NOTOP:  CALL    LODV1
 19DC 23                 INX     H       ;65536-HL = 65535-(HL+1)
 19DD C3D319             JMP     NEG0
                 ;
 19E0 CDEC18     ANDOP:  CALL    LODV2
 19E3 7A                 MOV     A,D
 19E4 A4                 ANA     H
 19E5 67                 MOV     H,A
 19E6 7B                 MOV     A,E
 19E7 A5                 ANA     L
 19E8 6F                 MOV     L,A
 19E9 C3011A             JMP     ENDOP
                 ;
 19EC CDEC18     OROP:   CALL    LODV2
 19EF 7A                 MOV     A,D



CP/M MACRO ASSEM 2.0     0007    ASM OPERAND SCAN MODULE

 19F0 B4                 ORA     H
 19F1 67                 MOV     H,A
 19F2 7B                 MOV     A,E
 19F3 B5                 ORA     L
 19F4 6F                 MOV     L,A
 19F5 C3011A             JMP     ENDOP
                 ;
 19F8 CDEC18     XOROP:  CALL    LODV2
 19FB 7A                 MOV     A,D
 19FC AC                 XRA     H
 19FD 67                 MOV     H,A
 19FE 7B                 MOV     A,E
 19FF AD                 XRA     L
 1A00 6F                 MOV     L,A
                 ;
 1A01 C39318     ENDOP:  JMP     STKV
                 ;
                 ;
                 ;
                 ENDEXP: ;RETURNS ZERO FLAG IF SYMBOL IS CR, ;, OR ,
 1A04 3A8501             LDA     TOKEN
 1A07 FE04               CPI     SPECL
 1A09 C8                 RNZ             ;NOT END IF NOT SPECIAL
                 ;
 1A0A 3A8901             LDA     ACCUM
 1A0D FE0D               CPI     CR
 1A0F C8                 RZ
 1A10 FE3B               CPI     ';'
 1A12 C8                 RZ
 1A13 FE2C               CPI     ','
 1A15 C8                 RZ
 1A16 FE21               CPI     '!'
 1A18 C9                 RET
                 ;
                 OPAND:  ;SCAN THE OPERAND FIELD OF AN INSTRUCTION
                 ;       (NOT A DB WITH FIRST TOKEN STRING > 2 OR 0)
 1A19 AF                 XRA     A
 1A1A 329118             STA     OSP     ;ZERO OPERATOR STACK POINTER
 1A1D 329218             STA     VSP
 1A20 3D                 DCR     A       ;255
 1A21 326C18             STA     UNARY
 1A24 210000             LXI     H,0
 1A27 22C901             SHLD    EVALUE
                 ;
                 OP0:    ;ARRIVE HERE WITH NEXT ITEM ALREADY SCANNED
 1A2A CD041A             CALL    ENDEXP  ;DONE?
 1A2D C25D1A             JNZ     OP1
                 ;       EMPTY THE OPERATOR STACK
 1A30 219118     EMPOP:  LXI     H,OSP
 1A33 7E                 MOV     A,M     ;GET THE OSP AND CHECK FOR EMPTY
 1A34 B7                 ORA     A
 1A35 CA481A             JZ      CHKVAL  ;JUMP IF EMPTY
 1A38 35                 DCR     M       ;POP ELEMENT
 1A39 5F                 MOV     E,A     ;COPY FOR DOUBLE ADD
 1A3A 1D                 DCR     E
 1A3B 1600               MVI     D,0


```

```
CP/M MACRO ASSEM 2.0     0008     ASM OPERAND SCAN MODULE

1A3D 216D1B              LXI      H,OPERY
1A40 19                  DAD      D          ;INDEXED - OPERY(OSP)
1A41 7E                  MOV      A,M        ;GET OPERATOR
1A42 CDF41B              CALL     APPLY      ;APPLY OPERATOR
1A45 C3301A              JMP      EMPOP
                         ;
                         CHKVAL:
1A48 3A9218              LDA      VSP        ;MUST HAVE ONE ELEMENT IT THE STACK
1A4B FE02                CPI      2
1A4D C4851B              CNZ      ERREX
1A50 3A8C01              LDA      PBUFF
1A53 FE20                CPI      ' '
1A55 C0                  RNZ                 ;EVALUE REMAINS AT ZERO
1A56 2A8118              LHLD     VSTACK     ;GET DOUBLE BYTE IN STACK
1A59 22C901              SHLD     EVALUE
1A5C C9                  RET
                         ;
                         OP1:     ;MORE TO SCAN
1A5D 3A8C01              LDA      PBUFF
1A60 FE20                CPI      ' '
1A62 C27F1B              JNZ      GETOP
1A65 3A8501              LDA      TOKEN
1A68 FE03                CPI      STRNG      ;IS THIS A STRING?
1A6A C2891A              JNZ      OP3
                         ;
                         ;        STRING - CONVERT TO DOUBLE PRECISION
1A6D 3A8801              LDA      ACCLEN
1A70 B7                  ORA      A
1A71 CC851B              CZ       ERREX      ;ERROR IF LENGTH=0
1A74 FE03                CPI      3
1A76 D4851B              CNC      ERREX      ;ERROR IF LENGTH>2
1A79 1600                MVI      D,0
1A7B 218901              LXI      H,ACCUM
1A7E 5E                  MOV      E,M        ;LSBYTE
1A7F 23                  INX      H
1A80 3D                  DCR      A          ;A HAS THE LENGTH
1A81 CA851A              JZ       OP2        ;ONE OR TWO BYTES
1A84 56                  MOV      D,M        ;FILL HIGH ORDER
1A85 EB          OP2:    XCHG                ;VALUE TO H,L
1A86 C3711B              JMP      STNUM      ;STORE TO STACK
                         ;
                         OP3:     ;NOT A STRING, CHECK FOR NUMBER
1A89 FE02                CPI      NUMB
1A8B C2941A              JNZ      OP4
1A8E 2A8601              LHLD     VALUE      ;NUMERIC VALUE
1A91 C3711B              JMP      STNUM
                         ;
                         OP4:     ;NOT STRING OR NUMBER, MUST BE ID OR SPECL
1A94 CDA615              CALL     BGET       ;BINARY SEARCH, GET ATTRIBUTES
1A97 C2311B              JNZ      OP6        ;MATCH?
                         ;        YES, MAY BE OPERATOR
1A9A FE10                CPI      OPER+1
1A9C D2261B              JNC      OP5
                         ;        OPERATOR ENCOUNTERED MS NIBBLE OF B IS PRIORITY NUMBER LS NIE
                         ;        IS THE OPERATOR
                         ;        ACC HAS THE OPERATOR NUMBER, B HAS PRIORITY
```



```
CP/M MACRO ASSEM 2.0     0009     ASM OPERAND SCAN MODULE

1A9F FE0C                CPI      LPAR       ;(?
1AA1 4F                  MOV      C,A        ;SAVE COPY OF OPERATOR NUMBER
1AA2 3A6C1B              LDA      UNARY
1AA5 C2851A              JNZ      OPER1      ;JUMP IF NOT A (
                         ;        ( ENCOUNTERED, UNARY MUST BE TRUE
1AA8 B7                  ORA      A
1AA9 CC851B              CZ       ERREX
1AAC 3EFF                MVI      A,0FFH
1AAE 326C1B              STA      UNARY      ;UNARY IS SET TRUE
1AB1 79                  MOV      A,C        ;RECOVER OPERATOR
1AB2 C3031B              JMP      OPER4      ;CALLS STKD AND SETS UNARY TO TRUE
                         ;
                         ;
                         OPER1:   ;NOT A LEFT PAREN
1AB5 B7                  ORA      A
1AB6 C20E1B              JNZ      OPER6      ;MUST BE + OR - SINCE UNARY IS SET
                         ;
                         ;        UNARY NOT SET, MUST BE BINARY OPERATOR
                         OPER2:   ;COMPARE HIERARCHY OF TOS
1AB9 C5                  PUSH     B          ;SAVE PRIORITY AND OPERATOR NUMBER
1ABA 3A911B              LDA      OSP
1ABD B7                  ORA      A
1ABE CADE1A              JZ       OPER3      ;NO MORE OPERATORS IN STACK
1AC1 5F                  MOV      E,A        ;OSP TO E
1AC2 1D                  DCR      E          ;OSP-1
1AC3 1600                MVI      D,0
1AC5 217F1B              LXI      H,HIERY
1AC8 19                  DAD      D          ;HL ADDRESSES TOP OF OPERATOR STACK
1AC9 7E                  MOV      A,M        ;PRIORITY OF TOP OPERATOR
1ACA B8                  CMP      B          ;CURRENT GREATER?
1ACB DADE1A              JC       OPER3      ;JUMP IF SO
                         ;        APPLY TOP OPERATOR TO VALUE STACK
1ACE 21911B              LXI      H,OSP
1AD1 73                  MOV      M,E        ;OSP=OSP-1
1AD2 216D1B              LXI      H,OPERY
1AD5 19                  DAD      D
1AD6 7E                  MOV      A,M        ;OPERATOR NUMBER TO ACC
1AD7 CDF41B              CALL     APPLY
1ADA C1                  POP      B          ;RESTORE OPERATOR NUMBER AND PRIORITY
1ADB C3891A              JMP      OPER2      ;FOR ANOTHER TEST
                         ;
                         OPER3:   ;ARRIVE HERE WHEN OPERATOR IS STACKED
                         ;        CHECK FOR RIGHT PAREN BALANCE
1ADE C1                  POP      B          ;OPERATOR NUMBER IN C, PRIORITY IN B
1ADF 79                  MOV      A,C
1AE0 FE0D                CPI      RPAR
1AE2 C2031B              JNZ      OPER4      ;JUMP IF NOT A RIGHT PAREN
                         ;
                         ;        RIGHT PAREN FOUND, STACK MUST CONTAIN LEFT PAREN TO DELETE
1AE5 219118              LXI      H,OSP
1AE8 7E                  MOV      A,M
1AE9 B7                  ORA      A          ;ZERO?
1AEA CAFC1A              JZ       LPERR      ;PAREN ERROR IF SO
1AED 3D                  DCR      A          ;OSP-1
1AEE 77                  MOV      M,A        ;STORED TO MEMORY
1AEF 5F                  MOV      E,A
```

```
1AF0 1600           MVI     D,0
1AF2 216D1B         LXI     H,OPERV
1AF5 19             DAD     D
1AF6 7E             MOV     A,M         ;TOP OPERATOR IN REG-A
1AF7 FE0C           CPI     LPAR
1AF9 CAFF1A         JZ      NLERR       ;JMP IF NO ERROR - PARENS BALANCE
1AFC CD851B   LPERR:  CALL    ERREX
              NLERR:  ;ERROR REPORTING COMPLETE
1AFF AF             XRA     A
1B00 C3081B         JMP     OPER5       ;TO CLEAR UNARY FLAG
              ;
              OPER4:  ;ORDINARY OPERATOR
1B03 CD8018         CALL    STKO
1B06 3EFF           MVI     A,0FFH      ;TO SET UNARY FLAG
1B08 326C18   OPER5:  STA     UNARY
1B0B C37F1B         JMP     GETOP       ;FOR ANOTHER ELEMENT
              ;
              OPER6:  ;UNARY SET, MUST BE + OR -
1B0E 79             MOV     A,C         ;RECALL OPERATOR
1B0F FE05           CPI     PLUS
1B11 CA7F1B         JZ      GETOP       ;IGNORE UNARY PLUS
1B14 FE06           CPI     MINUS
1B16 C21E1B         JNZ     CHKNOT
1B19 3C             INR     A           ;CHANGE TO UNARY MINUS
1B1A 4F             MOV     C,A
1B1B C3B91A         JMP     OPER2
              CHKNOT: ;UNARY NOT SYMBOL?
1B1E FE08           CPI     NOTF
1B20 C4851B         CNZ     ERREX
1B23 C3B91A         JMP     OPER2
              ;
              ;
              OP5:    ;ELEMENT FOUND IN TABLE, NOT AN OPERATOR
1B26 FE11           CPI     PT          ;PSEUDO OPERATOR?
1B28 CC851B         CZ      ERREX       ;ERROR IF SO
1B2B 68             MOV     L,B         ;GET LOW VALUE TO L
1B2C 2600           MVI     H,0         ;ZERO HIGH ORDER BYTE
1B2E C3711B         JMP     STNUM       ;STORE IT
              ;
              OP6:    ;NOT FOUND IN TABLE SCAN, $?
1B31 3A8501         LDA     TOKEN
1B34 FE04           CPI     SPECL
1B36 C2501B         JNZ     OP7
1B39 3A8901         LDA     ACCUM
1B3C FE24           CPI     '$'
1B3E CA4A1B         JZ      CURPC       ;USE CURRENT PC
1B41 CD851B         CALL    ERREX
1B44 210000         LXI     H,0
1B47 C3711B         JMP     STNUM
1B4A 2AD201   CURPC:  LHLD    ASPC        ;GET CURRENT PC
1B4D C3711B         JMP     STNUM
              ;
              OP7:    ;NOT $, LOOK IT UP
1B50 CD4613         CALL    LOOKUP
1B53 CD4913         CALL    FOUND
1B56 C2641B         JNZ     FIDENT
```



```
              ;       NOT FOUND IN SYMBOL TABLE, ENTER IF PASS 1
1B59 3E50           MVI     A,'P'
1B5B CD1B02         CALL    PERR
1B5E CD4C13         CALL    ENTER       ;ENTER SYMBOL WITH ZERO TYPE FIELD
1B61 C36E1B         JMP     FIDE0
1B64 CD5213   FIDENT: CALL    GETTY       ;TYPE TO H,L
1B67 E607           ANI     11B
1B69 3E55           MVI     A,'U'
1B6B CC1B02         CZ      PERR
              ;
              FIDE0:
1B6E CD5013         CALL    GETVAL      ;VALUE TO H,L
              ;
              STNUM:  ;STORE H,L TO VALUE STACK
1B71 3A6C18         LDA     UNARY
1B74 B7             ORA     A           ;UNARY OPERATION SET
1B75 CC851B         CZ      ERREX       ;OPERAND ENCOUNTERED WITH UNARY OFF
1B78 AF             XRA     A
1B79 326C18         STA     UNARY       ;SET TO OFF
1B7C CD9318         CALL    STKV        ;STACK THE VALUE
              ;
1B7F CD8611   GETOP:  CALL    SCAN
1B82 C32A1A         JMP     OP0
              ;
              ERREX:  ;PUT 'E' ERROR IN OUTPUT BUFFER
1B85 E5             PUSH    H
1B86 3E45           MVI     A,'E'
1B88 CD1B02         CALL    PERR
1B8B E1             POP     H
1B8C C9             RET
              ;
1BA0 =        ENDMOD  EQU     ($ AND 0FFE0H) + 20H    ;NEXT HALF PAGE
1B8D                END
```

```
CP/M MACRO ASSEM 2.0     0012     ASM OPERAND SCAN MODULE

0188 ACCLEN     0189 ACCUM      0040 ACMAX      19C2 ADD0       190F ADDOP
19E0 ANDOP      18F4 APPLY      01D2 ASPC       15A6 BGET       15A0 BMOD
196D BNUM       15A3 BSEAR      1B1E CHKNOT     1A48 CHKVAL     000D CR
1B4A CURPC      1938 DIVE       1935 DIVF       1992 DIVOP      0002 DLABT
1946 DLOOP      1964 DSKIP      1968 DTEMP      1A30 EMPOP      1A04 ENDEXP
1BA0 ENDMOD     1A01 ENDOP      134C ENTER      0004 EQUT       1B05 ERREX
01C9 EVALUE     0000 EXIT       106E FIDE0      1B64 FIDENT     1349 FOUND
01D0 FPC        1B7F GETOP      1352 GETTY      1358 GETVAL     000C GLBT
1877 HIERV      0001 IDEN       0200 IOMOD      10DE LODOK      10CF LODV1
18EC LODV2      1346 LOOKUP     000C LPAR       1AFC LPERR      0006 MACT
0006 MINUS      1999 MODOP      1973 MULB       1982 MULI       1983 MUL2
196E MULF       1989 MULOP      19D3 NEGB       192D NEGF       19D0 NEGOP
1AFF NLERR      0008 NOTF       19D9 NOTOP      0002 NUMB       0012 OBASE
1A2A OP0        1A5D OP1        1A85 OP2        1A89 OP3        1A94 OP4
1B26 OP5        1B31 OP6        1B58 OP7        1A19 OPAND      000F OPER
1AB5 OPER1      1AB9 OPER2      1ADE OPER3      1B03 OPER4      1B08 OPER5
1B0E OPER6      186D OPERV      1901 OPTAB      19EC OROP       0B0A OSMAX
1891 OSP        01CF PASS       0078 PBMAX      0184 PBP        010C PBUFF
0218 PERR       0001 PLABT      0005 PLUS       0011 PT         000B REFT
000D RPAR       0010 RT         1106 SCAN       1100 SCMOD      0005 SETT
134F SETTY      1355 SETVAL     1927 SHERR      1918 SHFT       19A2 SHL0
199F SHLOP      19AE SHR0       19A8 SHROP      0004 SPECL      18B0 STKO
18BF STKO1      1893 STKV       18A2 STKV0      1871 STNUM      0003 STRNG
19C6 SUBOP      01CD SYMAX      1348 SYMOD      01CB SYTOP      0165 TOKEN
186C UNARY      0186 VALUE      0010 VSMAX      1892 VSP        1801 VSTACK
0000 XBASE      19F8 XOROP
```

```
CP/M MACRO ASSEM 2.0     0001     ASM MAIN MODULE

                         TITLE    'ASM MAIN MODULE'
                ;        CP/M RESIDENT ASSEMBLER MAIN PROGRAM
                ;
                ;                 COPYRIGHT (C) 1976, 1977, 1978
                ;                 DIGITAL RESEARCH
                ;                 BOX 579, PACIFIC GROVE
                ;                 CALIFORNIA, 93950
                ;
                ;
1BA0                     ORG      1BA0H
                ;        MODULE ENTRY POINTS
0200 =          IOMOD    EQU      200H     ;IO MODULE
1100 =          SCMOD    EQU      1100H    ;SCANNER MODULE
1348 =          SYMOD    EQU      1348H    ;SYMBOL TABLE MODULE
15A0 =          BMOD     EQU      15A0H    ;BINARY SEARCH MODULE
1060 =          OPMOD    EQU      1060H    ;OPERAND SCAN MODULE
                ;
0203 =          SETUP    EQU      IOMOD+3H         ;FILE SETUP FOR EACH PASS
0212 =          PCON     EQU      IOMOD+12H        ;WRITE CONSOLE BUFFER TO CR
0215 =          WOBUFF   EQU      IOMOD+15H        ;WRITE PRINT BUFFER AND REINITIAL
0218 =          PERR     EQU      IOMOD+18H        ;WRITE ERROR CHARACTER TO PRINT
021B =          DHEX     EQU      IOMOD+1BH        ;SEND HEX CHARACTER TO MACHINE C
021E =          EOR      EQU      IOMOD+1EH        ;END OF PROCESSING, CLOSE FILES
                ;
1103 =          INITS    EQU      SCMOD+3H         ;INITIALIZE SCANNER MODULE
1106 =          SCAN     EQU      SCMOD+6H         ;SCAN NEXT TOKEN
                ;
1343 =          INISY    EQU      SYMOD+3H         ;INITIALIZE SYMBOL TABLE
1346 =          LOOKUP   EQU      SYMOD+6H         ;LOOKUP SYMBOL IN ACCUMULATOR
1349 =          FOUND    EQU      SYMOD+9H         ;FOUND IF NZ FLAG
134C =          ENTER    EQU      SYMOD+0CH        ;ENTER SYMBOL IN ACCUMULATOR
134F =          SETTY    EQU      SYMOD+0FH        ;SET TYPE FIELD
1352 =          GETTY    EQU      SYMOD+12H        ;GET TYPE FIELD
1355 =          SETVAL   EQU      SYMOD+15H        ;SET VALUE FIELD
1358 =          GETVAL   EQU      SYMOD+18H        ;GET VALUE FIELD
                ;
15A6 =          BGET     EQU      BMOD+6H          ;BINARY SEARCH AND GET TYPE/VALUE
                ;
1063 =          OPAND    EQU      OPMOD+3H         ;GET OPERAND VALUE TO 'EVALUE'
1066 =          MULF     EQU      OPMOD+6H         ;MULT D,E BY H,L TO H,L
1069 =          DIVF     EQU      OPMOD+9H         ;DIVIDE HL BY DE, RESULT TO DE
                ;
                ;
                ;        COMMON EQUATES
0078 =          PBMAX    EQU      120      ;MAX PRINT SIZE
010C =          PBUFF    EQU      10CH     ;PRINT BUFFER
0184 =          PBP      EQU      PBUFF+PBMAX      ;PRINT BUFFER POINTER
                ;
0185 =          TOKEN    EQU      PBP+1    ;CURRENT TOKEN UDER SCAN
0186 =          VALUE    EQU      TOKEN+1  ;VALUE OF NUMBER IN BINARY
0188 =          ACCLEN   EQU      VALUE+2  ;ACCUMULATOR LENGTH
0040 =          ACMAX    EQU      64       ;MAX ACCUMULATOR LENGTH
0189 =          ACCUM    EQU      ACCLEN+1
                ;
01C9 =          EVALUE   EQU      ACCUM+ACMAX      ;VALUE FROM EXPRESSION ANALYSIS
                ;
```

```
CP/M MACRO ASSEM 2.0    0002    ASM MAIN MODULE

 01CB =         SYTOP   EQU     EVALUE+2        ;CURRENT SYMBOL TOP
 01CD =         SYMAX   EQU     SYTOP+2         ;MAX ADDRESS+1
                ;
 01CF =         PASS    EQU     SYMAX+2 ;CURRENT PASS NUMBER
 01D0 =         FPC     EQU     PASS+1  ;FILL ADDRESS FOR NEXT HEX BYTE
 01D2 =         ASPC    EQU     FPC+2   ;ASSEMBLER'S PSEUDO PC
 01D4 =         SYBAS   EQU     ASPC+2  ;BASE OF SYMBOL TABLE
 01D6 =         SYADR   EQU     SYBAS+2 ;CURRENT SYMBOL ADDRESS
                ;
                ;       GLOBAL EQUATES
 0001 =         IDEN    EQU     1       ;IDENTIFIER
 0002 =         NUMB    EQU     2       ;NUMBER
 0003 =         STRNG   EQU     3       ;STRING
 0004 =         SPECL   EQU     4       ;SPECIAL CHARACTER
                ;
 0001 =         PLABT   EQU     0001B   ;PROGRAM LABEL
 0002 =         DLABT   EQU     0010B   ;DATA LABEL
 0004 =         EQUT    EQU     0100B   ;EQUATE
 0005 =         SETT    EQU     0101B   ;SET
 0006 =         MACT    EQU     0110B   ;MACRO
                ;
 0008 =         EXTT    EQU     1000B   ;EXTERNAL
 000B =         REFT    EQU     1011B   ;REFER
 000C =         GLBT    EQU     1100B   ;GLOBAL
                ;
 000D =         CR      EQU     0DH     ;CARRIAGE RETURN
 000A =         LF      EQU     0AH     ;LINE FEED
 001A =         EOF     EQU     1AH     ;END OF FILE
 0010 =         HBMAX   EQU     16      ;STARTING POSITION OF PRINT LINE
                ;
                ;
 0010 =         RT      EQU     16      ;REGISTER TYPE
 0011 =         PT      EQU     RT+1    ;PSEUDO OPERATION
 0005 =         PENDIF  EQU     5       ;PSEUDO OPERATOR 'ENDIF'
 0012 =         OBASE   EQU     PT+1
 0013 =         01      EQU     OBASE+1 ;FIRST OPERATOR
 0021 =         015     EQU     OBASE+15;LAST OPERATOR
                ;
                ;       MAIN STATEMENT PROCESSING LOOP
 1BA0 AF                XRA     A
 1BA1 32CF01            STA     PASS    ;SET TO PASS 0 INITIALLY
 1BA4 CD4313            CALL    INISY   ;INITIALIZE THE SYMBOL TABLE
                RESTART,                ;PASS LOOP GOES FROM 0 TO 1
 1BA7 CD0311            CALL    INITS   ;INITIALIZE THE SCANNER
 1BAA CD0302            CALL    SETUP   ;SET UP THE INPUT FILE
 1BAD 210000            LXI     H,0
 1BB0 22EB20            SHLD    SYLAB   ;ASSUME NO STARTING LABEL
 1BB3 22D001            SHLD    FPC
 1BB6 22D201            SHLD    ASPC
 1BB9 22ED20            SHLD    EPC     ;END PC
                ;
                SCNEXT, ;SCAN THE NEXT INPUT ITEM
 1BBC CD0611            CALL    SCAN
 1BBF 3A8501    SCN0,   LDA     TOKEN
 1BC2 FE02              CPI     NUMB    ;SKIP LEADING NUMBERS FROM LINE EDITORS
 1BC4 CABC1B            JZ      SCNEXT
```



```
CP/M MACRO ASSEM 2.0    0003    ASM MAIN MODULE

 1BC7 FE04              CPI     SPECL   ;MAY BE PROCESSOR TECH'S COMMENT
 1BC9 C2DD1B            JNZ     SCN1
                ;       SPECIAL CHARACTER, CHECK FOR *
 1BCC 3A8901            LDA     ACCUM
 1BCF FE2A              CPI     '*'
 1BD1 C2311F            JNZ     CHEND   ;END OF LINE IF NOT *
                ;       * FOUND, NO PRECEDING LABEL ALLOWED
 1BD4 CD0020            CALL    SETLA
 1BD7 C27C1F            JNZ     STERR   ;ERROR IF LABEL
 1BDA C3521F            JMP     CHEN1   ;SCAN THE COMMENT OTHERWISE
                ;
                SCN1,   ;NOT NUMBER OR SPECIAL CHARACTER, CHECK FOR IDENTIFIER
 1BDD FE01              CPI     IDEN
 1BDF C27C1F            JNZ     STERR   ;ERROR IF NOT
                ;
                ;       IDENTIFIER FOUND, MAY BE LABEL, OPCODE, OR MACRO
 1BE2 CDA615            CALL    BGET    ;BINARY SEARCH FIXED DATA
 1BE5 CA301C            JZ      CHKPT   ;CHECK FOR PSEUDO OR REAL OPERATOR
                ;
                ;       BINARY SEARCH WAS UNSUCCESSFUL, CHECK FOR MACRO
 1BE8 CD4613            CALL    LOOKUP
 1BEB CD4913            CALL    FOUND
 1BEE C2FE1B            JNZ     LFOUN   ;NZ FLAG SET IF FOUND
                ;
                ;       NOT FOUND, ENTER IT
 1BF1 CD4C13            CALL    ENTER   ;THIS MUST BE PASS 0
 1BF4 3ACF01            LDA     PASS
 1BF7 B7                ORA     A
 1BF8 C4D720            CNZ     ERRP    ;PHASE ERROR IF NOT
 1BFB C30C1C            JMP     SETSY   ;SET SYLAB
                ;
                ;       ITEM WAS FOUND, CHECK FOR MACRO
 1BFE CD5213    LFOUN,  CALL    GETTY
 1C01 FE06              CPI     MACT
 1C03 C20C1C            JNZ     SETSY
                ;
                ;       MACRO DEFINITION FOUND, EXPAND MACRO
 1C06 CDE320            CALL    ERRN    ;NOT CURRENTLY IMPLEMENTED
 1C09 C3521F            JMP     CHEN1   ;SCANS TO END OF CURRENT LINE
                ;
                SETSY,  ;LABEL FOUND - IS IT THE ONLY ONE?
 1C0C 2AEB20            LHLD    SYLAB
 1C0F 7D                MOV     A,L
 1C10 B4                ORA     H
 1C11 C4DD20            CNZ     ERRL    ;LABEL ERROR IF NOT
 1C14 2AD601            LHLD    SYADR   ;ADDRESS OF SYMBOL
 1C17 22EB20            SHLD    SYLAB   ;MARK AS LABEL FOUND
                ;
                ;       LABEL FOUND, SCAN OPTIONAL ':'
 1C1A CD0611            CALL    SCAN
 1C1D 3A8501            LDA     TOKEN
 1C20 FE04              CPI     SPECL
 1C22 C2BF1B            JNZ     SCN0    ;SKIP NEXT SCAN IF NOT SPECIAL
 1C25 3A8901            LDA     ACCUM
 1C28 FE3A              CPI     ':'
 1C2A C2BF1B            JNZ     SCN0
```

```
1C2D C3BC1B             JMP     SCNEXT  ;TO IGNORE ','
                ;
                ;       BINARY SEARCH FOUND SYMBOL, CHECK FOR PSEUDO OR REAL OP
1C30 FE11       CHKPT.  CPI     PT      ;PSEUDO OPCODE?
1C32 C2D71D             JNZ     CHKOT
                ;
                ;       PSEUDO OPCODE FOUND, BRANCH TO CASES
1C35 58                 MOV     E,B     ;B HAS PARTICULAR OPERATOR NUMBER
1C36 1600               MVI     D,0     ;DOUBLE PRECISION VALUE TO D,E
1C38 1B                 DCX     D       ;BIASED BY +1
1C39 21431C             LXI     H,PTTAB ;BASE OF JUMP TABLE
1C3C 19                 DAD     D
1C3D 19                 DAD     D
1C3E 5E                 MOV     E,M
1C3F 23                 INX     H
1C40 66                 MOV     H,M
1C41 6B                 MOV     L,E
1C42 E9                 PCHL            ;JUMP INTO TABLE
                ;
                PTTAB.  ;PSEUDO OPCODE JUMP TABLE
1C43 5B1C               DW      SDB     ;DB
1C45 A91C               DW      SDS     ;DS
1C47 C01C               DW      SDW     ;DW
1C49 DE1C               DW      SEND    ;END
1C4B 151D               DW      SENDIF  ;ENDIF
1C4D 181D               DW      SENDM   ;ENDM
1C4F 1E1D               DW      SEQU    ;EQU
1C51 401D               DW      SIF     ;IF
1C53 871D               DW      SMACRO  ;MACRO
1C55 8D1D               DW      SORG    ;ORG
1C57 A71D               DW      SSET    ;SET
1C59 CE1D               DW      STITLE  ;TITLE
                ;
                SDB.
1C5B CD0A20             CALL    FILAB   ;SET LABEL FOR THIS LINE TO ASPC
                SDB0.
1C5E CD0611             CALL    SCAN    ;PAST DB TO NEXT ITEM
1C61 3A8501             LDA     TOKEN   ;LOOK FOR LONG STRING
1C64 FE03               CPI     STRNG
1C66 C28C1C             JNZ     SDBC    ;SKIP IF NOT STRING
1C69 3A8801             LDA     ACCLEN
1C6C 3D                 DCR     A       ;LENGTH 1 STRING?
1C6D CA8C1C             JZ      SDBC
                ;       LENGTH 0,2,... STRING
1C70 47                 MOV     B,A
1C71 04                 INR     B
1C72 04                 INR     B       ;BECOMES 1,3,... FOR 0,2,... LENGTHS
1C73 218901             LXI     H,ACCUM ;ADDRESS CHARACTERS IN STRING
1C76 05         SDB1.   DCR     B       ;COUNT DOWN TO ZERO
1C77 CA861C             JZ      SDB2    ;SCAN DELIMITER AT END OF STRING
1C7A C5                 PUSH    B       ;SAVE COUNT
1C7B 46                 MOV     B,M     ;GET CHARACTER
1C7C 23                 INX     H
1C7D E5                 PUSH    H       ;SAVE ACCUM POINTER
1C7E CD4020             CALL    FILHB   ;SEND TO HEX FILE
1C81 E1                 POP     H
```



```
1C82 C1                 POP     B
1C83 C3761C             JMP     SDB1
1C86 CD0611     SDB2.   CALL    SCAN    ;TO THE DELIMITER
1C89 C39B1C             JMP     SDB3
                ;
                ;       NOT A LONG STRING
1C8C CD6318     SDBC.   CALL    OPAND   ;COMPUTE OPERAND
1C8F 2AC901             LHLD    EVALUE  ;VALUE TO H,L
1C92 7C                 MOV     A,H
1C93 B7                 ORA     A       ;HIGH ORDER MUST BE ZERO
1C94 C4D120             CNZ     ERRD    ;DATA ERROR
1C97 45                 MOV     B,L     ;GET LOW BYTE
1C98 CD4020             CALL    FILHB
                SDB3.   ;END OF ITEM - UPDATE ASPC
1C9B CDF91F             CALL    SETAS   ;SET ASPC TO FPC
1C9E CDBA1E             CALL    DELIM
1CA1 FE2C               CPI     ','
1CA3 CA5E1C             JZ      SDB0    ;FOR ANOTHER ITEM
1CA6 C3311F             JMP     CHEND   ;CHECK END OF LINE SYNTAX
                ;
                SDS.
1CA9 CD0A20             CALL    FILAB   ;HANDLE LABEL IF IT OCCURRED
1CAC CDA620             CALL    PADD    ;PRINT ADDRESS
1CAF CDD11E             CALL    EXP16   ;SCAN AND GET 16BIT OPERAND
1CB2 EB                 XCHG            ;TO D,E
1CB3 2AD201             LHLD    ASPC    ;CURRENT PSEUDO PC
1CB6 19                 DAD     D       ;+EXPRESSION
1CB7 22D201             SHLD    ASPC
1CBA 22D001             SHLD    FPC     ;NEXT TO FILL
1CBD C3311F             JMP     CHEND
                ;
                SDW.
1CC0 CD0A20             CALL    FILAB   ;HANDLE OPTIONAL LABEL
                SDW0.
1CC3 CDD11E             CALL    EXP16   ;GET 16BIT OPERAND
1CC6 E5                 PUSH    H       ;SAVE A COPY
1CC7 45                 MOV     B,L     ;LOW BYTE FIRST
1CC8 CD4020             CALL    FILHB   ;SEND LOW BYTE
1CCB E1                 POP     H       ;RECLAIM A COPY
1CCC 44                 MOV     B,H     ;HIGH BYTE NEXT
1CCD CD4020             CALL    FILHB   ;SEND HIGH BYTE
1CD0 CDF91F             CALL    SETAS   ;SET ASPC=FPC
1CD3 CDBA1E             CALL    DELIM   ;CHECK DELIMITER SYNTAX
1CD6 FE2C               CPI     ','
1CD8 CAC31C             JZ      SDW0    ;GET MORE DATA
1CDB C3311F             JMP     CHEND
                ;
                SEND.
1CDE CD0A20             CALL    FILAB
1CE1 CDA620             CALL    PADD    ;WRITE LAST LOC
1CE4 3A8C01             LDA     PBUFF
1CE7 FE20               CPI     ' '
1CE9 C2311F             JNZ     CHEND
1CEC CDD11E             CALL    EXP16   ;GET EXPRESSION IF IT'S THERE
1CEF 3A8C01             LDA     PBUFF
1CF2 FE20               CPI     ' '
```

```
1CF4 C2FA1C             JNZ     SEND0
1CF7 22ED20             SHLD    EPC     ;EXPRESSION FOUND, STORE IT FOR LATER
1CFA 3E20       SEND0.  MVI     A,' '
1CFC 320C01             STA     PBUFF   ;CLEAR ERROR, IF IT OCCURRED
1CFF CD0611             CALL    SCAN    ;CLEAR CR
1D02 3A8501             LDA     TOKEN
1D05 FE04               CPI     SPECL
1D07 C27C1F             JNZ     STERR
1D0A 3A8901             LDA     ACCUM
1D0D FE0A               CPI     LF
1D0F C27C1F             JNZ     STERR
1D12 C30B1F             JMP     ENDAS   ;END OF ASSEMBLER
                ;
                SENDIF.
1D15 C3D11D             JMP     POEND
                ;
                SENDM.
1D18 CDE320             CALL    ERRN
1D1B C3D11D             JMP     POEND
                ;
                SEQU.
1D1E CD0020             CALL    SETLA
1D21 CA7C1F             JZ      STERR   ;MUST BE A LABEL
1D24 2AD201             LHLD    ASPC    ;HOLD TEMP ASPC
1D27 E5                 PUSH    H       ;IN STACK
1D28 CDD11E             CALL    EXP16   ;GET 16BIT OPERAND
1D2B 22D201             SHLD    ASPC    ;VALUE OF EXPRESSION
1D2E CD0A20             CALL    FILAB
1D31 CDA920             CALL    PADDR   ;COMPUTED VALUE
1D34 211201             LXI     H,PBUFF+6       ;SPACE AFTER VALUE
1D37 363D               MVI     M,'='
1D39 E1                 POP     H       ;REAL ASPC
1D3A 22D201             SHLD    ASPC    ;CHANGE BACK
1D3D C3311F             JMP     CHEND
                ;
                SIF.
1D40 CD0A20             CALL    FILAB   ;IN CASE OF LABEL
1D43 CDD11E             CALL    EXP16   ;GET IF EXPRESSION
1D46 3A0C01             LDA     PBUFF
1D49 FE20               CPI     ' '
1D4B C2311F             JNZ     CHEND   ;SKIP IF ERROR
1D4E 7D                 MOV     A,L     ;GET LSB
1D4F 1F                 RAR
1D50 DA311F             JC      CHEND   ;TRUE IF CARRY BIT SET
                ;
                ;       SKIP TO EOF OR ENDIF
1D53 CD0611     SIF0.   CALL    SCAN
1D56 3A8501             LDA     TOKEN
1D59 FE04               CPI     SPECL
1D5B C26E1D             JNZ     SIF1
1D5E 3A8901             LDA     ACCUM
1D61 FE1A               CPI     EOF
1D63 3E42               MVI     A,'B'   ;BALANCE ERROR
1D65 CC1002             CZ      PERR
1D68 CA8B1F             JZ      ENDAS
1D6B C3531D             JMP     SIF0    ;FOR ANOTHER
```



```
                SIF1.   ;NOT A SPECIAL CHARACTER
1D6E FE01               CPI     IDEN
1D70 C2531D             JNZ     SIF0    ;NOT AN IDENTIFIER
1D73 CDA615             CALL    BGET    ;LOOK FOR ENDIF
1D76 C2531D             JNZ     SIF0    ;NOT FOUND
1D79 FE11               CPI     PI      ;PSEUDO OP?
1D7B C2531D             JNZ     SIF0
1D7E 78                 MOV     A,B     ;GET OPERATOR NUMBER
1D7F FE05               CPI     PENDIF  ;ENDIF?
1D81 C2531D             JNZ     SIF0    ;GET ANOTHER TOKEN
1D84 C3D11D             JMP     POEND   ;OK, CHECK END OF LINE
                ;
                $MACRO.
1D87 CDE320             CALL    ERRN
1D8A C3311F             JMP     CHEND
                ;
                SORG.
1D8D CDD11E             CALL    EXP16
1D90 3A0C01             LDA     PBUFF
1D93 FE20               CPI     ' '
1D95 C2311F             JNZ     CHEND   ;SKIP ORG IF ERROR
1D98 22D201             SHLD    ASPC    ;CHANGE PC
1D9B 22D001             SHLD    FPC     ;CHANGE NEXT TO FILL
1D9E CD0A20             CALL    FILAB   ;IN CASE OF LABEL
1DA1 CDA620             CALL    PADD
1DA4 C3311F             JMP     CHEND
                ;
                SSET.
1DA7 CD0020             CALL    SETLA
1DAA CA7C1F             JZ      STERR   ;MUST BE LABELLED
                ;
1DAD CD5213             CALL    GETTY
1DB0 FE05               CPI     SETT
1DB2 C4DD20             CNZ     ERRL    ;LABEL ERROR
1DB5 3E05               MVI     A,SETT
1DB7 CD4F13             CALL    SETTY   ;REPLACE TYPE WITH 'SET'
1DBA CDD11E             CALL    EXP16   ;GET THE EXPRESSION
1DBD E5                 PUSH    H       ;SAVE IT
1DBE CD0020             CALL    SETLA   ;RE-ADDRESS LABEL
1DC1 E1                 POP     H       ;RECLAIM IT
1DC2 CD5513             CALL    SETVAL
1DC5 210000             LXI     H,0
1DC8 22EB20             SHLD    SYLAB   ;PREVENT LABEL PROCESSING
1DCB C3311F             JMP     CHEND
                ;
                ;
                STITLE.
1DCE CDE320             CALL    ERRN    ;NOT IMPLEMENTED
                ;
                POEND.  ;PSEUDO OPERATOR END - SCAN TO NEXT TOKEN
1DD1 CD0611             CALL    SCAN
1DD4 C3311F             JMP     CHEND
                ;
                ;       NOT A PSEUDO OPCODE, CHECK FOR REAL OPCODE
1DD7 D613       CHKOT.  SUI     01      ;BASE OF OPCODES
1DD9 FE21               CPI     015     ;PAST LAST OPCODE?
```

```
CP/M MACRO ASSEM 2.0     0008    ASM MAIN MODULE

 1DDB D27C1F             JNC     STERR   ;STATEMENT ERROR IF SO
                         ;
                         ;       FOUND OPCODE, COMPUTE INDEX INTO TABLE AND JUMP TO CASE
 1DDE 5F                 MOV     E,A
 1DDF 1600               MVI     D,0
 1DE1 21EB1D             LXI     H,OPTAB
 1DE4 19                 DAD     D
 1DE5 19                 DAD     D
 1DE6 5E                 MOV     E,M
 1DE7 23                 INX     H
 1DE8 66                 MOV     H,M
 1DE9 6B                 MOV     L,E
 1DEA E9                 PCHL            ;JUMP TO CASE
                         ;
                 OPTAB:  ;OPCODE CATEGORIES
 1DEB 091E               DW      SSIMP   ;SIMPLE
 1DED 121E               DW      SLXI    ;LXI
 1DEF 1E1E               DW      SDAD    ;DAD
 1DF1 241E               DW      SPUSH   ;PUSH/POP
 1DF3 381E               DW      SJMP    ;JMP/CALL
 1DF5 411E               DW      SMOV    ;MOV
 1DF7 501E               DW      SMVI    ;MVI
 1DF9 601E               DW      SACCI   ;ACCUM IMMEDIATE
 1DFB 691E               DW      SLDAX   ;LDAX/STAX
 1DFD 701E               DW      SLHLD   ;LHLD/SHLD/LDA/STA
 1DFF 811E               DW      SACCR   ;ACCUM-REGISTER
 1E01 881E               DW      SINC    ;INC/DCR
 1E03 8F1E               DW      SINX    ;INX/DCX
 1E05 9E1E               DW      SRST    ;RESTART
 1E07 A51E               DW      SIN     ;IN/OUT
                         ;
                 SSIMP:  ;SIMPLE OPERATION CODES
 1E09 CD4820             CALL    FILHB   ;SEND HEX VALUE TO MACHINE CODE FILE
 1E0C CD0611             CALL    SCAN    ;TO NEXT TOKEN
 1E0F C3B11E             JMP     INCPC
                         ;
                 SLXI:   ;LXI H,16B
 1E12 CDFC1E             CALL    SHDREG  ;SCAN DOUBLE PRECISION REGISTER
 1E15 CD171F             CALL    CHCOM   ;CHECK FOR COMMA FOLLOWING REGISTER
 1E18 CD111F             CALL    SETADR  ;SCAN AND EMIT DOUBLE PRECISION OPERAND
 1E1B C3B11E             JMP     INCPC
                         ;
                 SDAD:   ;DAD B
 1E1E CDFC1E             CALL    SHDREG  ;SCAN AND EMIT DOUBLE PRECISION REGISTER
 1E21 C3B11E             JMP     INCPC
                         ;
                 SPUSH:  ;PUSH B POP D
 1E24 CDF21E             CALL    SHREG   ;SCAN SINGLE PRECISION REGISTER TO A
 1E27 FE30               CPI     11000B  ;MAY BE PSW
 1E29 CA311E             JZ      SPU0
                         ;       NOT PSW, MUST BE B,D, OR H
 1E2C E608               ANI     00100B  ;LOW BIT MUST BE 0
 1E2E C4BD20             CNZ     ERRR    ;REGISTER ERROR IF NOT
 1E31 79         SPU0:   MOV     A,C     ;RECALL REGISTER AND MASK IN CASE OF ERROR
 1E32 E630               ANI     110000B
 1E34 B0                 ORA     B       ;MASK IN OPCODE FOR PUSH OR POP
```



```
CP/M MACRO ASSEM 2.0     0009    ASM MAIN MODULE

 1E35 C3AE1E             JMP     FILINC  ;FILL HEX VALUE AND INCREMENT PC
                         ;
                 SJMP:   ;JMP 16B/ CALL 16B
 1E38 CD4820             CALL    FILHB   ;EMIT JMP OR CALL OPCODE
 1E3B CD111F             CALL    SETADR  ;EMIT 16BIT OPERAND
 1E3E C3B11E             JMP     INCPC
                         ;
                 SMOV:   ;MOV A,B
 1E41 CDF21E             CALL    SHREG
 1E44 B0                 ORA     B       ;MASK IN OPCODE
 1E45 47                 MOV     B,A     ;SAVE IN B TEMPORARILY
 1E46 CD171F             CALL    CHCOM   ;MUST BE COMMA SEPARATOR
 1E49 CDE71E             CALL    EXP3    ;VALUE MUST BE 0-7
 1E4C B0                 ORA     B       ;MASK IN OPCODE
 1E4D C3AE1E             JMP     FILINC
                         ;
                 SMVI:   ;MVI A,8B
 1E50 CDF21E             CALL    SHREG
 1E53 B0                 ORA     B       ;MASK IN OPCODE
 1E54 CD4720             CALL    FILHEX  ;EMIT OPCODE
 1E57 CD171F             CALL    CHCOM   ;SCAN COMMA
 1E5A CD0B1F             CALL    SETBYTE ;EMIT 8BIT VALUE
 1E5D C3B11E             JMP     INCPC
                         ;
                 SACCI:  ;ADI 0B
 1E60 CD4820             CALL    FILHB   ;EMIT IMMEDIATE OPCODE
 1E63 CD0B1F             CALL    SETBYTE ;EMIT 8BIT OPERAND
 1E66 C3B11E             JMP     INCPC
                         ;
                 SLDAX:  ;LDAX B/STAX D
 1E69 CDF21E             CALL    SHREG
 1E6C E620               ANI     10000B  ;MUST BE B OR D
 1E6E C4BD20             CNZ     ERRR    ;REGISTER ERROR IF NOT
 1E71 79                 MOV     A,C     ;RECOVER REGISTER NUMBER
 1E72 E610               ANI     10000B  ;CHANGE TO B OR D IF ERROR
 1E74 B0                 ORA     B       ;MASK IN OPCODE
 1E75 C3AE1E             JMP     FILINC  ;EMIT OPCODE
                         ;
                 SLHLD:  ;LHLD 16B/ SHLD 16B/ LDA 16B/ STA 16B
 1E78 CD4820             CALL    FILHB   ;EMIT OPCODE
 1E7B CD111F             CALL    SETADR  ;EMIT OPERAND
 1E7E C3B11E             JMP     INCPC
                         ;
                 SACCR:  ;ADD B
 1E81 CDE71E             CALL    EXP3    ;RIGHT ADJUSTED 3BIT VALUE FOR REGISTER
 1E84 B0                 ORA     B       ;MASK IN OPCODE
 1E85 C3AE1E             JMP     FILINC
                         ;
                 SINC:   ;INR B/DCR D
 1E88 CDF21E             CALL    SHREG   ;GET REGISTER
 1E8B B0                 ORA     B
 1E8C C3AE1E             JMP     FILINC
                         ;
                 SINX:   ;INX H/DCX B
 1E8F CDF21E             CALL    SHREG
 1E92 E608               ANI     00100B  ;MUST BE B D H OR SP
```

```
1E94 C4BD20             CNZ     ERRR    ;REGISTER ERROR IF NOT
1E97 79                 MOV     A,C     ;RECOVER REGISTER
1E98 E630               ANI     110000B ;IN CASE OF ERROR
1E9A B0                 ORA     B       ;MASK IN OPCODE
1E9B C3AE1E             JMP     FILINC
                ;
                SRST,   ;RESTART 4
1E9E CDF21E             CALL    SHREG   ;VALUE IS 0-7
1EA1 B0                 ORA     B       ;OPCODE MASKED
1EA2 C3AE1E             JMP     FILINC
                ;
                SIN,    ;IN 8B/OUT 8B
1EA5 CD4820             CALL    FILHB   ;EMIT OPCODE
1EA8 CD0B1F             CALL    SETBYTE ;EMIT 8BIT OPERAND
1EAB C3B11E             JMP     INCPC
                ;
                FILINC, ;FILL HEX VALUE FROM A BEFORE INCREMENTING PC
1EAE CD4720             CALL    FILHEX
                ;
                INCPC,  ;CHANGE ASSEMBLER'S PSEUDO PROGRAM COUNTER
1EB1 CD0A20             CALL    FILAB   ;SET ANY LABELS WHICH OCCUR ON THE LINE
1EB4 CDF91F             CALL    SETAS   ;ASPC=FPC
1EB7 C3311F             JMP     CHEND   ;END OF LINE SCAN
                ;
                ;
                ;       UTILITY SUBROUTINES FOR OPERATION CODES
                ;
                DELIM,  ;CHECK DELIMITER SYNTAX FOR DATA STATEMENTS
1EBA 3A8501             LDA     TOKEN
1EBD FE04               CPI     SPECL
1EBF C4D120             CNZ     ERRD
1EC2 3A8901             LDA     ACCUM
1EC5 FE2C               CPI     ','
1EC7 C8                 RZ
1EC8 FE3B               CPI     ';'
1ECA C8                 RZ
1ECB FE0D               CPI     CR
1ECD C4D120             CNZ     ERRD
1ED0 C9                 RET
                ;
                EXP16,  ;GET 16BIT VALUE TO H,L
1ED1 C5                 PUSH    B
1ED2 CD0611             CALL    SCAN    ;START SCANNING OPERAND FIELD
1ED5 CD6318             CALL    OPAND
1ED8 2AC901             LHLD    EVALUE  ;VALUE TO H,L
1EDB C1                 POP     B
1EDC C9                 RET
                ;
                EXP8,   ;GET 8BIT VALUE TO REG A
1EDD CDD11E             CALL    EXP16
1EE0 7C                 MOV     A,H
1EE1 B7                 ORA     A
1EE2 C4C720             CNZ     ERRV    ;VALUE ERROR IF HIGH BYTE NOT ZERO
1EE5 7D                 MOV     A,L
1EE6 C9                 RET
                ;
```



```
                EXP3,   ;GET 3BIT VALUE TO REG A
1EC7 CDDD1E             CALL    EXP8
1EEA FE08               CPI     8
1EEC D4C720             CNC     ERRV    ;VALUE ERROR IF >=8
1EEF E607               ANI     111B    ;REDUCE IF ERROR OCCURS
1EF1 C9                 RET
                ;
                SHREG,  ;GET 3BIT VALUE AND SHIFT LEFT BY 3
1EF2 CDE71E             CALL    EXP3
1EF5 17                 RAL
1EF6 17                 RAL
1EF7 17                 RAL
1EF8 E638               ANI     111000B
1EFA 4F                 MOV     C,A     ;COPY TO C
1EFB C9                 RET
                ;
                SHDREG, ;GET DOUBLE REGISTER TO A
1EFC CDF21E             CALL    SHREG
1EFF E608               ANI     001000B ;CHECK FOR A,C,E, OR L
1F01 C4BD20             CNZ     ERRR    ;REGISTER ERROR
1F04 79                 MOV     A,C     ;RECOVER REGISTER
1F05 E630               ANI     110000B ;FIX IT IF ERROR OCCURRED
1F07 B0                 ORA     B       ;MASK OPCODE
1F08 C34720             JMP     FILHEX  ;EMIT IT
                ;
                SETBYTE,        ;EMIT 16BIT OPERAND
1F0B CDDD1E             CALL    EXP8
1F0E C34720             JMP     FILHEX
                ;
                SETADR, ;EMIT 16BIT OPERAND
1F11 CDD11E             CALL    EXP16
1F14 C37420             JMP     FILADR
                ;
                CHCOM,  ;CHECK FOR COMMA FOLLOWING EXPRESSION
1F17 F5                 PUSH    PSW
1F18 C5                 PUSH    B
1F19 3A8501             LDA     TOKEN
1F1C FE04               CPI     SPECL
1F1E C2291F             JNZ     COMER
                ;       SPECIAL CHARACTER, CHECK FOR COMMA
1F21 3A8901             LDA     ACCUM
1F24 FE2C               CPI     ','
1F26 CA2E1F             JZ      COMRET  ;RETURN IF COMMA FOUND
                COMER,  ;COMMA ERROR
1F29 3E43               MVI     A,'C'
1F2B CD1802             CALL    PERR
                COMRET,
1F2E C1                 POP     B
1F2F F1                 POP     PSW
1F30 C9                 RET
                ;
                CHEND,  ;END OF LINE CHECK
1F31 CD0A20             CALL    FILAB   ;IN CASE OF A LABEL
1F34 3A8501             LDA     TOKEN
1F37 FE04               CPI     SPECL
1F39 C27C1F             JNZ     STERR   ;MUST BE A SPECIAL CHARACTER
```

```
1F3C 3A8901          LDA     ACCUM
1F3F FE0D            CPI     CR      ;CARRIAGE RETURN
1F41 C24A1F          JNZ     CHEN0
                ;    CARRIAGE RETURN FOUND, SCAN PICKS UP LF AND PUSHES LINE
1F44 CD0611          CALL    SCAN
1F47 C3BC1B          JMP     SCNEXT
                ;
                CHEN0:  ;NOT CR, CHECK FOR COMMENT
1F4A FE3B            CPI     ';'
1F4C C2721F          JNZ     CHEN2
1F4F CD0A20          CALL    FILAB   ;IN CASE LABELLED EMPTY LINE
                ;    CLEAR COMMENT TO END OF LINE
1F52 CD0611     CHEN1:  CALL    SCAN
1F55 3A8501          LDA     TOKEN
1F58 FE04            CPI     SPECL
1F5A C2521F          JNZ     CHEN1
1F5D 3A8901          LDA     ACCUM
1F60 FE0A            CPI     LF
1F62 CABC1B          JZ      SCNEXT
1F65 FE1A            CPI     EOF
1F67 CA881F          JZ      ENDAS   ;END OF ASSEMBLY IF EOF
1F6A FE21            CPI     '!'
1F6C CABC1B          JZ      SCNEXT  ;LOGICAL END OF LINE
1F6F C3521F          JMP     CHEN1   ;NONE OF THE ABOVE
                ;
                ;    NOT CR OR LF, MAY BE LOGICAL END OF LINE
1F72 FE21       CHEN2:  CPI     '!'
1F74 CABC1B          JZ      SCNEXT
1F77 FE1A            CPI     EOF
1F79 CA881F          JZ      ENDAS
                ;
                ;    STATEMENT ERROR IN OPERAND FIELD
1F7C 3E53       STERR:  MVI     A,'S'
1F7E CD1802          CALL    PERR
1F81 C3521F          JMP     CHEN1   ;TO BUMP LINE
                ;
                DIFF:   ;COMPUTE DE-HL TO HL
1F84 78              MOV     A,E
1F85 95              SUB     L
1F86 6F              MOV     L,A
1F87 7A              MOV     A,D
1F88 9C              SBB     H
1F89 67              MOV     H,A
1F8A C9              RET
                ;
                ENDAS:  ;END OF ASSEMBLY FOR THIS PASS
1F8B 21CF01          LXI     H,PASS
1F8E 7E              MOV     A,M
1F8F 34              INR     M       ;PASS NUMBER INCREMENTED
1F90 B7              ORA     A
1F91 CAA71B          JZ      RESTART
1F94 CD0611          CALL    SCAN    ;TO CLEAR LAST LINE FEED
1F97 CDA620          CALL    PADD    ;WRITE LAST ADDRESS
1F9A 211101          LXI     H,PBUFF+5
1F9D 360D            MVI     M,CR    ;SET TO CR FOR END OF MESSAGE
1F9F 210D01          LXI     H,PBUFF+1
```



```
1FA2 CD1282          CALL    PCON    ;PRINT LAST ADDRESS
                ;
                ;    COMPUTE REMAINING SPACE
1FA5 2ACB01          LHLD    SYTOP
1FA8 EB              XCHG
1FA9 2AD401          LHLD    SYBAS
1FAC CD841F          CALL    DIFF    ;DIFFERENCE TO H,L
1FAF E5              PUSH    H       ;SYTOP-SYBAS TO STACK
1FB0 2ACD01          LHLD    SYMAX
1FB3 EB              XCHG
1FB4 2AD401          LHLD    SYBAS
1FB7 CD841F          CALL    DIFF    ;SYMAX-SYBAS TO H,L
1FBA 5C              MOV     E,H
1FBB 1600            MVI     D,0     ;DIVIDED BY 256
1FBD E1              POP     H       ;SYTOP-SYBAS TO H,L
1FBE CD6918          CALL    DIVF    ;RESULT TO DE
1FC1 EB              XCHG
1FC2 CDA920          CALL    PADDR   ;PRINT H,L TO PBUFF
1FC5 211101          LXI     H,PBUFF+5       ;MESSAGE
1FC8 11D61F          LXI     D,EMSG          ;END MESSAGE
1FCB 1A         ENDA0:  LDAX    D
1FCC B7              ORA     A       ;ZERO?
1FCD CAE41F          JZ      ENDA1
1FD0 77              MOV     M,A
1FD1 23              INX     H
1FD2 13              INX     D
1FD3 C3CB1F          JMP     ENDA0
                ;
1FD6 4820555345EMSG:    DB      'H USE FACTOR',CR,0
                ;
1FE4 210E01     ENDA1:  LXI     H,PBUFF+2       ;BEGINNING OF RATIO
1FE7 CD1282          CALL    PCON
1FEA 2AED20          LHLD    EPC
1FED 22D001          SHLD    FPC     ;END PROGRAM COUNTER
1FF0 C31E02          JMP     EOR
                ;
                ;    UTILITY SUBROUTINES
                COMDH:  ;COMPARE D,E WITH H,L FOR EQUALITY (NZ FLAG IF NOT EQUAL)
1FF3 7A              MOV     A,D
1FF4 BC              CMP     H
1FF5 C0              RNZ
1FF6 7B              MOV     A,E
1FF7 BD              CMP     L
1FF8 C9              RET
                ;
                SETAS:  ;ASPC=FPC
1FF9 2AD001          LHLD    FPC
1FFC 22D201          SHLD    ASPC
1FFF C9              RET
                ;
                SETLA:  ;SYADR=SYLAB, FOLLOWED BY CHECK FOR ZERO
2000 2AEB20          LHLD    SYLAB
2003 22D601          SHLD    SYADR
2006 CD4913          CALL    FOUND
2009 C9              RET
                ;
```

```
                FILAB:  ;FILL LABEL VALUE WITH CURRENT ASPC, IF LABEL FOUND
200A CD0020             CALL    SETLA
200D C8                 RZ              ;RETURN IF NO LABEL DETECTED
                        ;
                        ;       LABEL FOUND, MUST BE DEFINED ON PASS-1
200E 210000             LXI     H,0
2011 22EB20             SHLD    SYLAB   ;TO MARK NEXT STATEMENT WITH NO LABEL
2014 3ACF01             LDA     PASS
2017 B7                 ORA     A
2018 C23120             JNZ     FIL1
                        ;
                        ;       PASS 0
201B CD5213             CALL    GETTY
201E F5                 PUSH    PSW     ;SAVE A COPY OF TYPE
201F E607               ANI     111B    ;CHECK FOR UNDEFINED
2021 C4DD20             CNZ     ERRL    ;LABEL ERROR
2024 F1                 POP     PSW     ;RESTORE TYPE
2025 F601               ORI     PLABT   ;SET TO LABEL TYPE
2027 CD4F13             CALL    SETTY   ;SET TYPE FIELD
202A 2AD201             LHLD    ASPC    ;GET CURRENT PC
202D CD5513             CALL    SETVAL  ;PLACE INTO VALUE FIELD
2030 C9                 RET
                        ;
                FIL1:   ;CHECK FOR DEFINED VALUE
2031 CD5213             CALL    GETTY
2034 E607               ANI     111B
2036 CCD720             CZ      ERRP    ;PHASE ERROR
                        ;       GET VALUE AND COMPARE WITH ASPC
2039 CD5813             CALL    GETVAL  ;TO H,L
203C EB                 XCHG
203D 2AD201             LHLD    ASPC
2040 CDF31F             CALL    COMDH
2043 C4D720             CNZ     ERRP    ;PHASE ERROR IF NOT THE SAME
2046 C9                 RET
                        ;
                FILHEX: ;WRITE HEX BYTE IN REGISTER A TO MACHINE CODE FILE IF PASS-
2047 47                 MOV     B,A
2048 3ACF01     FILHB:  LDA     PASS
204B B7                 ORA     A
204C 78                 MOV     A,B
204D CA6C20             JZ      FILHI
                        ;
                        ;       PASS - 1, WRITE HEX AND PRINT DATA
2050 C5                 PUSH    B       ;SAVE A COPY
2051 CD1B02             CALL    BHEX    ;INTO MACHINE CODE FILE
                        ;       MAY BE COMPLETELY EMPTY LINE, SO CHECK ADDRESS
2054 3A0D01             LDA     PBUFF+1
2057 FE20               CPI     ' '
2059 2AD201             LHLD    ASPC
205C CCA920             CZ      PADDR   ;PRINT ADDRESS FIELD
                        ;
205F 3AEF20             LDA     NBP
2062 FE10               CPI     NBMAX   ;TRUNCATE CODE IF TOO MUCH ON THIS LINE
2064 C1                 POP     B       ;RECALL HEX DIGIT
2065 D26C20             JNC     FILHI
                        ;       ROOM FOR DIGIT ON THIS LINE
```



```
2068 78                 MOV     A,B
2069 CD9620             CALL    WHEXB   ;WRITE HEX BYTE TO PRINT LINE
206C 2AD001     FILHI:  LHLD    FPC
206F 23                 INX     H
2070 22D001             SHLD    FPC     ;READY FOR NEXT BYTE
2073 C9                 RET
                        ;
                FILADR: ;EMIT DOUBLE PRECISION VALUE FROM H,L
2074 E5                 PUSH    H       ;SAVE A COPY
2075 45                 MOV     B,L
2076 CD4820             CALL    FILHB   ;LOW BYTE EMITTED
2079 E1                 POP     H       ;RECOVER A COPY OF H,L
207A 44                 MOV     B,H
207B C34820             JMP     FILHB   ;EMIT HIGH BYTE AND RETURN
                        ;
                        ;       UTILITY FUNCTIONS FOR PRINTING HEX ADDRESSES AND DATA
                CHEX:   ;CONVERT TO HEX
207E C630               ADI     '0'
2080 FE3A               CPI     '0'+10
2082 D8                 RC
2083 C607               ADI     'A'-'0'-10
2085 C9                 RET
                        ;
                WHEXN:  ;WRITE HEX NIBBLE
2086 CD7E20             CALL    CHEX    ;CONVERT TO ASCII FROM HEX
2089 21EF20             LXI     H,NBP
208C 5E                 MOV     E,M     ;NEXT POSITION TO PRINT
208D 1600               MVI     D,0     ;DOUBLE PRECISION
208F 34                 INR     M       ;NBP=NBP+1
2090 210C01             LXI     H,PBUFF
2093 19                 DAD     D
2094 77                 MOV     M,A     ;STORE IN PRINT BUFFER
2095 C9                 RET
                        ;
                WHEXB:  ;WRITE HEX BYTE TO PRINT BUFFER
2096 F5                 PUSH    PSW
2097 1F                 RAR
2098 1F                 RAR
2099 1F                 RAR
209A 1F                 RAR
209B E60F               ANI     0FH     ;HIGH ORDER NIBBLE NORMALIZE IN A
209D CD8620             CALL    WHEXN   ;WRITE IT
20A0 F1                 POP     PSW
20A1 E60F               ANI     0FH
20A3 C38620             JMP     WHEXN   ;WRITE AND RETURN
                        ;
20A6 2AD201     PADD:   LHLD    ASPC
                PADDR:  ;PRINT ADDRESS FIELD OF PRINT LINE FROM H,L
20A9 EB                 XCHG
20AA 21EF20             LXI     H,NBP   ;INITIALIZE NEXT TO FILL
20AD E5                 PUSH    H       ;SAVE A COPY OF NBP'S ADDRESS
20AE 3601               MVI     M,1
20B0 7A                 MOV     A,D     ;PRINT HIGH BYTE
20B1 D5                 PUSH    D       ;SAVE A COPY
20B2 CD9620             CALL    WHEXB
20B5 D1                 POP     D
```

```
CP/M MACRO ASSEM 2.0     0016     ASM MAIN MODULE

 2086 7B                 MOV      A,E
 2087 CD9620             CALL     WHEXB
 208A E1                 POP      H          ;ADDRESSING HBP
 208B 34                 INR      M          ;SKIP A SPACE AFTER ADDRESS FIELD
 208C C9                 RET
                 ;
                 ERRR.   ;EMIT REGISTER ERROR
 208D F5                 PUSH     PSW
 208E C5                 PUSH     B
 208F 3E52               MVI      A,'R'
 20C1 CD1802             CALL     PERR
 20C4 C1                 POP      B
 20C5 F1                 POP      PSW
 20C6 C9                 RET
                 ;
                 ERRV.   ;EMIT VALUE ERROR
 20C7 F5                 PUSH     PSW
 20C8 E5                 PUSH     H
 20C9 3E56               MVI      A,'V'
 20CB CD1802             CALL     PERR
 20CE E1                 POP      H
 20CF F1                 POP      PSW
 20D0 C9                 RET
                 ;
 20D1 F5         ERRD.   PUSH     PSW
 20D2 3E44               MVI      A,'D'      ;DATA ERROR
 20D4 C3E620             JMP      ERR
                 ;
 20D7 F5         ERRP.   PUSH     PSW
 20D8 3E50               MVI      A,'P'
 20DA C3E620             JMP      ERR
                 ;
 20DD F5         ERRL.   PUSH     PSW
 20DE 3E4C               MVI      A,'L'      ;LABEL ERROR
 20E0 C3E620             JMP      ERR
                 ;
 20E3 F5         ERRN.   PUSH     PSW
 20E4 3E4E               MVI      A,'N'      ;NOT IMPLEMENTED
                 ;
                 ERR.
 20E6 CD1802             CALL     PERR
 20E9 F1                 POP      PSW
 20EA C9                 RET
                 ;
 20EB            SYLAB.  DS       2          ;ADDRESS OF LINE LABEL
 20ED            EPC.    DS       2          ;END PC VALUE
 20EF            HBP.    DS       1          ;NEXT BYTE POSITION TO WRITE FOR MACHINE CO
 20F0                    END
```



```
CP/M MACRO ASSEM 2.0     0017     ASM MAIN MODULE

0180 ACCLEN     0189 ACCUM      0040 ACMAX      01D2 ASPC       15A6 BGET
15A0 BMOD       1F17 CHCOM      1F4A CHEN0      1F52 CHEN1      1F72 CHEN2
1F31 CHEND      207E CHEX       1DD7 CHKDT      1C30 CHKPT      1FF3 COMDH
1F29 COMER      1F2E COMRET     000D CR         1E8A DELIM      021B DHEX
1F84 DIFF       1069 DIVF       0002 DLABT      1FD6 EMSG       1FCB ENDA0
1FE4 ENDA1      1F8B ENDAS      134C ENTER      001A EOF        021E EOR
20ED EPC        0004 EQUT       20D1 ERRD       20DD ERRL       20E3 ERRN
20D7 ERRP       20BD ERRR       20C7 ERRV       20E6 ERR        01C9 EVALUE
1ED1 EXP16      1EE7 EXP3       1EDD EXPB       0008 EXTT       2031 FIL1
200A FILAB      2074 FILADR     2048 FILHB      2047 FILHEX     206C FILHI
1EAE FILINC     1349 FOUND      01D0 FPC        1352 GETTY      1358 GETVAL
000C GLBT       0001 IDEN       1E81 INCPC      1343 INISY      1103 INITS
0200 IOMOD      000A LF         1BFE LFOUN      1346 LOOKUP     0006 MACT
1066 MULF       0010 NBMAX      20EF NBP        0002 NUMB       0013 O1
0021 O15        0012 OBASE      1063 OPAND      1060 OPMOD      1DEB OPTAB
20A6 PADD       20A9 PADDR      01CF PASS       0078 PBMAX      0184 PBP
010C PBUFF      0212 PCON       0005 PENDIF     0218 PERR       0001 PLABT
1DD1 POEND      0011 PT         1C43 PTTAB      0008 REFT       18A7 RESTART
0010 RT         1E60 SACCI      1E81 SACCR      1106 SCAN       1100 SCMOD
1BBF SCN0       1BDD SCN1       1BBC SCNEXT     1E1E SDAD       1C5B SDB
1C5E SDB0       1C76 SDB1       1C06 SDB2       1C9B SDB3       1C8C SDBC
1CA9 SDS        1CC0 SDW        1CC3 SDW0       1CDE SEND       1CFA SEND0
1D15 SENDIF     1D10 SENDM      1D1E SEQU       1F11 SETADR     1FF9 SETAS
1F0B SETBYTE    2000 SETLA      1C0C SETSY      0005 SETT       134F SETTY
0203 SETUP      1355 SETVAL     1EFC SHDREG     1EF2 SHREG      1D40 SIF
1D53 SIF0       1D6E SIF1       1E08 SINC       1EA5 SIN        1E8F SINX
1E38 SJMP       1E69 SLDAX      1E78 SLHLD      1E12 SLKI       1D87 SMACRO
1E41 SMOV       1E50 SMVI       1D0D SORG       0004 SPECL      1E31 SPU0
1E24 SPUSH      1E9E SRST       1DA7 SSET       1E09 SSIMP      1F7C STERR
1DCE STITLE     0003 STRNG      01D6 SYADR      01D4 SYBAS      20EB SYLAB
01CD SYMAX      1340 SYMOD      01CB SYTOP      0185 TOKEN      0186 VALUE
2096 WHEXB      2086 WHEXN      0215 WOBUFF
```

```
                ;       DDT RELOCATOR PROGRAM, INCLUDED WITH THE MODULE TO PERFORM
                ;       THE MOVE FROM 200H TO THE DESTINATION ADDRESS
000E =          VERSION EQU     14      ;1.4
                ;
                ;       COPYRIGHT (C) 1976, 1977, 1978
                ;       DIGITAL RESEARCH
                ;       BOX 579 PACIFIC GROVE
                ;       CALIFORNIA 93950
                ;
0100                    ORG     100H
0200 =          STACK   EQU     200H
0005 =          BDOS    EQU     0005H
0009 =          PRNT    EQU     9       ;BDOS PRINT FUNCTION
0200 =          MODULE  EQU     200H    ;MODULE ADDRESS
                ;
0100 010000             LXI     B,0     ;ADDRESS FIELD FILLED-IN WHEN MODULE BUILT
0103 C33D01             JMP     START
0106 434F505952         DB      'COPYRIGHT (C) 1978, DIGITAL RESEARCH      '
0130 4444542056SIGNON:  DB      'DDT VERS '
0139 312E               DB      VERSION/10+'0','.'
013B 3424               DB      VERSION MOD 10 + '0','$'
013D 310002     START:  LXI     SP,STACK
0140 C5                 PUSH    B
0141 C5                 PUSH    B
0142 113001             LXI     D,SIGNON
0145 0E09               MVI     C,PRNT
0147 CD0500             CALL    BDOS
014A C1                 POP     B       ;RECOVER LENGTH OF MOVE
014B 210700             LXI     H,BDOS+2;ADDRESS FIELD OF JUMP TO BDOS (TOP MEMORY)
014E 7E                 MOV     A,M     ;A HAS HIGH ORDER ADDRESS OF MEMORY TOP
014F 3D                 DCR     A       ;PAGE DIRECTLY BELOW BDOS
0150 90                 SUB     B       ;A HAS HIGH ORDER ADDRESS OF RELOC AREA
0151 57                 MOV     D,A
0152 1E00               MVI     E,0     ;D,E ADDRESSES BASE OF RELOC AREA
0154 D5                 PUSH    D       ;SAVE FOR RELOCATION BELOW
                ;
0155 210002             LXI     H,MODULE;READY FOR THE MOVE
0158 78         MOVE:   MOV     A,B     ;BC=0?
0159 B1                 ORA     C
015A CA6501             JZ      RELOC
015D 0B                 DCX     B       ;COUNT MODULE SIZE DOWN TO ZERO
015E 7E                 MOV     A,M     ;GET NEXT ABSOLUTE LOCATION
015F 12                 STAX    D       ;PLACE IT INTO THE RELOC AREA
0160 13                 INX     D
0161 23                 INX     H
0162 C35801             JMP     MOVE
                ;
                RELOC:  ;STORAGE MOVED, READY FOR RELOCATION
                ;       HL ADDRESSES BEGINNING OF THE BIT MAP FOR RELOCATION
0165 D1                 POP     D       ;RECALL BASE OF RELOCATION AREA
0166 C1                 POP     B       ;RECALL MODULE LENGTH
0167 E5                 PUSH    H       ;SAVE BIT MAP BASE IN STACK
0168 62                 MOV     H,D     ;RELOCATION BIAS IS IN D
                ;
0169 78         REL0:   MOV     A,B     ;BC=0?
016A B1                 ORA     C
016B CA8701             JZ      ENDREL
                ;
                ;       NOT END OF THE RELOCATION, MAY BE INTO NEXT BYTE OF BIT
016E 0B                 DCX     B       ;COUNT LENGTH DOWN
016F 7B                 MOV     A,E
0170 E607               ANI     111B    ;0 CAUSES FETCH OF NEXT BYTE
0172 C27A01             JNZ     REL1
                ;       FETCH BIT MAP FROM STACKED ADDRESS
0175 E3                 XTHL
0176 7E                 MOV     A,M     ;NEXT 8 BITS OF MAP
0177 23                 INX     H
0178 E3                 XTHL            ;BASE ADDRESS GOES BACK TO STACK
0179 6F                 MOV     L,A     ;L HOLDS THE MAP AS WE PROCESS 8 LOCATIO
017A 7D         REL1:   MOV     A,L
017B 17                 RAL             ;CY SET TO 1 IF RELOCATION NECESSARY
017C 6F                 MOV     L,A     ;BACK TO L FOR NEXT TIME AROUND
017D D28301             JNC     REL2    ;SKIP RELOCATION IF CY=0
                ;
                ;       CURRENT ADDRESS REQUIRES RELOCATION
0180 1A                 LDAX    D
0181 84                 ADD     H       ;APPLY BIAS IN H
0182 12                 STAX    D
0183 13         REL2:   INX     D       ;TO NEXT ADDRESS
0184 C36901             JMP     REL0    ;FOR ANOTHER BYTE TO RELOCATE
                ;
                ENDREL: ;END OF RELOCATION
0187 D1                 POP     D       ;CLEAR STACKED ADDRESS
0188 2E00               MVI     L,0
018A E9                 PCHL            ;GO TO RELOCATED PROGRAM
018B                    END
```



```
0005 BDOS       0187 ENDREL     0200 MODULE     0158 MOVE       0009 PRNT
0169 REL0       017A REL1       0183 REL2       0165 RELOC      0130 SIGNON
0200 STACK      013D START      000E VERSION
```

```
CP/M MACRO ASSEM 2.0    0001    CP/M DEBUGGER (ASMOD) 1/78

                ;       CP/M DEBUGGER DISASSEMBLER/ASSEMBLER MODULE
                        TITLE   'CP/M DEBUGGER (ASMOD) 1/78'
                ;
                ;       COPYRIGHT (C) 1976, 1977, 1978
                ;       DIGITAL RESEARCH
                ;       BOX 579 PACIFIC GROVE, CA
                ;       93950
                ;
0000 =          FALSE   EQU     0               ;VALUE OF "FALSE"
FFFF =          TRUE    EQU     NOT FALSE       ;VALUE OF "TRUE"
0000 =          DEBUG   EQU     FALSE           ;TRUE IF CHECK-OUT TIME
FFFF =          RELOC   EQU     TRUE            ;TRUE IF GENERATING RELOC IMAGE
                ;
                        IF      DEBUG
                        ORG     1000H           ;IN LOW MEMORY FOR DEBUG
                        ELSE
                        IF      RELOC
0000                    ORG     0000H           ;READY FOR RELOCATION
                        ELSE
                        ORG     0D000H          ;DEBUG IN 64K
                        ENDIF
                        ENDIF
                ;
0005 =          JLOC1   EQU     0005H           ;BDOS JUMP LOCATION
                ;
                ;
                ;       ENTRY POINTS FOR DEBUGGING MONITOR
0600 =          DEMON   EQU     6+600H
0603 =          BEGIN   EQU     DEMON+03H       ;BEGINNING OF DEBUGGING MONITOR
0609 =          GETBUFF EQU     DEMON+9H        ;READ BUFFER
060C =          GNC     EQU     DEMON+0CH
060F =          PCHAR   EQU     DEMON+0FH       ;PRINT CHARACTER IN REG A
0612 =          PBYTE   EQU     DEMON+12H       ;PRINT BYTE
0615 =          PADDX   EQU     DEMON+15H       ;PRINT ADDRESS IN REG D,E
0618 =          SCANEXP EQU     DEMON+18H       ;SCAN 0,1, OR 2 EXPRESSIONS
061E =          BREAK   EQU     DEMON+1EH       ;CHECK FOR BREAK AT CONSOLE
06A1 =          PRLABEL EQU     DEMON+21H       ;PRINT SYMBOLIC LABEL
                ;
                ;
060C =          CI      EQU     GNC             ;SYNONYM FOR GNC
                ;
000D =          CR      EQU     0DH
000A =          LF      EQU     0AH
0009 =          TAB     EQU     09H
                ;
                MODLOC. ;MODULE LOCATION
0000 C30306             JMP     BEGIN   ;ADDRESS FIELD IS ALTERED AT "BEGIN"
0003 000000             DB      0,0,0   ;FILLER (USED IN SYMBOL TABLE)
0006 C34F03             JMP     DISENT
0009 C32405             JMP     ASMEN   ;ENTRY POINT FOR ASSEMBLER
000C            PC.     DS      2       ;CURRENT FAKED PC DURING DISASSEMBLY
000E            MPC.    DS      2       ;MAX VALUE FOR PC (STOP ADDRESS)
0010            PAGM.   DS      1       ;PAGE MODE IF NON ZERO
0011            TPC.    DS      2       ;TEMPORARY PC FOR ASSEMBLER RESTORE ON ERRO
0013            OLDSP.  DS      2       ;ENTRY SP VALUE
                ;
```


COPYRIGHT © 1978
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA. 93950
SER. # _____

```
CP/M MACRO ASSEM 2.0    0002    CP/M DEBUGGER (ASMOD) 1/78

                ;
                CO.     ;PRINT CHARACTER IN REGISTER C
0015 F5                 PUSH    PSW
0016 79                 MOV     A,C             ;PCHAR EXPECTS VALUE IN C
0017 CD0F06             CALL    PCHAR           ;TO PRINT THE CHARACTER
001A F1                 POP     PSW
001B C9                 RET
                ;
                ;
                DELIM.  ;CHECK FOR DELIMITER
001C FE20               CPI     ' '
001E C8                 RZ
001F FE09               CPI     TAB
0021 C8                 RZ
0022 FE2C               CPI     ','
0024 C8                 RZ
0025 FE0D               CPI     CR
0027 C8                 RZ
0028 FE7F               CPI     7FH
002A CA2405             JZ      ASMEN   ;RESTART CURRENT LINE
002D C9                 RET
                ;
                CRLF.   ;RETURN AND LINE FEED
002E 0E0D               MVI     C,CR
0030 CD1500             CALL    CO
0033 0E0A               MVI     C,LF
0035 CD1500             CALL    CO
0038 C9                 RET
                ;
                SCAN.   ;FILL OPCODE WITH CHARACTERS
                ;
0039 CD0C06     SC1.    CALL    CI
                SCAN8.  ;ENTER HERE IF CHARACTER SCANNED
003C FE0D               CPI     CR
003E CA1805             JZ      ERR
0041 CD1C00             CALL    DELIM
0044 CA3900             JZ      SC1
                ;
                ;       CLEAR BUFFER
0047 0E04               MVI     C,4
0049 217A06             LXI     H,OPCODE
004C 3620       SC0.    MVI     M,' '
004E 23                 INX     H
004F 0D                 DCR     C
0050 C24C00             JNZ     SC0
                ;
                ;       GARBAGE REMOVED AT BEGINNING OF SCAN
0053 0E05               MVI     C,5
0055 217A06             LXI     H,OPCODE
0058 77         SC2.    MOV     M,A     ;STORE CHARACTER
0059 CD0C06             CALL    CI
005C CD1C00             CALL    DELIM
005F CA6A00             JZ      SC3
0062 23                 INX     H
0063 0D                 DCR     C
0064 CA1805             JZ      ERR     ;TOO LONG
```

```
0067 C35800             JMP     SC2
                        ;
                SC3.    ;END OF CURRENT SCAN, COMPARE FOR EMPTY
006A 3A7A06             LDA     OPCODE
006D FE20               CPI     ' '
006F C9                 RET
                        ;
                HEX.    ;CONVERT ACCUMULATOR TO HEXADECIMAL
0070 D630               SUI     '0'
0072 FE0A               CPI     10
0074 D8                 RC              ;'0' - '9'
0075 C6F9               ADI     ('0'-'A'+10) AND 0FFH
0077 FE10               CPI     16
0079 D8                 RC
007A C31805             JMP     ERR
                        ;
                GADDR.  ;GET ADDRESS VALUE TO B (HIGH ORDER) AND C (LOW) WITH COPY
007D CD9806             CALL    SCANEXP ;READ 1 EXPRESSION
0080 3D                 DCR     A       ;GOES TO ZERO
0081 C21805             JNZ     ERR     ;? IF NOT A SINGLE EXPRESSION
0084 EB                 XCHG            ;ADDRESS OF EXPRESSION TO HL
0085 4E                 MOV     C,M     ;LOW BYTE
0086 23                 INX     H
0087 46                 MOV     B,M     ;HIGH BYTE
0088 79                 MOV     A,C     ;COPY OF LOW BYTE TO A
0089 05                 DCR     B
008A 04                 INR     B       ;SETS ZERO FLAG IF B IS ZERO
008B C9                 RET
                        ;
                GBYTE.  ;GET BYTE VALUE TO ACCUMULATOR AND C, CHECK FOR HIGH ORDER
008C CD7D80             CALL    GADDR
008F C21805             JNZ     ERR
0092 C9                 RET
                        ;
                        ;
                        ;*****************************************************
                        ;********* ASSEMBLER MODULE STARTS HERE **************
                        ;*****************************************************
                        ;
                ADJ.    ;MOVE REGISTER INDICATOR TO MIDDLE FIELD OF CODE
0093 17                 RAL
0094 17                 RAL
0095 17                 RAL
0096 E638               ANI     111000B
0098 C9                 RET
                        ;
                ADJ4.   ;MOVE TO LEFT BY 4 AND MASK
0099 17                 RAL
009A 17                 RAL
009B 17                 RAL
009C 17                 RAL
009D E630               ANI     110000B
009F C9                 RET
                        ;
                SEAR2.  ;SAME AS SEAR, EXCEPT 2 CHARACTER MATCH
                        ;       H,L ADDRESS TABLE TO MATCH ON
```



```
00A0 EB                 XCHG
00A1 2A7A06             LHLD    OPCODE  ;2ND BYTE IN D, 1ST BYTE IN E
00A4 EB                 XCHG            ;H,L ADDRESS TABLE
00A5 7B         SEA0.   MOV     A,E     ;GET 1ST BYTE
00A6 BE                 CMP     M       ;MATCH?
00A7 C2AF00             JNZ     SEA1    ;TO ADDRESS NEXT ELT
00AA 23                 INX     H       ;NEXT TO MATCH
00AB 7A                 MOV     A,D     ;2ND CHAR
00AC BE                 CMP     M
00AD C8                 RZ              ;MATCH AT CURRENT ENTRY
00AE 2B                 DCX     H
00AF 2B         SEA1.   DCX     H
00B0 2B                 DCX     H       ;ADDRESSES NEXT ELEMENT
00B1 0D                 DCR     C
00B2 C2A500             JNZ     SEA0    ;FOR ANOTHER COMPARE
                        ;
                        ;       NO MATCH IN TABLE, RETURN WITH NON-ZERO VALUE
00B5 0D                 DCR     C
00B6 C9                 RET
                        ;
                        ;
                SEAR.   ;SEARCH FOR MATCH IN OPCODE TABLE, LENGTH OF TABLE IN RE
                        ;       D,E CONTAINS ADDRESS OF BINARY EQUIVALENT OF OPCODE
                        ;       H,L ADDRESS FOUR CHARACTER OPCODE TO MATCH
                        ;       OPCODE CONTAINS FOUR BYTE OPCODE TYPED AT CONSOLE
                        ;       RETURNS WITH ZERO VALUE IF OPCOE FOUND, WITH D,E
                        ;       ADDRESSING PROPER BYTE, NON-ZERO IF NOT FOUND.
00B7 0604               MVI     B,4     ;4 CHARACTER MATCH
                        ;
00B9 D5                 PUSH    D       ;SAVE THE CURRENT BYTE VALUE LOCATION
00BA 117A06             LXI     D,OPCODE        ;ADDRESS CHARACTERS TYPED
00BD 1A         SE1.    LDAX    D       ;POINT TO FIRST BYTE TO MATCH
00BE BE                 CMP     M       ;SAME CHARACTER AS TABLE?
00BF C2CA00             JNZ     SE2     ;NO, SKIP TO NXT TABLE ENTRY
00C2 23                 INX     H       ;YES, LOOK AT NEXT CHARACTER
00C3 13                 INX     D       ;MOVE TO NEXT CHARACTER TYPED
00C4 05                 DCR     B       ;DECREMENT CHARACTER COUNT
00C5 C2BD00             JNZ     SE1     ;MORE TO MATCH?
                        ;
                        ;       COMPLETE MATCH, RETURN WITH D,E ADDRESSING BYTE VALUE
00C8 D1                 POP     D
00C9 C9                 RET
                        ;
                        ;       MISMATCH, FINISH COUNT
00CA 23         SE2.    INX     H
00CB 05                 DCR     B
00CC C2CA00             JNZ     SE2
                        ;
                        ;       H,L AT END OF FOUR BYTE AREA, MOVE BACK 8
00CF 11F8FF             LXI     D,-8
00D2 19                 DAD     D       ;H,L READY FOR NXT MATCH
                        ;
00D3 D1                 POP     D       ;RESTORE BYTE POINTER
00D4 13                 INX     D       ;MOVE TO NEXT IN CASE MATCH OK
00D5 0D                 DCR     C       ;MORE OPCODES TO MATCH?
00D6 C2B700             JNZ     SEAR    ;LOOK FOR MORE
```

```
                ;
                ;       NO MATCH FOUND IN TABLE, SET NON-ZERO VALUE AND RETURN
00D9 0D                 DCR     C
00DA C9                 RET
                ;
                ;
                GETREG: ;SCAN FOR SIMPLE REGISTER REFERENCE
00DB C5                 PUSH    B
00DC CD3900             CALL    SCAN
00DF CA1005             JZ      ERR
00E2 0E08               MVI     C,8     ;8 REGISTERS
00E4 216006             LXI     H,SREG  ;SIMPLE REGISTERS
00E7 CDA000             CALL    SEAR2   ;LOOK FOR 2 CHAR MATCH
00EA C21005             JNZ     ERR
00ED 0D                 DCR     C
00EE 79                 MOV     A,C
00EF C1                 POP     B
00F0 C9                 RET
                ;
                GETD:   ;GET DOUBLE PRECISION REGISTER
00F1 C5                 PUSH    B
00F2 CD3900             CALL    SCAN
00F5 CA1005             JZ      ERR
00F8 0E05               MVI     C,5
00FA 217206             LXI     H,DREG
00FD CDB700             CALL    SEAR
0100 C21005             JNZ     ERR
0103 0D                 DCR     C
0104 79                 MOV     A,C
0105 C1                 POP     B
0106 C9                 RET
                ;
                GETDR:  ;GET DOUBLE REGISTER (BDHSP)
0107 CDF100             CALL    GETD
010A FE04               CPI     4       ;PSW?
010C CA1005             JZ      ERR
010F C9                 RET
                ;
                GETPR:  ;GET PUSH/POP REGISTER (BDH OR PSW)
0110 CDF100             CALL    GETD
0113 FE03               CPI     3
0115 CA1005             JZ      ERR
0118 FE04               CPI     4
011A C0                 RNZ
011B 3D                 DCR     A       ;PSW MUST BE ADJUSTED
011C C9                 RET
                ;
                GCON:   ;GET CONDITION CODE
                ;       BUFFER IS SCANNED, MOVE LEFT BEFORE COMPARE
011D 217A06             LXI     H,OPCODE
0120 117B06             LXI     D,OPCODE+1
0123 0E02               MVI     C,2     ;MOVE TWO CHARACTERS
0125 1A         MOP:    LDAX    D       ;LOAD CHARACTER TO MOVE
0126 77                 MOV     M,A     ;MOVE LEFT
0127 23                 INX     H       ;NEXT DESTINATION
0128 13                 INX     D       ;NEXT SOURCE
```



```
0129 0D                 DCR     C
012A C22501             JNZ     MOP
                ;
                ;       MUST BE BLANK AT END
012D 1A                 LDAX    D
012E FE20               CPI     ' '
0130 C21005             JNZ     ERR
0133 77                 MOV     M,A
                ;
                ;       NOW READY TO DO THE COMPARE
0134 215006             LXI     H,CREG
0137 0E08               MVI     C,8
0139 CDA000             CALL    SEAR2
013C C21005             JNZ     ERR
013F 0D                 DCR     C
0140 79                 MOV     A,C
0141 CD9300             CALL    ADJ     ;MOVE TO BITS 3,4,5 OF BYTE (LSB = 0)
0144 C9                 RET
                ;
                GCONA:  ;GET CONDITION CODE TO REGISTER A, DOUBLE ADDRESS TO B,C
0145 CD1D01             CALL    GCON    ;CONDITION CODE TO A
0148 F5                 PUSH    PSW
0149 CD7D00             CALL    GADDR   ;VALUE TO B,C
014C F1                 POP     PSW
                ;       INCLUDE HIGH ORDER 11'S FOR J AND C OPCODES
014D F6C0               ORI     11000000B
014F C9                 RET
                ;
                SETMD:  ;SET MEMORY AT LOCATION PC TO VALUE ADDRESSED BY D
0150 1A                 LDAX    D       ;VALUE TO ACCUM
                ;
                SETM:   ;SET MEMORY AT LOCATION PC TO VALUE IN ACCUM, INC PC
0151 2A1100             LHLD    TPC
0154 77                 MOV     M,A     ;STORE AT PC
0155 23                 INX     H       ;PC=PC+1
0156 221100             SHLD    TPC
0159 C9                 RET
                ;
                ;
                ;
                GETOP:  ;PROCESS NEXT OPCODE
015A CD8C06             CALL    CI
015D FE0D               CPI     CR
015F CA4005             JZ      GOBACK  ;RETURN IF SIMPLE INPUT
0162 FE2E               CPI     '.'     ;ALTERNATE RETURN IS .
0164 CA4005             JZ      GOBACK
0167 CD3C00             CALL    SCAN0
016A CA1005             JZ      ERR
                ;
                CHK0:   ;CHECK FOR OPCODES WITH NO OPERANDS
016D 0E11               MVI     C,17    ;LENGTH OF GROUP-0
016F 21A605             LXI     H,ETAB1 ;END OF GROUP-0
0172 114505             LXI     D,TABLE ;FIRST BYTE VALUE
0175 CDB700             CALL    SEAR    ;LOOK FOR MATCH
0178 C27E01             JNZ     CHK1    ;NO MATCH, CHECK FOR GROUP-1
                ;
```

```
                ;       MATCHED OPCODE, D,E ADDRESS BYTE VALUE
017B C35081             JMP     SETMD   ;SET MEMORY AT PC AND INC PC
                ;
                ;       CHECK GROUP-1 VALUES
017E 0E0A       CHK1.   MVI     C,10    ;LENGTH OF GROUP-1
0180 21CE05             LXI     H,ETAB2
0183 CDB700             CALL    SEAR    ;D,E REMAIN SET
0186 C29201             JNZ     CHK2    ;NO MATCH, CHECK NEXT GROUP
                ;
                ;       MATCH FOUND, SET BYTE AND GET BYTE OPERAND
0189 CD5081             CALL    SETMD
018C CD8C00             CALL    GBYTE   ;GETS BYTE VALUE TO ACCUMULATOR
018F C35101             JMP     SETM    ;PUTS BYTE VALUE TO MEMORY AT PC
                ;
                ;       CHECK GROUP-2 OPCODES, REQUIRE DOUBLE BYTE OPERAND
0192 0E06       CHK2.   MVI     C,6
0194 21E605             LXI     H,ETAB3
0197 CDB700             CALL    SEAR
019A C2AA01             JNZ     CHK3    ;NO MATCH
                ;
                ;       FOUND MATCH, GET OPCODE BIT PATTERN AND STORE
019D CD5081             CALL    SETMD
01A0 CD7D00     OP2.    ;ENTER HERE FOR DOUBLE BYTE OPERANDS
                        CALL    GADDR   ;VALUE IN B,A
01A3 CD5101             CALL    SETM
01A6 78                 MOV     A,B
01A7 C35101             JMP     SETM
                ;
01AA 0E01       CHK3.   ;CHECK FOR MOV INSTRUCTION
                        MVI     C,1
01AC 21EA05             LXI     H,PMOV
01AF CDB700             CALL    SEAR
01B2 C2C601             JNZ     CHK4
                ;
                ;       MOV INSTRUCTION GET DESTINATION OPERAND
01B5 CDDB00             CALL    GETREG  ;VALUE TO ACCUMULATOR
01B8 CD9300             CALL    ADJ
01BB 47                 MOV     B,A     ;SAVE IN B
01BC 0E40               MVI     C,01000000B     ;BIT PATTERN FOR MOV
                ;
01BE CDDB00     OP1.    ;GET NEXT OPERAND FOR MOV, FIRST OPERAND FOR ACCUM/REG OPER
                        CALL    GETREG
01C1 B1                 ORA     C       ;SETS HIGH ORDER TWO BITS
01C2 B0                 ORA     B       ;SETS DESTINATION/OPERATOR
01C3 C35101             JMP     SETM
                ;
01C6 0E08       CHK4.   ;CHECK FOR GROUP-5 (ACCUM/REG OPERATOR)
                        MVI     C,8
01C8 21BA06             LXI     H,ETAB5
01CB CDB700             CALL    SEAR
01CE C2DC01             JNZ     CHK5
                ;
                ;       ACCUM/REG INSTRUCTION, C COUNTS OPERATORS AS SEARCH PROCEED
01D1 0D                 DCR     C
01D2 79                 MOV     A,C
01D3 CD9300             CALL    ADJ
```



```
01D6 47                 MOV     B,A
                ;       OPERATOR NUMBER (SHIFTED) SAVED FOR LATER MASK
01D7 0E80               MVI     C,10000000B     ;ACCUM/REG OPERATOR INDICATOR
01D9 C3BE01             JMP     OP1     ;GETS OPERAND AND SAVES BYTE IN MEMORY
                ;
01DC 0E02       CHK5.   ;MAY BE INR/DCR
                        MVI     C,2
01DE 211206             LXI     H,PDCR
01E1 CDB700             CALL    SEAR
01E4 C2F401             JNZ     CHK6
                ;
                ;       C=2 IF DCR, =1 IF INR
01E7 0C                 INR     C       ;+1
01E8 0C                 INR     C       ;+2
01E9 0C                 INR     C       ;+3
01EA CDDB00             CALL    GETREG  ;VALUE TO ACCUM
01ED CD9300             CALL    ADJ
01F0 B1                 ORA     C       ;FILL PROPER INSTRUCTION INDICATOR
01F1 C35101             JMP     SETM
                ;
01F4 0E01       CHK6.   ;MAY BE A MVI INSTRUCTION
                        MVI     C,1
01F6 211606             LXI     H,PMVI
01F9 CDB700             CALL    SEAR
01FC C21002             JNZ     CHK7
                ;
                ;       MVI INSTRUCTION, GET REGISTER
01FF CDDB00             CALL    GETREG  ;VALUE GOES TO ACCUMULATOR
0202 CD9300             CALL    ADJ
0205 F606               ORI     110B
0207 CD5101             CALL    SETM
020A CD8C00             CALL    GBYTE
020D C35101             JMP     SETM
                ;
0210 0E06       CHK7.   ;CHECK FOR GROUP-7
                        MVI     C,6
0212 212E06             LXI     H,ETAB7
0215 CDB700             CALL    SEAR
0218 C23602             JNZ     CHK8
                ;
                ;       LXI,STAX,INX,DAD,LDA, OR DCX
021B 79                 MOV     A,C     ;A=1...6
021C FE04               CPI     4
021E DA2302             JC      IN0
                ;
                ;       MUST BE DAD,LDA, OR DCX
0221 C605               ADI     5       ;CHANGES ACCUM TO 9,10, OR 11
0223 47         IN0.    ;ACCUMULATOR CONTAINS CODE, SAVE IT
                        MOV     B,A
0224 CD8701             CALL    GETDR   ;DOUBLE REGISTER VALUE TO ACCUM
0227 CD9900             CALL    ADJ4    ;ADJUST VALUE TO MIDDLE FIELD
022A B0                 ORA     B       ;FILLS REMAINING BITS
022B CD5101             CALL    SETM
                ;       MAY BE LXI
022E E6CF               ANI     11001111B
0230 FE01               CPI     1
```

```
CP/M MACRO ASSEM 2.0     0009    CP/M DEBUGGER (ASMOD) 1/78

0232 C0                  RNZ     ;NOT LXI
0233 C3A001              JMP     OP2     ;PICK UP OPERAND
                 ;
                 ;
                 ;
                 CHK8.   ;RST?
0236 0E01                MVI     C,1
0238 213206              LXI     H,PRST
023B CDB700              CALL    SEAR
023E C25102              JNZ     CHK9
                 ;
                 ;       RST, GET OPERAND
0241 CD8C00              CALL    GBYTE
0244 FE08                CPI     8
0246 D21005              JNC     ERR
0249 CD9300              CALL    ADJ
024C F6C7                ORI     11000111B
024E C35101              JMP     SETM
                 ;
                 CHK9.   ;POP/PUSH?
0251 0E02                MVI     C,2
0253 213E06              LXI     H,PPOP+4
0256 CDB700              CALL    SEAR
0259 C27102              JNZ     CHK10
                 ;
                 ;       C=2 IF PUSH, 1 IF POP
025C 0D                  DCR     C
025D C26502              JNZ     PP0
                 ;
                 ;       POP, SET BIT PATTERN
0260 0EC1                MVI     C,11000001B
0262 C36702              JMP     PP1
                 ;
                 PP0.    ;PUSH
0265 0EC5                MVI     C,11000101B
0267 CD1001      PP1.    CALL    GETPR   ;DOUBLE PUSH/POP REGISTER TO PROPER FIELD
026A CD9900              CALL    ADJ4    ;MOVE TO FIELD
026D B1                  ORA     C
026E C35101              JMP     SETM    _
                 ;
                 CHK10.  ;J/C/R?
0271 3A7A06              LDA     OPCODE
0274 FE4A                CPI     'J'
0276 C28102              JNZ     CHK11
0279 CD4501              CALL    GCOHA
                 ;       CONDITION CODE TO FIELD IN ACCUM, ADDRESS TO B,C
027C F602                ORI     010B
027E C38B02              JMP     FADDR   ;FILL ADDRESS
                 ;
0281 FE43        CHK11.  CPI     'C'
0283 C29602              JNZ     CHK12
0286 CD4501              CALL    GCOHA
0289 F604                ORI     100B
                 ;
028B CD5101      FADDR.  CALL    SETM
028E 79                  MOV     A,C
```



```
CP/M MACRO ASSEM 2.0     0010    CP/M DEBUGGER (ASMOD) 1/78

028F CD5101              CALL    SETM
0292 78                  MOV     A,B
0293 C35101              JMP     SETM
                 ;
0296 FE52        CHK12.  CPI     'R'
0298 C21005              JNZ     ERR
029B CD1D01              CALL    GCOH
029E F6C0                ORI     11000000B
02A0 C35101              JMP     SETM
                 ;
                 ;       ********************************************************
                 ;       ********** END OF ASSEMBLER MODULE, START DISASSEMBLER **
                 ;       ********************************************************
                 ;
02A3 2A0E00      RDBYTE. LHLD    MPC
02A6 D5                  PUSH    D       ;SAVE DE
02A7 EB                  XCHG            ;MAX PC TO D,E
02A8 2A0C00              LHLD    PC      ;CURRENT PC
                 ;       SUBTRACT PC FROM MPC, STOP IF CARRY GENERATED
02AB 7B                  MOV     A,E
02AC 95                  SUB     L
02AD 7A                  MOV     A,D
02AE 9C                  SBB     H
02AF D2B702              JNC     RD0
                 ;
                 ;       PC EXCEEDS MPC, RETURN
02B2 2A1300              LHLD    OLDSP
02B5 F9                  SPHL            ;RESTORE ORIGINAL STACK POINTER
02B6 C9                  RET
                 ;
02B7 D1          RD0.    POP     D       ;RESTORE D,E
02B8 7E                  MOV     A,M
02B9 23                  INX     H
02BA 220C00              SHLD    PC
02BD C9                  RET

02BE 3C          RGPRNT. INR     A
02BF E607                ANI     07
02C1 FE06                CPI     06
02C3 DAC802              JC      RGP1
02C6 C603                ADI     03
02C8 FE05        RGP1.   CPI     05
02CA DACF02              JC      RGP2
02CD C602                ADI     02
02CF C641        RGP2.   ADI     41H
02D1 4F                  MOV     C,A
02D2 C31500              JMP     CO

02D5 47          DECODE. MOV     B,A
02D6 E6F0                ANI     0F0H
02D8 0F                  RRC
02D9 0F                  RRC
02DA 0F                  RRC
02DB 0F                  RRC
02DC C690                ADI     90H
02DE 27                  DAA
```

```
02DF CE40               ACI     40H
02E1 27                 DAA
02E2 4F                 MOV     C,A
02E3 CD1500             CALL    CO
02E6 78                 MOV     A,B
02E7 E60F               ANI     0FH
02E9 C690               ADI     90H
02EB 27                 DAA
02EC CE40               ACI     40H
02EE 27                 DAA
02EF 4F                 MOV     C,A
02F0 C31500             JMP     CO

02F3 0604       PRINT:  MVI     B,4
02F5 4E         P1:     MOV     C,M
02F6 CD1500             CALL    CO
02F9 23                 INX     H
02FA 05                 DCR     B
02FB C2F502             JNZ     P1
02FE 0E20               MVI     C,' '
0300 C31500             JMP     CO
                ;
                ;       EXTRACT THE REGISTER FIELD FROM THE OPCODE
0303 7A         XTRACT: MOV     A,D
0304 E638               ANI     00111000B
0306 0F                 RRC
0307 0F                 RRC
0308 0F                 RRC
0309 C9                 RET
                ;
                ;       PRINT CONDITION CODE
030A CD0303     CCPRNT: CALL    XTRACT
030D 87                 ADD     A
030E 4F                 MOV     C,A
030F 214206             LXI     H,CCODE
0312 09                 DAD     B
0313 4E                 MOV     C,M
0314 CD1500             CALL    CO
0317 23                 INX     H
0318 4E                 MOV     C,M
0319 CD1500             CALL    CO
031C 0E20               MVI     C,' '
031E CD1500             CALL    CO
0321 C31500             JMP     CO

                ;       PRINT REGISTER REFERENCE
0324 CD0303     RPPRNT: CALL    XTRACT
0327 E606               ANI     06
0329 FE06               CPI     06
032B C22E03             JNZ     RGPRNT
032E 0E53               MVI     C,'S'
0330 CD1500             CALL    CO
0333 0E50               MVI     C,'P'
0335 C31500             JMP     CO
                ;
                ;
```



```
                PRPC:   ;PRINT CRLF FOLLOWED BY PC VALUE
0338 CD2E00             CALL    CRLF
                ;       (ENTER HERE FROM DISASSEMBLER)
033B 2A0C00     PRPC0:  LHLD    PC
033E 7C                 MOV     A,H
033F CDD502             CALL    DECODE
0342 7D                 MOV     A,L
0343 CDD502             CALL    DECODE
0346 0E20               MVI     C,' '
0348 CD1500             CALL    CO
034B CD150A             CALL    CO
034E C9                 RET
                ;
                DISENT: ;ENTER HERE FROM DEBUGGER
034F 210000             LXI     H,0
0352 39                 DAD     SP
0353 221300             SHLD    OLDSP   ;SP SAVED FOR LATER RETURN
                ;
                ;       CHECK FOR PAGE MODE DISPLAY
0356 3A1000             LDA     PAGM    ;GET PAGE MODE (NUMBER OF LINES TO PRINT)
0359 B7                 ORA     A       ;SET FLAGS
035A CA7103             JZ      DISASM  ;NOT PAGE MODE
                ;
                ;       SET MPC TO 0FFFFH
035D 21FFFF             LXI     H,0FFFFH
0360 220E00             SHLD    MPC
                ;       255 IMPLIES TRACE MODE
0363 3C                 INR     A
0364 C27103             JNZ     DISASM  ;NOT TRACE MODE IF 0R
                ;       TRACE MODE, SET TO 1 AND IGNORE ADDRESS FIELD
0367 3C                 INR     A       ;1 IN ACC
0368 321000             STA     PAGM
036B 2A0C00             LHLD    PC      ;RECOVER PC
036E C39703             JMP     DIS1
                ;
                ;
                DISASM:
                ;       CHECK FOR BREAK AT CONSOLE
0371 CD9E06             CALL    BREAK
0374 C24005             JNZ     GOBACK
                ;
                ;       CHECK TO SEE IF ENOUGH LINES PRINTED IN PAGE MODE
0377 211000             LXI     H,PAGM
037A 7E                 MOV     A,M
037B B7                 ORA     A       ;ZERO?
037C CA8303             JZ      DIS0    ;JMP IF NOT PAGE MODE
                ;
                ;       PAGE MODE, DECREMENT AND CHECK FOR ZERO
037F 35                 DCR     M
0380 CA4005             JZ      GOBACK
                ;
0383 2A0C00     DIS0:   LHLD    PC      ;CURRENT PC
0386 CDA106             CALL    PRLABEL ;OPTIONAL LABEL
0389 CD2E00             CALL    CRLF    ;NEW LINE
038C 0E20               MVI     C,' '
038E CD1500             CALL    CO
```

```
0391 CD1508          CALL    CD      ;TWO LEADING BLANKS
0394 CD3B03          CALL    PRPC0   ;PRINT THE VALUE
0397 CDA302  DIS1.   CALL    RDBYTE
             ;       SAVE THE OPCODE IN THE D REGISTER
039A 57              MOV     D,A
             ;       SEARCH THE FIRST 17 ITEMS FOR SIMPLE OPCODES
             ;       EI (FB) THROUGH NOP (00).  NOTE THAT THE SEARCH PROCEEDS
             ;       THROUGH "TABLE" STARTING AT THE BEGINNING, BUT THE OPCODES
             ;       ARE ACTUALLY STORED IN SYMBOLIC FORM IN REVERSE ORDER.
             ;
039B 214585          LXI     H,TABLE
039E 011100          LXI     B,17    ;FIRST 17 SIMPLE OPCODES
03A1 BE      GROUP1. CMP     M       ;TABLE VALUE = OPCODE?
03A2 CAFD04          JZ      TYPE1   ;SKIP TO PRINT IF SO
03A5 23              INX     H       ;MOVE TO THE NEXT TABLE ELEMENT
03A6 0D              DCR     C       ;COUNT THE SIMPLE OPCODES DOWN
03A7 C2A103          JNZ     GROUP1  ;TRY FOR ANOTHER
             ;
             ;       NOT A SIMPLE OPERATION CODE, CHECK FOR IMMEDIATE OP
             ;       ADI, ACI, OUT, SUI, IN, SBI, ANI, XRI, ORI, CPI
03AA 0E0A            MVI     C,10
03AC BE      GROUP2. CMP     M
03AD CAE904          JZ      TYPE2
03B0 23              INX     H
03B1 0D              DCR     C
03B2 C2AC03          JNZ     GROUP2
             ;
             ;       NOT AN IMMEDIATE OPERATION, CHECK FOR MOV
             ;       SHLD LHLD STA LDA JMP OR CALL
03B5 0E06            MVI     C,6
03B7 BE      GROUP3. CMP     M
03B8 CACE04          JZ      TYPE3
03BB 23              INX     H
03BC 0D              DCR     C
03BD C2B703          JNZ     GROUP3
             ;
             ;       NOT TYPE3 OPERATION CODE, CHECK FOR MOV
             ;       BY MASKING THE HIGH ORDER TWO FIBITS -
             ;       XX00 0000 IS PRODUCED IN THE ACCUMULATOR
03C0 E6C0            ANI     0C0H
             ;       MOV IS GIVEN BY 01 DDD SSS (DDD IS DEST, SSS IS SOURCE)
03C2 FE40            CPI     40H
03C4 CAB404          JZ      MOVOP
             ;
             ;       NOT A MOV INSTRUCTION, CHECK FOR ACCUMLATOR-REGISTER OPS
             ;       BIT PATTERN 10 CCC RRR    CORRESPONDS TO
             ;       ADD (0), ADC (1), SUB (2), SBB (3), ANA (4),
             ;       XRA (5), ORA (6), CMP (7)
03C7 FE80            CPI     80H
03C9 CAA504          JZ      ACCREG
             ;
             ;       NOT ACCUM-REGISTER, RESTORE OPCODE FOR FURTHER CHECKS
03CC 7A              MOV     A,D
             ;
             ;       LOOK FOR INR, DCR, AND MVI OPERATIONS
             ;       INR = 00 RRR 100, DCR = 00 RRR 101, MVI = 00 RRR 110
```



```
             ;       OR, INR = 00 RRR 4, DCR = 00 RRR 5, MVI = 00 RRR 6
03CD E6C7            ANI     11000111B
03CF D604            SUI     04
             ;       INR GOES TO ZERO
03D1 CA9604          JZ      INRREG
             ;       NOT INR, MAY BE DCR
03D4 3D              DCR     A
03D5 CA9004          JZ      DCRREG
03D8 3D              DCR     A
             ;       NOT DCR, MAY BE MVI
03D9 CA7C04          JZ      MVIREG
             ;       NOT INR, DCR, OR MVI INSTRUCTION
             ;
             ;       RESTORE THE OPCODE
03DC 7A              MOV     A,D
             ;       LOOK FOR LXI STAX INX DAD LDAX DCX OPCODES
             ;       LXI = 00 RR 0001,
             ;       STAX= 00 RR 0010,
             ;       INX = 00 RR 0011,
             ;       DAD = 00 RR 1001,
             ;       LDAX= 00 RR 1010
             ;       DCX = 00 RR 1011
03DD E6C0            ANI     0C0H
03DF CA4A04          JZ      LXILST  ;TO PROCESS FURTHER
             ;
             ;       NOT ONE OF THE ABOVE, CHECK FURTHER
             ;       MUST BE OF THE FORM - 11 XXX XXX
03E2 7A              MOV     A,D
03E3 E607            ANI     00001111B       ;TO EXTRACT THE RIGHTMOST BITS
             ;       RETURN CONDITIONALS TAKE THE FORM 11 XXX 000
03E5 CA3F04          JZ      RETCON          ;RETURN CONDITIONALLY
             ;       JUMP CONDITIONALS TAKE THE FORM 11 XXX 010 = 2
03E8 D602            SUI     02
03EA CA3404          JZ      JMPCON
             ;       CALL CONDITIONALS TAKE THE FORM 11 XXX 100 = 4 - 2 = 2
03ED D602            SUI     02
03EF CA2904          JZ      CALLCON
             ;       RST'S TAKE THE FORM 11 XXX 111 = 7 - 4 = 3
03F2 D603            SUI     03
03F4 CA1A04          JZ      RSTOP
             ;
             ;       NONE OF THE ABOVE, PUSHES AND POP'S REMAIN
03F7 7A              MOV     A,D             ;RESTORE OPCODE
             ;       FIRST CAPTURE REMAINING OPCODES CB, D9, DD, ED, FD
03F8 E608            ANI     00001000B       ;THIS BIT RESET FOR POP,PUSH
03FA C20B05          JNZ     N8080           ;NOT 8080 OPCODE IF SET
03FD 7A              MOV     A,D             ;RESTORE IT
             ;       PUSH = 11 XX0 101 = 5, POP = 11 XX0 001 = 1
03FE E607            ANI     07
             ;       USE THE RESULTING VALUE TO INDEX TO REGISTER TABLE
0400 4F              MOV     C,A
0401 3D              DCR     A       ;POP GOES TO 00
0402 213906          LXI     H,PPOP-1
0405 09              DAD     B
0406 CDF302          CALL    PRINT
             ;       GET THE RELEVANT REGISTER
```

```
0489 CD0383             CALL    XTRACT
                ;       CHECK FOR PSW OPERATION CODE
048C FEB6               CPI     86
048E C29F04             JNZ     D6
0411 213606             LXI     H,PPSW
0414 CDF302             CALL    PRINT
0417 C37103             JMP     DISASM
                ;
                ;       PRINT RST XXX INSTRUCTION
041A 213206     RSTOP,  LXI     H,PRST
041D CDF302             CALL    PRINT
0420 CD0383             CALL    XTRACT
0423 CDD502             CALL    DECODE
0426 C37103             JMP     DISASM
                ;
                ;       CALL CONDITIONAL 'C'
                CALLCON,
0429 0E43               MVI     C,'C'
042B CD1500             CALL    CO
042E CD0A03             CALL    CCPRNT
0431 C3D904             JMP     PREXT   ;TO PRINT THE ADDRESS
                ;
                ;       JUMP CONDITIONAL 'J'
                JMPCON,
0434 0E4A               MVI     C,'J'
0436 CD1500             CALL    CO
0439 CD0A03             CALL    CCPRNT
043C C3D904             JMP     PREXT   ;TO PRINT THE ADDRESS
                ;
                ;       RETURN CONDITIONAL 'R'
                RETCON,
043F 0E52               MVI     C,'R'
0441 CD1500             CALL    CO
0444 CD0A03             CALL    CCPRNT
0447 C37103             JMP     DISASM
                ;
                ;
                ;       PROCESS ONE OF LXI STAX INX DAD LDAX DCX
044A 211A06     LXILST, LXI     H,PLXI
                ;       CAPTURE 00, 10, 10, 20, 20, 30, AND 30
044D 7A                 MOV     A,D     ;GET OPCODE
044E E607               ANI     1110    ;RIGHTMOST BITS ZERO?
0450 CA0B05             JZ      M0000   ;NOT 0000 IF SO
                ;       RECALL OPCODE TO DETERMINE WHICH ONE
0453 7A                 MOV     A,D
                ;       FIND THE PARTICULAR OPCODE
0454 E60F               ANI     0FH
                ;       LXI HAS LEAST SIGNIFICANT FOUR BITS = 0001
0456 3D                 DCR     A
0457 CA6E04             JZ      LXIREG
                ;       STAX 0010 BECOMES 0001 = 1
                ;       INX  0011 BECOMES 0010 = 2
                ;       DAD  1001 BECOMES 1000 = 8
                ;       LDAX 1010 BECOMES 1001 = 9
                ;       DCX  1011 BECOMES 1010 = 10
045A FE03               CPI     03
```



```
045C DA6104             JC      D4
                ;       MUST BE DAD, LDAX OR DCX
045F D605               SUI     05
                ;       DAD 8 BECOMES 3
                ;       LDAX9 BECOMES 4
                ;       DCX10 BECOMES 5
                ;       ACCUMULATOR NORMALIZED
0461 87         D4,     ADD     A
0462 87                 ADD     A
                ;       VALUE IN ACCUM MULTIPLIED BY FOUR
0463 4F                 MOV     C,A
0464 09                 DAD     B
0465 CDF302             CALL    PRINT
                ;       STAX, INX, DAD, LDAX, OR DC X PRINTED, PRINT REGISTER
0468 CD2403             CALL    RPPRNT
046B C37103             JMP     DISASM
                ;
                ;       PRINT REGISTER ADDRESSED BY HL (E.G., IN LXI)
046E CDF302     LXIREG, CALL    PRINT
0471 CD2403             CALL    RPPRNT
0474 0E2C               MVI     C,','
0476 CD1500             CALL    CO
0479 C3D904             JMP     PREXT   ;TO PRINT THE EXTENDED INSTRUCTION
                ;
                ;
047C 211606     MVIREG, LXI     H,PMVI
047F CDF302             CALL    PRINT
0482 CD0383             CALL    XTRACT
0485 CD8E02             CALL    RGPRNT
0488 0E2C               MVI     C,','
048A CD1500             CALL    CO
048D C3F404             JMP     DATA8
                ;
0490 211206     DCRREG, LXI     H,PDCR
0493 C39904             JMP     D5
                ;
0496 210E06     INRREG, LXI     H,PINR
                ;
                ;       PRINT THE INSTRUCTION GIVEN BY HL, FOLLOWED BY REGISTER
0499 CDF302     D5,     CALL    PRINT
049C CD0383             CALL    XTRACT
049F CD8E02     D6,     CALL    RGPRNT
04A2 C37103             JMP     DISASM
                ;
                ;       FOUND ACCUM REGISTER OPERATION - MIDDLE BITS GIVE PCODE
04A5 7A         ACCREG, MOV     A,D
04A6 E638               ANI     38H     ;SELECT OPCODE BITS
04A8 0F                 RRC             ;OPCODE * 4 FOR LENGTH FOUR STRING
04A9 4F                 MOV     C,A
04AA 21EE05             LXI     H,PADD  ;ADDRESS THE ACCUM-REGISTER LIST
04AD 09                 DAD     B
04AE CDF302             CALL    PRINT
04B1 C3C504             JMP     D9
                ;
                ;       MOV OPERATION FOUND
04B4 21EA05     MOVOP,  LXI     H,PMOV
```

```
CP/M MACRO ASSEM 2.0     #017    CP/M DEBUGGER (ASMOD) 1/70

04B7 CDF302              CALL    PRINT
04BA CD0303              CALL    XTRACT
04BD CD8E02              CALL    RGPRNT
04C0 0E2C                MVI     C,','   ;REGISTER DELIMITER
04C2 CD1500              CALL    CO
04C5 7A          D9.     MOV     A,D
04C6 E607                ANI     07
04C8 CD8E02              CALL    RGPRNT
04CB C37103              JMP     DISASM
                 ;
                 ;       TYPE GROUP3, CALL JMP LDA STA LHLD SHLD
                 TYPE3,
04CE 79                  MOV     A,C     ;*4 FOR LENGTH 4
04CF 87                  ADD     A
04D0 87                  ADD     A
04D1 4F                  MOV     C,A
04D2 21CE05              LXI     H,TAB3-4
04D5 09                  DAD     B
04D6 CDF302              CALL    PRINT
                 ;
                 ;       ARRIVE HERE TO PRINT THE ADDRESS FIELD
04D9 CDA302      PREXT.  CALL    RDBYTE  ;LOW ADDRESS TO A
04DC F5                  PUSH    PSW     ;SAVE IT
04DD CDA302              CALL    RDBYTE
04E0 57                  MOV     D,A     ;SET HIGH ADDRESS
04E1 F1                  POP     PSW     ;RECALL LOW ADDRESS
04E2 5F                  MOV     E,A     ;DE IS THE ADDRESS TO PRINT
04E3 CD9506              CALL    PADDX
04E6 C37103              JMP     DISASM
                 ;
                 ;       TYPE THE IMMEDIATE OPCODES (INCLUDING IN/OUT)
04E9 79          TYPE2.  MOV     A,C     ;TYPE IMMEDIATE OPERATION CODE
04EA 87                  ADD     A       ;*2
04EB 87                  ADD     A       ;*4 FOR LENGTH FOUR CHAR STRING
04EC 4F                  MOV     C,A     ;BC = INDEX * 4 FOR OPCODE
04ED 21A605              LXI     H,TAB2-4
04F0 09                  DAD     B
04F1 CDF302              CALL    PRINT
                 ;
                 ;       ARRIVE HERE TO PRINT THE IMMEDIATE VALUE
04F4 CDA302      DATA8.  CALL    RDBYTE
04F7 CD9206              CALL    PBYTE   ;BYTE VALUE PRINTED
04FA C37103              JMP     DISASM
                 ;
                 ;       FOUND OPCODE IN TABLE, POSITION GIVEN
                 ;       BY COUNT IN BC (NOTE THAT C IS COUNTED DOWN, WHILE
                 ;       INDEX WAS MOVING UP THE TABLE DURING THE SEARCH)
04FD 79          TYPE1.  MOV     A,C     ;TYPE SIMPLE OPCODES FROM GROUP 1
04FE 87                  ADD     A       ;POSITION * 2
04FF 87                  ADD     A       ;POSITION * 4 (FOUR CHAR CODES)
0500 4F                  MOV     C,A     ;BC IS INDEX * 4 OF OPCODE
0501 216205              LXI     H,TAB1-4
0504 09                  DAD     B       ;HL NOW HOLDS ADDRESS OF CODE TO PRINT
0505 CDF302              CALL    PRINT
0508 C37103              JMP     DISASM
                 ;
```


COPYRIGHT © 1978
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA. 93950
SER. # _____

```
CP/M MACRO ASSEM 2.0     #018    CP/M DEBUGGER (ASMOD) 1/70

                 M0000.  ;NOT AN 0000 OPERATION CODE
050B 217606              LXI     H,0BOP
050E CDF302              CALL    PRINT   ;PRINT THE '??='
0511 7A                  MOV     A,D     ;GET THE OPCODE
0512 CD9206              CALL    PBYTE   ;AND PRINT IT
0515 C37103              JMP     DISASM
                 ;
                 ERR.    ;ENTER HERE FOR ERROR REPORTING
0518 CD2E00              CALL    CRLF
051B 0E3F                MVI     C,'?'
051D CD1500              CALL    CO
                 ;
0520 2A1300              LHLD    OLDSP
0523 F9                  SPHL
                 ;       PC REMAINS UNCHANGED
                 ;
                 ;
                 ASMEN.  ;ENTER HERE FROM DEBUGGER
0524 210000              LXI     H,0
0527 39                  DAD     SP
0528 221300              SHLD    OLDSP
                 ;
052B CD3003      ASM0.   CALL    PRPC    ;PRINT PC VALUE
052E 221100              SHLD    TPC     ;SAVE PC VALUE
0531 CD0906              CALL    GETBUFF ;FILL INPUT BUFFER
0534 CD5A01              CALL    GETOP   ;GET OPERATION
                 ;       UPDATE PC, MUST BE CORRECT INPUT
0537 2A1100              LHLD    TPC
053A 220C00              SHLD    PC
053D C32B05              JMP     ASM0
                 ;
0540 2A1300      GOBACK. LHLD    OLDSP
0543 F9                  SPHL
0544 C9                  RET
                 ;
                 ;       THE FIRST 17 ITEMS CORRESPOND TO SIMPLE OPCODES
                 ;       (NOP BACKWARD THROUGH EI)
0545 0007BF17    TABLE.  DB      000H,007H,00FH,017H     ;NOP RLC RRC RAL
0549 1F272F37            DB      01FH,027H,02FH,037H     ;RAR DAA CMA STC
054D 3F76C9E3            DB      03FH,076H,0C9H,0E3H     ;CMC HLT RET XTHL
0551 E9EBF3F9            DB      0E9H,0EBH,0F3H,0F9H     ;PCHL XCHG DI SPHL
0555 FB                  DB      0FBH                    ;EI
                 ;
                 ;       THE NEXT 10 ITEMS CORRESPOND TO THE IMMEDIATE OPCODES
0556 C6CED3              DB      0C6H,0CEH,0D3H          ;ADI ACI OUT
0559 D6DBDEE6            DB      0D6H,0DBH,0DEH,0E6H     ;SUI IN SBI ANI
055D EEF6FE              DB      0EEH,0F6H,0FEH          ;XRI ORI
0560 22                  DB      022H                    ;SHLD
0561 2A323AC3            DB      02AH,032H,03AH,0C3H
0565 CD                  DB      0CDH
0566 4549202053TAB1.     DB      'EI  ','SPHL','DI  ','XCHG'


0576 5043484C58          DB      'PCHL','XTHL','RET ','HLT '
```

```
0586 434D432053          DB      'CMC ','STC ','CMA ','DAA '

0596 5241522052          DB      'RAR ','RAL ','RRC ','RLC '

05A6 4E4F5020  ETAB1.    DB      'NOP '
05AA 435049204FTAB2.     DB      'CPI ','ORI ','XRI ','ANI '

05BA 5342492049          DB      'SBI ','IN  ','SUI ','OUT '

05CA 41434920            DB      'ACI '
05CE 41444920  ETAB2.    DB      'ADI '
                ;
05D2 43414C4C4ATAB3.     DB      'CALL','JMP ','LDA ','STA '

05E2 4C484C44            DB      'LHLD'
05E6 53484C44  ETAB3.    DB      'SHLD'
                ;
05EA 4D4F5620  PMOV.     DB      'MOV '
05EE 4144442041PADD.     DB      'ADD ','ADC ','SUB ','SBB '

05FE 414E412058          DB      'ANA ','XRA ','ORA '
0608 434D5020  ETAB5.    DB      'CMP '

060E 494E5220  PINR.     DB      'INR '
0612 44435220  PDCR.     DB      'DCR '
0616 4D564920  PMVI.     DB      'MVI '
061A 4C5849205PLXI.      DB      'LXI ','STAX','INX ','DAD '

062A 4C444158            DB      'LDAX'
062E 44435820  ETAB7.    DB      'DCX '
                ;
0632 52535420  PRST.     DB      'RST '
0636 50535720  PPSW.     DB      'PSW '
063A 504F502050PPOP.     DB      'POP ','PUSH'

0642 4E5A5A204ECCODE.    DB      'NZ','Z ','NC','C '

064A 504F504550          DB      'PO','PE','P '
0650 4D20        CREG.   DB      'M '

0652 4220432044          DB      'B ','C ','D ','E '
065A 48204C204D          DB      'H ','L ','M '
0660 4120        SREG.   DB      'A '
                ;
0662 4220202044          DB      'B   ','D   ','H   ','SP  '
```



```
0672 50535720  DREG.     DB      'PSW '
                ;
0676 3F3F3D20  DBOP.     DB      '??= '
067A           OPCODE.   DS      4
067E                     END
```

PAGE 145
MISSING

```
CP/M MACRO ASSEM 2.0    #021    CP/M DEBUGGER (ASMDB) 1/78

04A5 ACCREG     0093 ADJ        0099 ADJ4       052B ASM0       0324 ASMEN
0683 BEGIN      069E BREAK      0429 CALLCON    0642 CCODE      030A CCPRNT
016D CHK0       017E CHK1       0271 CHK10      0281 CHK11      0296 CHK12
0192 CHK2       01AA CHK3       01C6 CHK4       01DC CHK5       01F4 CHK6
0210 CHK7       0236 CHK8       0251 CHK9       068C CI         0015 CO
000D CR         0650 CREG       002E CRLF       0461 D4         0499 D5
049F D6         04C5 D9         04F4 DATA0      0676 DBOP       0490 DCRREG
0000 DEBUG      02D5 DECODE     001C DELIM      0680 DEMON      0383 DIS0
0397 DIS1       0371 DISASM     034F DISENT     0672 DREG       0510 ERR
05A6 ETAB1      05CE ETAB2      05E6 ETAB3      060A ETAB5      062E ETAB7
0200 FADDR      0000 FALSE      007D GADDR      008C GBYTE      011D GCON
0145 GCONA      0689 GETBUFF    00F1 GETD       0187 GETDR      015A GETOP
0110 GETPR      00DB GETREG     060C GHC        0540 GOBACK     03A1 GROUP1
03AC GROUP2     03B7 GROUP3     0070 HEX        0223 IN0        0496 INRREG
0005 JLOC1      0434 JMPCON     000A LF         044A LXILST     046E LXIREG
0000 MODLOC     0125 MOP        04B4 MOVDP      000E MPC        047C MVIREG
0500 N8080      0013 OLDSP      01BE OP1        01A0 OP2        067A OPCODE
02F5 P1         05EE PADD       0695 PADDX      0010 PAGM       0692 PBYTE
060F PCHAR      000C PC         0612 PDCR       060E PIMR       061A PLXI
05EA PMOV       0616 PMVI       0265 PP0        0267 PP1        063A PPOP
0636 PPSW       04D9 PREXT      02F3 PRINT      06A1 PRLABEL    0338 PRPC
0338 PRPC0      0632 PRST       02B7 RD0        02A3 RDBYTE     FFFF RELOC
043F RETCON     02C0 RGP1       02CF RGP2       028E RGPRNT     0324 RPPRNT
041A RSTOP      004C SC0        0039 SC1        0058 SC2        0B6A SC3
003C SCAN0      0698 SCANEXP    0039 SCAN       000D SE1        00CA SE2
00A5 SEA0       00AF SEA1       00A0 SEAR2      0007 SEAR       0150 SETMD
0151 SETM       0660 SREG       0009 TAB        0566 TAB1       05AA TAB2
05D2 TAB3       0543 TABLE      0011 TPC        FFFF TRUE       04FD TYPE1
04E9 TYPE2      04CE TYPE3      0303 XTRACT
```

```
CP/M MACRO ASSEM 2.0    #001    CP/M DEBUGGER (DEMON) 1/78

                                TITLE   'CP/M DEBUGGER (DEMON) 1/78'
                        ;       CP/M DEBUGGER VERSION 1.4
                        ;
                        ;       COPYRIGHT (C) 1978
                        ;       DIGITAL RESEARCH
                        ;       BOX 579 PACIFIC GROVE
                        ;       CALIFORNIA 93950
                        ;
0000 =          FALSE   EQU     0
FFFF =          TRUE    EQU     NOT FALSE
0000 =          DEBUG   EQU     FALSE   ;TRUE IF DEBUGGING
FFFF =          RELOC   EQU     TRUE    ;TRUE IF RELOCATING
                        IF      DEBUG
                        ORG     1000H
                        ELSE
                        IF      RELOC
0000                    ORG     0000H
                        ELSE
                        ORG     0D000H  ;TESTING IN 64K
                        ENDIF
                        ENDIF
                        ;
0000 =          MODBAS  EQU     $       ;BASE OF ASSEM/DISASSEM MODULE
0000                    DS      680H            ;SIZE OF ASSEM/DISASSEM
0680 =          DEMON   EQU     $               ;BASE OF DEMON MODULE
0003 =          DISIN   EQU     MODBAS+3
1606 =          BDOS    EQU     $+1006H
0005 =          BDOSE   EQU     5H      ;ENTRY POINT TO DOS FROM USER PROGRAMS
0100 =          PCBASE  EQU     100H    ;DEFAULT PC
0100 =          SPBASE  EQU     100H    ;DEFAULT SP
0006 =          DISEN   EQU     DISIN+3         ;DISASSEMBLER ENTRY POINT
0009 =          ASSEM   EQU     DISEN+3 ;ASSEMBLER ENTRY POINT
000C =          DISPC   EQU     ASSEM+3         ;DISASSEMBLER PC VALUE
000E =          DISPM   EQU     DISPC+2         ;DISASSEMBLER PC MAX VALUE
0010 =          DISPG   EQU     DISPM+2         ;DISASSEMBLER PAGE MODE IF NON Z
000C =          PSIZE   EQU     12              ;NUMBER OF ASSEMBLY LINES TO LIS
0020 =          CSIZE   EQU     32              ;COMMAND BUFFER SIZE
0032 =          SSIZE   EQU     50              ;LOCAL STACK SIZE
                        ;
                        ;       BASIC DISK OPERATING SYSTEM CONSTANTS
0001 =          CIF     EQU     1
0002 =          COF     EQU     2
0003 =          RIF     EQU     3
0004 =          POF     EQU     4
0005 =          LOF     EQU     5
                        ;
0007 =          IDS     EQU     7
000A =          GETF    EQU     10      ;FILL BUFFER FROM CONSOLE
000B =          CHKIO   EQU     11      ;CHECK IO STATUS
000C =          LIFT    EQU     12      ;LIFT HEAD ON DISK
000F =          OPF     EQU     15      ;DISK FILE OPEN
0014 =          RDF     EQU     20      ;READ DISK FILE
001A =          DMAF    EQU     26      ;SET DMA ADDRESS
                        ;
005B =          DBP     EQU     5BH     ;DISK BUFFER POINTER
0080 =          DBF     EQU     80H     ;DISK BUFFER ADDRESS
```

```
005C =          DFCB    EQU     5CH     ;DISK FILE CONTROL BLOCK
005C =          FCB     EQU     DFCB
0000 =          FDN     EQU     0       ;DISK NAME
0001 =          FFN     EQU     1       ;FILE NAME
0009 =          FFT     EQU     9       ;FILE TYPE
000C =          FRL     EQU     12      ;REEL NUMBER
000F =          FRC     EQU     15      ;RECORD COUNT
0020 =          FCR     EQU     32      ;CURRENT RECORD
0021 =          FLN     EQU     33      ;FCB LENGTH
                ;
001A =          DEOF    EQU     1AH     ;CONTROL-Z (EOF)
000D =          CR      EQU     0DH
000A =          LF      EQU     0AH
                ;
                        IF      DEBUG
                RSTNUM  EQU     6       ;USE 6 IF DEBUGGING
                        ELSE
0007 =          RSTNUM  EQU     7       ;RESTART NUMBER
                        ENDIF
0038 =          RSTLOC  EQU     RSTNUM*8        ;RESTART LOCATION
00FF =          RSTIN   EQU     0C7H OR (RSTNUM SHL 3)  ;RESTART INSTRUCTION
                ;
                ;       TEMPLATE FOR PROGRAMMED BREAKPOINTS
                ;               ---------
                ;               PCH , PCL
                ;               HLH , HLL
                ;               SPH , SPL
                ;               RA  , FLG
                ;               B   , C
                ;               D   , E
                ;               ---------
                ;       FLG FIELD:  MZ0IDE1C (MINUS, ZERO, IDC, EVEN, CARRY)
                ;
0005 =          AVAL    EQU     5       ;A REGISTER COUNT IN HEADER
0006 =          BVAL    EQU     6
0007 =          DVAL    EQU     7
0008 =          HVAL    EQU     8
0009 =          SVAL    EQU     9
000A =          PVAL    EQU     10
                ;
                ;
                ;       DEMON ENTRY POINTS
0680 C3A286             JMP     TRAPAD  ;TRAP ADDRESS FOR RETURN IN CASE INTERRUPT
0683 C3AA86             JMP     BEGIN
                BREAKA:
0686 C3C98D             JMP     BREAKP
                ;       USEFUL ENTRY POINTS FOR PROGRAMS RUNNING WITH DDT
0689 C3B08B             JMP     GETBUFF ;GET ANOTHER BUFFER FULL
068C C3D78B             JMP     GNC     ;GET NEXT CHARACTER
068F C3C18B             JMP     PCHAR   ;PRINT A CHARACTER FROM A
0692 C3FF8B             JMP     PBYTE   ;PRINT BYTE IN REGISTER A
0695 C3278C             JMP     PADDX   ;PRINT ADDRESS IN REGISTERS D,E
0698 C3BA8C             JMP     SCANEXP ;SCAN 0,1,2, OR 3 EXPRESSIONS
069B C3688C             JMP     GETVAL  ;GET VALUE TO H,L
069E C3198C             JMP     BREAK   ;CHECK BREAK KEY
06A1 C9                 RET             ;TAKES PLACE OF PRLABEL IN SID
```



```
                ;
                ;
                TRAPAD: ;GET THE RETURN ADDRESS FOR THIS JUMP TO BDOS IN CASE OF
                ;       A SOFT INTERRUPT DURING BDOS PROCESSING.
06A2 E3                 XTHL    ;PC TO HL
06A3 22448F             SHLD    RETLOC  ;MAY NOT NEED IT
06A6 E3                 XTHL
                TRAPJMP:        ;ADDRESS FILLED AT "BEGIN"
06A7 C30000             JMP     0000H
                ;
                BEGIN:
                ;
06AA 2A0600             LHLD    BDOSE+1
06AD 22A806             SHLD    TRAPJMP+1       ;FILL JUMP TO BDOS
06B0 21A206             LXI     H,TRAPAD
06B3 220100             SHLD    MODBAS+1        ;ADDRESS FIELD CHANGED
06B6 210000             LXI     H,MODBAS
06B9 220600             SHLD    BDOSE+1         ;NOW INCLUDES ASSEM/DISASSEM
                ;
06BC AF                 XRA     A       ;ZERO TO ACCUM
06BD 32498F             STA     BREAKS  ;CLEARS BREAK POINT COUNT
                ;
06C0 210081             LXI     H,PCBASE
06C3 220C00             SHLD    DISPC           ;INITIAL VALUE FOR DISASSEMBLER
06C6 22578F             SHLD    DISLOC          ;INITIAL VALUE FOR DISPLAY
06C9 22818F             SHLD    MLOAD           ;MAX LOAD LOCATION
                ;
                ;       SETUP RESTART TEMPLATE
06CC 22B38F             SHLD    PLOC
06CF 210081             LXI     H,SPBASE
06D2 31B18F             LXI     SP,STACK-4
06D5 E5                 PUSH    H       ;INITIAL SP
06D6 210200             LXI     H,10B   ;INITIAL PSW
06D9 E5                 PUSH    H
06DA 2B                 DCX     H
06DB 2B                 DCX     H       ;CLEARED
06DC 22B18F             SHLD    HLOC    ;H,L CLEARED
06DF E5                 PUSH    H       ;B,C CLEARED
06E0 E5                 PUSH    H       ;D,E CLEARED
06E1 22478F             SHLD    TRACER  ;CLEAR TRACE FLAG
                ;
06E4 3EC3               MVI     A,0C3H  ;(JMP RESTART)
06E6 323800             STA     RSTLOC
06E9 218606             LXI     H,BREAKA        ;BREAK POINT SUBROUTINE
06EC 223900             SHLD    RSTLOC+1        ;RESTART LOCATION ADDRESS FIELD
                ;
                ;       CHECK FOR FILE NAME PASSED TO DEMON, AND LOAD IF PRESENT
06EF 3A5D00             LDA     FCB+FFH ;BLANK IF NO NAME PASSED
06F2 FE20               CPI     ' '
06F4 CAFE06             JZ      START
                ;
                ;       PUSH A ZERO, AND READ
06F7 210000             LXI     H,0
06FA E5                 PUSH    H
06FB C3A709             JMP     RINIT
                ;
```

```
                ;
                ;       MAIN COMMAND LOOP
                ;
                START:
06FE 31A98F             LXI     SP,STACK-12     ;INITIALIZE SP IN CASE OF ERROR
                ;       CHECK FOR DISASSEMBLER OVERLOAD
0701 CD8D89             CALL    CHKDIS
0704 DA0D07             JC      DISASMOK
                ;
                ;       DISASSEMBLER NOT PRESENT, SET BDOS JMP
0707 218086             LXI     H,DEMON
070A 220600             SHLD    BDOSE+1 ;(RE)SET JMP ADDRESS
                DISASMOK:
070D CD0F8C             CALL    CRLF    ;INITIAL CRLF
                        IF      DEBUG
                        MVI     A,'.'
                        ELSE
0710 3E2D               MVI     A,'-'
                        ENDIF
0712 CDC10B             CALL    PCHAR   ;OUTPUT PROMPT
                ;
                ;       GET INPUT BUFFER
0715 CDB08B             CALL    GETBUFF ;FILL COMMAND BUFFER
                ;
0718 CDD70B             CALL    GNC     ;GET CHARACTER
071B FE0D               CPI     CR
071D CAFE06             JZ      START
0720 D641               SUI     'A'     ;LEGAL CHARACTER?
0722 DAA50B             JC      CERROR  ;COMMAND ERROR
0725 FE1A               CPI     'Z'-'A'+1
0727 D2A50B             JNC     CERROR
                ;       CHARACTER IN REGISTER A IS COMMAND, MUST BE IN THE RANGE A-
072A 5F                 MOV     E,A     ;INDEX TO E
072B 1600               MVI     D,0     ;DOUBLE PRECISION INDEX
072D 213707             LXI     H,JMPTAB;BASE OF TABLE
0730 19                 DAD     D
0731 19                 DAD     D       ;INDEXED
0732 5E                 MOV     E,M     ;LO BYTE
0733 23                 INX     H
0734 56                 MOV     D,M     ;HO BYTE
0735 EB                 XCHG            ;TO H,L
0736 E9                 PCHL    ;GONE...
                ;
                JMPTAB: ;JUMP TABLE TO SUBROUTINES
0737 7E07               DW      ASSM    ;A ENTER ASSEMBLER LANGUAGE
0739 A50B               DW      CERROR  ;B
073B A50B               DW      CERROR  ;C
073D C607               DW      DISPLAY ;D DISPLAY RAM MEMORY
073F A50B               DW      CERROR  ;E
0741 5C08               DW      FILL    ;F FILL MEMORY
0743 7008               DW      GOTO    ;G GO TO MEMORY ADDRESS
0745 DA08               DW      HEXARI  ;H HEXADECIMAL SUM AND DIFFERENCE
0747 8409               DW      INFCB   ;I FILL INPUT FILE CONTROL BLOCK
0749 A50B               DW      CERROR  ;J
074B A50B               DW      CERROR  ;K
074D 9707               DW      LASSM   ;L LIST ASSEMBLY LANGUAGE
```



```
074F 5A09               DW      MOVE    ;M MOVE MEMORY
0751 A50B               DW      CERROR  ;N
0753 A50B               DW      CERROR  ;O
0755 A50B               DW      CERROR  ;P
0757 A50B               DW      CERROR  ;Q
0759 9609               DW      READ    ;R READ HEXADECIMAL FILE
075B 740A               DW      SETMEM  ;S SET MEMORY COMMAND
075D BD0A               DW      TRACE   ;T
075F B90A               DW      UNTRACE ;U
0761 A50B               DW      CERROR  ;V
0763 A50B               DW      CERROR  ;W
0765 E10A               DW      EXAMINE ;X EXAMINE AND MODIFY REGISTERS
0767 A50B               DW      CERROR  ;Y
0769 A50B               DW      CERROR  ;Z
                ;
                ;
                OPN:    ;FILE OPEN ROUTINE.  THIS SUBROUTINE OPENS THE DISK INPU
076B E5                 PUSH    H
076C D5                 PUSH    D
076D C5                 PUSH    B
076E AF                 XRA     A
076F 325B0B             STA     DBP     ;CLEAR BUFFER POINTER
0772 0E0F               MVI     C,OPF
0774 115C00             LXI     D,DFCB
0777 CDA206             CALL    TRAPAD  ;TO BDS
077A C1                 POP     B
077B D1                 POP     D
077C E1                 POP     H
077D C9                 RET
                ;
                ASSM:   ;ASSEMBLER LANGUAGE INPUT
                ;       CHECK FOR ASSM PRESENT
077E CD8D89             CALL    CHKDIS  ;ASSM/DISASSM PRESENT
0781 D2A50B             JNC     CERROR  ;NOT THERE
                ;
0784 CD8A8C             CALL    SCANEXP ;SCAN THE EXPRESSIONS WHICH FOLLOW
0787 3D                 DCR     A       ;ONE EXPRESSION EXPECTED
0788 C2A50B             JNZ     CERROR
078B CD608C             CALL    GETVAL  ;GET EXPRESSION TO H,L
078E 220C00             SHLD    DISPC
0791 CD0900             CALL    ASSEM
0794 C3FE06             JMP     START
                ;
                LASSM:  ;ASSEMBLER LANGUAGE OUTPUT LISTING
                ;       L<CR> LISTS FROM CURRENT DISASSM PC FOR SEVERAL LINES
                ;       L<NUMBER><CR> LISTS FROM <NUMBER> FOR SEVERAL LINES
                ;       L<NUMBER>,<NUMBER> LISTS BETWEEN LOCATIONS
0797 CD8D89             CALL    CHKDIS  ;DISASSM PRESENT?
079A D2A50B             JNC     CERROR
                ;
079D CD8A8C             CALL    SCANEXP ;SCAN EXPRESSIONS WHICH FOLLOW
07A0 CABB07             JZ      SPAGE   ;BRANCH IF NOT EXPRESSIONS
07A3 CD608C             CALL    GETVAL  ;EXP1 TO H,L
07A6 220C00             SHLD    DISPC   ;SETS BASE PC FOR LIST
07A9 3D                 DCR     A       ;ONLY EXPRESSION?
07AA CABB07             JZ      SPAGE   ;SETS SINGLE PAGE MODE
```

```
                ;
                ;       ANOTHER EXPRESSION FOLLOWS
07AD CD608C             CALL    GETVAL
07B0 220E08             SHLD    DISPM   ;SETS MAX VALUE
07B3 3D                 DCR     A
07B4 C2A508             JNZ     CERROR  ;ERROR IF MORE EXPN'S
07B7 AF                 XRA     A       ;CLEAR PAGE MODE
07B8 C3BD07             JMP     SPAG0
                ;
07BB 3E0C       SPAGE:  MVI     A,PSIZE ;SCREEN SIZE FOR LIST
07BD 321008     SPAG0:  STA     DISPG
07C0 CD0600             CALL    DISEN   ;CALL DISASSEMBLER
07C3 C3FE06             JMP     START   ;FOR ANOTHER COMMAND
                ;
                ;       DISPLAY MEMORY, FORMS ARE
                ;       D               DISPLAY FROM CURRENT DISPLAY LINE
                ;       DNNN            SET DISPLAY LINE AND ASSUME D
                ;       DNNN,MMM        DISPLAY NNN TO MMM
                ;       NEW DISPLAY LINE IS SET TO NEXT TO DISPLAY
                ;
                DISPLAY:
07C6 CD8A0C             CALL    SCANEXP ;GET 0,1,OR 2 EXPNS
07C9 CAE507             JZ      DISP1   ;ASSUME CURRENT DISLOC
07CC CD608C             CALL    GETVAL  ;GET VALUE TO H,L
07CF DAD507             JC      DISP0   ;CARRY SET IF ,B FORM
07D2 22570F             SHLD    DISLOC  ;OTHERWISE DISPC ALREADY SET
                DISP0:  ;GET NEXT VALUE
07D5 E67F               ANI     7FH     ;IN CASE ,B
07D7 3D                 DCR     A
07D8 CAE507             JZ      DISP1   ;SET HALF PAGE MODE
07DB CD608C             CALL    GETVAL
07DE 3D                 DCR     A       ;A,B,C NOT ALLOWED
07DF C2A508             JNZ     CERROR
07E2 C3F007             JMP     DISP2
                ;
                DISP1:  ;0 OR 1 EXPN, DISPLAY HALF SCREEN
07E5 2A570F             LHLD    DISLOC
07E8 7D                 MOV     A,L
07E9 E6F0               ANI     0F0H    ;NORMALIZE TO LINE START
07EB 6F                 MOV     L,A
07EC 110F00             LXI     D,PSIZE*16-1
07EF 19                 DAD     D
07F0 22590F     DISP2:  SHLD    DISMAX
                ;       DISPLAY MEMORY FROM DISLOC TO DISMAX
07F3 CD0D0C     DISP3:  CALL    CRLF
07F6 CD190C             CALL    BREAK   ;BREAK KEY?
07F9 C2FE06             JNZ     START   ;STOP CURRENT EXPANSION
07FC 2A570F             LHLD    DISLOC
07FF 225B0F             SHLD    TDISP
0802 CD280C             CALL    PADDR   ;PRINT LINE ADDRESS
0805 CD0F0B     DISP4:  CALL    BLANK
0808 7E                 MOV     A,M     ;GET NEXT DATA BYTE
0809 CDFF0B             CALL    PBYTE   ;PRINT BYTE
080C 23                 INX     H
080D CD3F0C             CALL    DISCOM  ;COMPARE H,L WITH DISMAX
0810 DA1908             JC      DISCH   ;CARRY SET IF H,L > DISMAX
```



```
0813 7D                 MOV     A,L     ;CHECK FOR LINE OVERFLOW
0814 E60F               ANI     0FH
0816 C20508             JNZ     DISP4   ;JUMP FOR ANOTHER BYTE
                ;
                DISCH:  ;DISPLAY AREA IN CHARACTER FORM
0819 22570F             SHLD    DISLOC  ;UPDATE FOR NEXT WRITE
081C 2A5B0F             LHLD    TDISP
081F EB                 XCHG
0820 CD0F0B             CALL    BLANK
                ;
0823 1A         DISCH0: LDAX    D       ;GET BYTE
0824 CD300C             CALL    PGRAPH  ;PRINT IF GRAPHIC CHARACTER
0827 13                 INX     D
0828 2A570F             LHLD    DISLOC  ;COMPARE FOR END OF LINE
082B 7D                 MOV     A,L
082C 93                 SUB     E
082D C22308             JNZ     DISCH0
0830 7C                 MOV     A,H
0831 92                 SUB     D
0832 C22308             JNZ     DISCH0
                ;
                ;       DROP THRU AT END OF CHARACTERS
0835 2A570F             LHLD    DISLOC
0838 CD3F0C             CALL    DISCOM  ;END OF DISPLAY?
083B DAFE06             JC      START
                ;
                ;       NO, CONTINUE WITH NEXT LINE
083E C3F307             JMP     DISP3
                ;
                ;
                ;       FILL MEMORY AREA WITH FIXED DATA ELEMENT
                ;
                SCAN3:  ;SCAN THREE EXPN'S FOR FILL AND MOVE
0841 CD8A0C             CALL    SCANEXP
0844 FE03               CPI     3
0846 C2A508             JNZ     CERROR
0849 CD608C             CALL    GETVAL
084C E5                 PUSH    H
084D CD608C             CALL    GETVAL
0850 E5                 PUSH    H
0851 CD608C             CALL    GETVAL
0854 D1                 POP     D
0855 C1                 POP     B       ;BC,DE,HL
0856 C9                 RET
                ;
                BCDE:   ;COMPARE BC > DE (CARRY GEN'D IF TRUE)
0857 7B                 MOV     A,E
0858 91                 SUB     C
0859 7A                 MOV     A,D
085A 98                 SBB     B
085B C9                 RET
                ;
                FILL:
085C CD4108             CALL    SCAN3   ;EXPRESSIONS SCANNED BC , DE , HL
085F 7C                 MOV     A,H     ;MUST BE ZERO
0860 B7                 ORA     A
```

```
0861 C2A50B             JNZ     CERROR
0864 CD5708     FILL0,  CALL    BCDE    ;END OF FILL?
0867 DAFE06             JC      START
086A 7D                 MOV     A,L     ;DATA
086B 02                 STAX    B       ;TO MEMORY
086C 03                 INX     B       ;NEXT TO FILL
086D C36408             JMP     FILL0
                ;
                ;       GO COMMAND WITH OPTIONAL BREAKPOINTS
                ;
                GOTO,
0870 CD0F0C             CALL    CRLF    ;READY FOR GO.
0873 CD8A0C             CALL    SCANEXP ;0,1, OR 2 EXPS
0876 CD600C             CALL    GETVAL
0879 E5                 PUSH    H       ;START ADDRESS
087A CD600C             CALL    GETVAL
087D E5                 PUSH    H       ;BKPT1
087E CD600C             CALL    GETVAL
0881 44                 MOV     B,H     ;BKPT2
0882 4D                 MOV     C,L
0883 D1                 POP     D       ;BKPT1
0884 E1                 POP     H       ;GOTO ADDRESS
                ;
                GOPR,
0885 F3                 DI
0886 CAA108             JZ      GOP1    ;NO BREAK POINTS
0889 DA8F08             JC      GOP0
                ;       SET PC
088C 22B30F             SHLD    PLOC    ;INTO MACHINE STATE
                GOP0,   ;SET BREAKS
088F E67F               ANI     7FH     ;CLEAR 7 BIT
0891 3D                 DCR     A       ;IF 1 THEN SKIP (2,3 IF BREAKPOINTS)
0892 CAA108             JZ      GOP1
0895 CDB208             CALL    SETBK   ;BREAK POINT FROM D,E
0898 3D                 DCR     A
0899 CAA108             JZ      GOP1
                ;       SECOND BREAK POINT
089C 59                 MOV     E,C
089D 50                 MOV     D,B     ;TO D,E
089E CDB208             CALL    SETBK   ;SECOND BREAK POINT SET
                ;
                GOP1,   ;RESTORE MACHINE STATE AND START IT
08A1 31A90F             LXI     SP,STACK-12
08A4 D1                 POP     D
08A5 C1                 POP     B
08A6 F1                 POP     PSW
08A7 E1                 POP     H       ;SP IN HL
08A8 F9                 SPHL
08A9 2AB30F             LHLD    PLOC    ;PC IN HL
08AC E5                 PUSH    H       ;INTO USER'S STACK
08AD 2AB10F             LHLD    HLOC    ;HL RESTORED
08B0 FB                 EI
08B1 C9                 RET
                ;
                SETBK,  ;SET BREAK POINT AT LOCATION D,E
08B2 F5                 PUSH    PSW
```



```
08B3 C5                 PUSH    B
08B4 21490F             LXI     H,BREAKS        ;NUMBER OF BREAKS SET SO FAR
08B7 7E                 MOV     A,M
08B8 34                 INR     M       ;COUNT BREAKS UP
08B9 B7                 ORA     A       ;ONE SET ALREADY?
08BA CACD08             JZ      SETBK0
                ;       ALREADY SET, MOVE PAST ADDR,DATA FIELDS
08BD 23                 INX     H
08BE 7E                 MOV     A,M     ;CHECK = ADDRESSES
08BF 23                 INX     H
08C0 46                 MOV     B,M     ;CHECK HO ADDRESS
08C1 23                 INX     H
                ;       DON'T SET TWO BREAKPOINTS IF EQUAL
08C2 BB                 CMP     E       ;LOW =?
08C3 C2CD08             JNZ     SETBK0
08C6 78                 MOV     A,B
08C7 BA                 CMP     D       ;HIGH =?
08C8 C2CD08             JNZ     SETBK0
                ;       EQUAL ADDRESSES, REPLACE REAL DATA
08CB 7E                 MOV     A,M     ;GET DATA BYTE
08CC 12                 STAX    D       ;PUT BACK INTO CODE
08CD 23         SETBK0, INX     H       ;ADDRESS FIELD
08CE 73                 MOV     M,E     ;LSB
08CF 23                 INX     H
08D0 72                 MOV     M,D     ;MSB
08D1 23                 INX     H       ;DATA FIELD
08D2 1A                 LDAX    D       ;GET BYTE FROM PROGRAM
08D3 77                 MOV     M,A     ;TO BREAKS VECTOR
08D4 3EFF               MVI     A,RSTIN ;RESTART INSTRUCTION
08D6 12                 STAX    D       ;TO CODE
08D7 C1                 POP     B
08D8 F1                 POP     PSW
08D9 C9                 RET
                ;
                ;
                ;       HEXADECIMAL ARITHMETIC
                ;
                HEXAR1,
08DA CD8A0C             CALL    SCANEXP
08DD FE02               CPI     2
08DF C2A50B             JNZ     CERROR
08E2 CD600C             CALL    GETVAL  ;FIRST VALUE TO H,L
08E5 E5                 PUSH    H
08E6 CD600C             CALL    GETVAL  ;SECOND VALUE TO H,L
08E9 D1                 POP     D       ;FIRST VALUE TO D,E
08EA E5                 PUSH    H       ;SAVE A COPY OF SECOND VAALUE
08EB CD0F0C             CALL    CRLF    ;NEW LINE
08EE 19                 DAD     D       ;SUM IN H,L
08EF CD280C             CALL    PADDR
08F2 CD0F0B             CALL    BLANK
08F5 E1                 POP     H       ;RESTORE SECOND VALUE
08F6 AF                 XRA     A       ;CLEAR ACCUM FOR SUBTRACTION
08F7 95                 SUB     L
08F8 6F                 MOV     L,A     ;BACK TO L
08F9 3E00               MVI     A,0     ;CLEAR IT AGAIN
08FB 9C                 SBB     H
```

```
08FC 67                 MOV     H,A
08FD 19                 DAD     D       ;DIFFERENCE IN HL
08FE CD208C             CALL    PADDR
0901 C3FE06             JMP     START
                        ;
                        ; SET INPUT FILE CONTROL BLOCK (AT 5CH) TO SIMULATE CONSOLE COMMAND
                        INFCB,
                        ;       FILL FCB AT 5CH
0904 AF                 XRA     A
0905 327C00             STA     FCB+FCR ;CLEAR CURRENT RECORD
0908 325C00             STA     FCB     ;CLEAR DISK NUMBER
090B CDD70B             CALL    GNC     ;CHARACTER IN A
090E 0E09               MVI     C,9     ;FILE NAME LENGTH+1
0910 215D00             LXI     H,FCB+FFN       ;START OF NAME
                        ;
                        FLP,    ;FILL NAME
0913 77                 MOV     M,A
0914 23                 INX     H
0915 0D                 DCR     C
0916 CAA50B             JZ      CERROR  ;FILE NAME TOO LONG.
                        ;
0919 CDD70B             CALL    GNC     ;READ NEXT CHAR
091C FE2E               CPI     '.'
091E CA2609             JZ      FLB     ;FOUND .. BLANK OUT
                        ;       NOT ., MAY BE CR
0921 FE0D               CPI     CR
0923 C21309             JNZ     FLP     ;FOR ANOTHER STORE
                        ;
                        ;       NAME FILLED. EXTEND WITH BLANKS
0926 0D                 FLB,    DCR     C
0927 CA3009             JZ      TFT
092A 3620               MVI     M,' '
092C 23                 INX     H
092D C32609             JMP     FLB
                        ;
                        ;       BLANKS FILLED. SCAN FILE TYPE IF '.' FOUND
0930 0E04               TFT,    MVI     C,4
0932 FE2E               CPI     '.'     ;ENDED WITH . OR CR
0934 C24B09             JNZ     FLB1    ;FILL REMAINDER WITH BLANKS
                        ;
                        ;       SCAN FILE TYPE
0937 216500             LXI     H,FCB+FFT
                        ;
093A CDD70B             FLP1,   CALL    GNC
093D FE0D               CPI     CR
093F CA4B09             JZ      FLB1
0942 77                 MOV     M,A
0943 23                 INX     H
0944 0D                 DCR     C
0945 CAA50B             JZ      CERROR  ;TOO LONG
0948 C33A09             JMP     FLP1
                        ;
                        ;       FILL WITH BLANKS
094B 0D                 FLB1,   DCR     C
094C CA5509             JZ      FLZ
094F 3620               MVI     M,' '
```



```
0951 23                 INX     H
0952 C34B09             JMP     FLB1
                        ;
                        ;       ZERO THE EXTENT
0955 3600               FLZ,    MVI     M,0
0957 C3FE06             JMP     START
                        ;
                        ;       MOVE MEMORY
                        MOVE,
095A CD4100             CALL    SCAN3   ;BC,DE,HL
                        MOVE0,  ;HAS B,C PASSED D,E?
095D CD5700             CALL    BCDE
0960 DAFE06             JC      START   ;END OF MOVE
0963 0A                 LDAX    B       ;CHAR TO ACCUM
0964 03                 INX     B       ;NEXT TO GET
0965 77                 MOV     M,A     ;MOVE IT TO MEMORY
0966 23                 INX     H
0967 C35D09             JMP     MOVE0   ;FOR ANOTHER
                        ;
                        ;       READ FILES (HEX OR COM)
                        ;
                        QHEX,   ;HEX FILE IF ZERO AT END
096A 216500             LXI     H,FCB+FFT
096D 7E                 MOV     A,M
096E FE48               CPI     'H'
0970 C0                 RNZ
0971 23                 INX     H
0972 7E                 MOV     A,M
0973 FE45               CPI     'E'
0975 C0                 RNZ
0976 23                 INX     H
0977 7E                 MOV     A,M
0978 FE50               CPI     'X'
097A C9                 RET
                        ;
                        COMLOAD,        ;COMPARE HL > MLOAD
097B EB                 XCHG    ;H,L TO D,E
097C 2A818F             LHLD    MLOAD   ;MLOAD TO H,L
097F 7D                 MOV     A,L     ;MLOAD LSB
0980 93                 SUB     E
0981 7C                 MOV     A,H
0982 9A                 SBB     D       ;MLOAD-OLDHL GENS CARRY IF HL>MLOAD
0983 EB                 XCHG
0984 C9                 RET
                        ;
                        CKMLOAD,        ;CHECK FOR HL > MLOAD AND SET MLOAD IF SO
0985 CD7B09             CALL    COMLOAD ;CARRY IF HL>MLOAD
0988 D0                 RNC
0989 22818F             SHLD    MLOAD   ;CHANGE IT
098C C9                 RET
                        ;
                        CHKDIS, ;CHECK FOR DISASSM PRESENT
098D E5                 PUSH    H
098E 218080             LXI     H,MODBAS        ;ENTRY POINT
0991 CD7B09             CALL    COMLOAD
0994 E1                 POP     H
```

```
0995 C9                 RET
                ;
                READ.
0996 CD8A8C             CALL    SCANEXP
0999 210000             LXI     H,0
099C CAA609             JZ      READH
099F 3D                 DCR     A       ;ONE EXPRESSION?
09A0 C2A588             JNZ     CERROR
09A3 CD608C             CALL    GETVAL  ;EXPRESSION TO H,L
09A6 E5         READH.  PUSH    H       ;SAVE IT FOR BELOW
09A7 CD6B87     RINIT.  CALL    OPN     ;OPEN INPUT FILE
09AA FEFF               CPI     255
09AC CAA588             JZ      CERROR
                ;       CONTINUE IF FILE OPEN WENT OK
                ;       DISK FILE OPENED AND INITIALIZED
                ;
                ;       CHECK FOR 'HEX' FILE AND LOAD DIRECT TIL EOF
09AF CD6A89             CALL    QHEX    ;LOOK FOR 'HEX'
09B2 CADB09             JZ      HREAD
                ;
                ;       COM FILE, LOAD WITH OFFSET GIVEN BY PUSHED REGISTER H
09B5 E1                 POP     H
09B6 110001             LXI     D,100H  ;BASE OF TRANSIENT AREA
09B9 19                 DAD     D
                ;       REG H HOLDS LOAD ADDRESS
                LCOM0.  ;LOAD COM FILE
09BA E5                 PUSH    H       ;SAVE DMA ADDRESS
09BB 115C00             LXI     D,5CH
09BE 0E14               MVI     C,RDF   ;READ SECTOR
09C0 CDA286             CALL    TRAPAD
09C3 E1                 POP     H
09C4 B7                 ORA     A       ;SET FLAGS TO CHECK RETURN CODE
09C5 C2408A             JNZ     RLIFT
                ;       MOVE FROM 80H TO LOAD ADDRESS IN H,L
09C8 118000             LXI     D,80H
09CB 0E80               MVI     C,80H   ;BUFFER SIZE
09CD 1A         LCOM1.  LDAX    D       ;LOAD NEXT BYTE
09CE 13                 INX     D
09CF 77                 MOV     M,A     ;STORE NEXT BYTE
09D0 23                 INX     H
09D1 0D                 DCR     C
09D2 C2CD09             JNZ     LCOM1
                ;       LOADED, CHECK ADDRESS AGAINST MLOAD
09D5 CD8589             CALL    CKMLOAD
09D8 C3BA89             JMP     LCOM0
                ;
                ;
                ;       OTHERWISE ASSUME HEX FILE IS BEING LOADED
09DB CD6E8B     HREAD.  CALL    DISKR   ;NEXT CHAR TO ACCUM
09DE FE1A               CPI     EOF     ;PAST END OF TAPE?
09E0 CAA588             JZ      CERROR  ;FOR ANOTHER COMMAND
09E3 DE3A               SBI     ':'
09E5 C2DB09             JNZ     HREAD   ;LOOKING FOR START OF RECORD
                ;
                ;       START FOUND, CLEAR CHECKSUM
09E8 57                 MOV     D,A
```



```
09E9 E1                 POP     H
09EA E5                 PUSH    H
09EB CD208A             CALL    RBYTE
09EE 5F                 MOV     E,A     ;SAVE LENGTH
09EF CD208A             CALL    RBYTE   ;HIGH ORDER ADDR
09F2 F5                 PUSH    PSW
09F3 CD208A             CALL    RBYTE   ;LOW ORDER ADDR
09F6 C1                 POP     B
09F7 4F                 MOV     C,A
09F8 09                 DAD     B       ;BIASED ADDR IN H
09F9 7B                 MOV     A,E     ;CHECK FOR LAST RECORD
09FA B7                 ORA     A
09FB C2068A             JNZ     RDTYPE
                ;       END OF TAPE, SET LOAD ADDRESS
09FE 60                 MOV     H,B
09FF 69                 MOV     L,C
0A00 228B8F             SHLD    PLOC    ;SET PC VALUE
0A03 C3408A             JMP     RLIFT   ;FOR ANOTHER COMMAND
                ;
                RDTYPE.
0A06 CD208A             CALL    RBYTE   ;RECORD TYPE = 0
                ;
                ;       LOAD RECORD
0A09 CD208A     RED1.   CALL    RBYTE
0A0C 77                 MOV     M,A
0A0D 23                 INX     H
0A0E 1D                 DCR     E
0A0F C2090A             JNZ     RED1    ;FOR ANOTHER BYTE
                ;       OTHERWISE AT END OF RECORD - CHECKSUM
0A12 CD208A             CALL    RBYTE
0A15 F5                 PUSH    PSW     ;FOR CHECKSUM CHECK
0A16 CD8589             CALL    CKMLOAD ;CHECK AGAINST MLOAD
0A19 F1                 POP     PSW
0A1A C2A588             JNZ     CERROR  ;CHECKSUM ERROR
0A1D C3DB09             JMP     HREAD   ;FOR ANOTHER RECORD
                ;
                RBYTE.  ;READ ONE BYTE FROM BUFF AT WBP TO REG-A
                ;       COMPUTE CHECKSUM IN REG-D
0A20 C5                 PUSH    B
0A21 E5                 PUSH    H
0A22 D5                 PUSH    D
                ;
0A23 CD6E8B             CALL    DISKR   ;GET ONE MORE CHARACTER
0A26 CD538C             CALL    HEXCON  ;CONVERT TO HEX (OR ERROR)
                ;
                ;       SHIFT LEFT AND MASK
0A29 07                 RLC
0A2A 07                 RLC
0A2B 07                 RLC
0A2C 07                 RLC
0A2D E6F0               ANI     0F0H
0A2F F5                 PUSH    PSW     ;SAVE FOR A FEW STEPS
0A30 CD6E8B             CALL    DISKR
0A33 CD538C             CALL    HEXCON
                ;
                ;       OTHERWISE SECOND NIBBLE OK, SO MERGE
```

```
CP/M MACRO ASSEM 2.0     0814    CP/M DEBUGGER (DEMON) 1/78

0A36 C1                  POP     B       ;PREVIOUS NIBBLE TO REG-B
0A37 B0                  ORA     B
0A38 47                  MOV     B,A     ;VALUE IS NOW IN B TEMPORARILY
0A39 D1                  POP     D       ;CHECKSUM
0A3A 82                  ADD     D       ;ACCUMULATING
0A3B 57                  MOV     D,A     ;BACK TO CS
                 ;       ZERO FLAG REMAINS SET
0A3C 78                  MOV     A,B     ;BRING BYTE BACK TO ACCUMULATOR
0A3D E1                  POP     H
0A3E C1                  POP     B       ;BACK TO INITIAL STATE WITH ACCUM SET
0A3F C9                  RET
                 RLIFT.  ;LIFT HEAD ON DISK BEFORE RETURNING
0A40 0E0C                MVI     C,LIFT
0A42 CDA206              CALL    TRAPAD
                 ;       'NEXT' ' PC'
0A45 21690A              LXI     H,LMSG  ;LOAD MESSAGE
0A48 7E          RLI0.   MOV     A,M
0A49 B7                  ORA     A       ;LAST CHAR?
0A4A CA540A              JZ      RLI1
0A4D CDC100              CALL    PCHAR
0A50 23                  INX     H       ;NEXT CHAR
0A51 C3480A              JMP     RLI0
0A54 CD0F0C      RLI1.   CALL    CRLF
0A57 2A810F              LHLD    MLOAD
0A5A CD280C              CALL    PADDR
0A5D CD0F0B              CALL    BLANK
0A60 2A830F              LHLD    PLOC
0A63 CD280C              CALL    PADDR
0A66 C3FE06              JMP     START
0A69 0D0A4E4558LMSG.     DB      CR,LF,'NEXT  PC',0
                 ;
                 ;       SET MEMORY COMMAND
                 ;
                 SETMEM. ;ONE EXPRESSION EXPECTED
0A74 CD8A0C              CALL    SCANEXP ;SETS FLAGS
0A77 3D                  DCR     A       ;ONE EXPRESSION ONLY
0A78 C2A50B              JNZ     CERROR
0A7B CD600C              CALL    GETVAL  ;START ADDRESS IS IN H,L
0A7E CD0F0C      SETM0.  CALL    CRLF    ;NEW LINE
0A81 E5                  PUSH    H       ;SAVE CURRENT ADDRESS
0A82 CD280C              CALL    PADDR   ;PRINTED
0A85 CD0F0B              CALL    BLANK   ;SEPARATOR
0A88 E1                  POP     H       ;GET DATA
0A89 7E                  MOV     A,M
0A8A E5                  PUSH    H       ;SAVE ADDRESS TO FILL
0A8B CDFF0B              CALL    PBYTE   ;PRINT BYTE
0A8E CD0F0B              CALL    BLANK   ;ANOTHER SEPARATOR
0A91 CD000B              CALL    GETBUFF ;FILL INPUT BUFFER
0A94 CDD70B              CALL    GHC     ;MAY BE EMPTY (NO CHANGE)
0A97 E1                  POP     H       ;RESTORE ADDRESS TO FILL
0A98 FE0D                CPI     CR
0A9A CAB50A              JZ      SETM1
0A9D FE2E                CPI     '.'
0A9F CAFE06              JZ      START
                 ;       DATA IS BEING CHANGED
0AA2 E5                  PUSH    H       ;SAVE ADDR TO FILL
```



```
CP/M MACRO ASSEM 2.0     0815    CP/M DEBUGGER (DEMON) 1/78

0AA3 CD8D0C              CALL    SCANEX  ;FIRST CHARACTER ALREADY SCANNED
0AA6 3D                  DCR     A       ;ONE ITEM?
0AA7 C2A50B              JNZ     CERROR  ;MORE THAN ONE
0AAA CD600C              CALL    GETVAL  ;VALUE TO H,L
0AAD 7C                  MOV     A,H
0AAE B7                  ORA     A       ;NO ZERO?
0AAF C2A50B              JNZ     CERROR  ;DATA IS IN L
0AB2 7D                  MOV     A,L
0AB3 E1                  POP     H       ;RESTORE DATA VALUE
0AB4 77                  MOV     M,A
0AB5 23          SETM1.  INX     H       ;NEXT ADDRESS READY
0AB6 C37E0A              JMP     SETM0
                 ;
                 ;       UNTRACE MODE
                 UNTRACE.
0AB9 AF                  XRA     A       ;CLEAR TRACE MODE FLAG
0ABA C3BF0A              JMP     ETRACE
                 ;
                 ;       START TRACE
0ABD 3EFF        TRACE.  MVI     A,0FFH  ;SET TRACE MODE FLAG
                 ETRACE.
0ABF 32460F              STA     TMODE
0AC2 CD8A0C              CALL    SCANEXP
0AC5 210000              LXI     H,0
0AC8 CAD00A              JZ      TRAC0
                 ;       MUST BE T OR TN (H NOT 0)
0ACB 3D                  DCR     A       ;COUNT MUST BE ONE
0ACC C2A50B              JNZ     CERROR
0ACF CD600C              CALL    GETVAL  ;GET VALUE TO HL
0AD2 7D                  MOV     A,L     ;CHECK FOR ZERO
0AD3 B4                  ORA     H
0AD4 CAA50B              JZ      CERROR
0AD7 2B                  DCX     H       ;TRACE VALUE - 1
0AD8 22470F      TRAC0.  SHLD    TRACER
0ADB CD3E0D              CALL    DSTATE  ;STARTING STATE IS DISPLAYED
0ADE C30500              JMP     GOPR    ;SETS BREAKPOINTS AND STARTS EXECUTION
                 ;
                 ; EXAMINE AND MODIFY CPU REGISTERS.
                 EXAMINE.
0AE1 CDD70B              CALL    GHC     ;CR?
0AE4 FE0D                CPI     CR
0AE6 C2EF0A              JNZ     EXAM0
0AE9 CD3E0D              CALL    DSTATE  ;DISPLAY CPU STATE
0AEC C3FE06              JMP     START
                 ;
                 EXAM0.  ;REGISTER CHANGE OPERATION
0AEF 010B00              LXI     B,PVAL+1        ;B=0,C=PVAL (MAX REGISTER NUMBER)
                 ;       LOOK FOR REGISTER MATCH IN RVECT
0AF2 21AD0D              LXI     H,RVECT
0AF5 BE          EXAM1.  CMP     M       ;MATCH IN RVECT?
0AF6 CA020B              JZ      EXAM2
0AF9 23                  INX     H       ;NEXT RVECT
0AFA 04                  INR     B       ;INCREMENT COUNT
0AFB 0D                  DCR     C       ;END OF RVECT?
0AFC C2F50A              JNZ     EXAM1
                 ;       NO MATCH
```

```
0AFF C3A58B             JMP     CERROR

                ;
                EXAM2,  ;MATCH IN RVECT, B HAS REGISTER NUMBER
0802 CDD70B             CALL    GNC
0805 FE0D               CPI     CR      ;ONLY CHARACTER?
0807 C2A58B             JNZ     CERROR
                ;
                ;       WRITE CONTENTS, AND GET ANOTHER BUFFER
080A C5                 PUSH    B       ;SAVE COUNT
080B CD0F0C             CALL    CRLF    ;NEW LINE FOR ELEMENT
080E CD148D             CALL    DELT    ;ELEMENT WRITTEN
0811 CD0F0B             CALL    BLANK
0814 CD800B             CALL    GETBUFF ;FILL COMMAND BUFFER
0817 CD8A0C             CALL    SCANEXP ;GET INPUT EXPRESSION
081A B7                 ORA     A       ;NONE?
081B CAFE06             JZ      START
081E 3D                 DCR     A       ;MUST BE ONLY ONE
081F C2A58B             JNZ     CERROR
0822 CD600C             CALL    GETVAL  ;VALUE IS IN H,L
0825 C1                 POP     B       ;RECALL REGISTER NUMBER
                ;       CHECK CASES FOR FLAGS, REG-A, OR DOUBLE REGISTER
0826 78                 MOV     A,B
0827 FE05               CPI     AVAL
0829 D2538B             JNC     EXAM4
                ;       SETTING FLAGS, MUST BE ZERO OR ONE
082C 7C                 MOV     A,H
082D B7                 ORA     A
082E C2A58B             JNZ     CERROR
0831 7D                 MOV     A,L
0832 FE02               CPI     2
0834 D2A58B             JNC     CERROR
                ;       0 OR 1 IN H,L REGISTERS - GET CURRENT FLAGS AND MASK POSITION
0837 CDDD0C             CALL    FLGSHF
                ;       SHIFT COUNT IN C, D,E ADDRESS FLAG POSITION
083A 67                 MOV     H,A     ;FLAGS TO H
083B 41                 MOV     B,C     ;SHIFT COUNT TO B
083C 3EFE               MVI     A,0FEH  ;11111110 IN ACCUM TO ROTATE
083E CD4D0B             CALL    LROTATE ;ROTATE REG-A LEFT
0841 A4                 ANA     H       ;MASK ALL BUT ALTERED BIT
0842 41                 MOV     B,C     ;RESTORE SHIFT COUNT TO B
0843 67                 MOV     H,A     ;SAVE MASKED FLAGS
0844 7D                 MOV     A,L     ;0/1 TO LSB OF ACCUM
0845 CD4D0B             CALL    LROTATE ;ROTATED TO CHANGED POSITION
0848 B4                 ORA     H       ;RESTORE ALL OTHER FLAGS
0849 12                 STAX    D       ;BACK TO MACHINE STATE
084A C3FE06             JMP     START   ;FOR ANOTHER COMMAND
                ;
                LROTATE,        ;LEFT ROTATE FOR FLAG SETTING
                ;       PATTERN IS IN REGISTER A, COUNT IN REGISTER B
084D 05                 DCR     B
084E C8                 RZ      ;ROTATE COMPLETE
084F 07                 RLC     ;END-AROUND ROTATE
0850 C34D0B             JMP     LROTATE
                ;
                EXAM4,  ;MAY BE ACCUMULATOR CHANGE
0853 C2638B             JNZ     EXAM5
```



```
                ;       MUST BE BYTE VALUE
0856 7C                 MOV     A,H
0857 B7                 ORA     A
0858 C2A58B             JNZ     CERROR
085B 7D                 MOV     A,L     ;GET BYTE TO STORE
085C 21AE0F             LXI     H,ALOC  ;A REG LOCATION IN MACHINE STATE
085F 77                 MOV     M,A     ;STORE IT AWAY
0860 C3FE06             JMP     START
                ;
                EXAM5,  ;MUST BE DOUBLE REGISTER PAIR
0863 E5                 PUSH    H       ;SAVE VALUE
0864 CDFB0C             CALL    GETDBA  ;DOUBLE ADDRESS TO HL
0867 D1                 POP     D       ;VALUE TO D,E
0868 73                 MOV     M,E
0869 23                 INX     H
086A 72                 MOV     M,D     ;ALTERED MACHINE STATE
086B C3FE06             JMP     START
                ;
                DISKR,  ;DISK READ
086E E5                 PUSH    H
086F D5                 PUSH    D
0870 C5                 PUSH    B
                ;
                RDI,    ;READ DISK INPUT
0871 3A5B0B             LDA     DBP
0874 E67F               ANI     7FH
0876 CA8E0B             JZ      NDI     ;GET NEXT DISK INPUT RECORD
                ;
                ;       READ CHARACTER
                RDC,
0879 1600               MVI     D,0
087B 5F                 MOV     E,A
087C 21B000             LXI     H,DBF
087F 19                 DAD     D
0880 7E                 MOV     A,M
0881 FE1A               CPI     DEOF
0883 CAA00B             JZ      DEF     ;END OF FILE
0886 215B0B             LXI     H,DBP
0889 34                 INR     M
088A B7                 ORA     A
088B C3A10B             JMP     RRET
                ;
                NDI,    ;NEXT BUFFER IN
088E 0E14               MVI     C,RDF
0890 115C00             LXI     D,DFCB
0893 CDA206             CALL    TRAPAD
0896 B7                 ORA     A
0897 C2A00B             JNZ     DEF
                ;
                ;       BUFFER READ OK
089A 325B0B             STA     DBP     ;STORE 00H
089D C3790B             JMP     RDC
                ;
                DEF,    ;SET CARRY AND RETURN (END FILE)
08A0 37                 STC
                RRET,
```

```
0BA1 C1                 POP     B
0BA2 D1                 POP     D
0BA3 E1                 POP     H
0BA4 C9                 RET
                ;
                CERROR. ;ERROR IN COMMAND
0BA5 CD0F0C             CALL    CRLF
0BA8 3E3F               MVI     A,'?'
0BAA CDC10B             CALL    PCHAR
0BAD C3FE06             JMP     START
                ;
                ; SUBROUTINES
                GETBUFF.        ;FILL COMMAND BUFFER AND SET POINTERS
0BB0 0E0A               MVI     C,GETF  ;GET BUFFER FUNCTION
0BB2 115F0F             LXI     D,COMLEN;START OF COMMAND BUFFER
0BB5 CDA206             CALL    TRAPAD  ;FILL BUFFER
0BB8 21610F             LXI     H,COMBUF;NEXT TO GET
0BBB 225D0F             SHLD    NEXTCOM
0BBE C9                 RET
                ;
                BLANK.
0BBF 3E20               MVI     A,' '
                ;
                PCHAR.  ;PRINT CHARACTER TO CONSOLE
0BC1 E5                 PUSH    H
0BC2 D5                 PUSH    D
0BC3 C5                 PUSH    B
0BC4 5F                 MOV     E,A
0BC5 0E02               MVI     C,CDF
0BC7 CDA206             CALL    TRAPAD
0BCA C1                 POP     B
0BCB D1                 POP     D
0BCC E1                 POP     H
0BCD C9                 RET
                ;
                TRANS.
                ;       TRANSLATE TO UPPER CASE
0BCE FE7F               CPI     7FH     ;RUBOUT?
0BD0 C8                 RZ
0BD1 FE61               CPI     ('A' OR 01000000B)      ;UPPER CASE A
0BD3 D8                 RC
0BD4 E65F               ANI     10111111B       ;CLEAR UPPER CASE BIT
0BD6 C9                 RET
                ;
                GNC.
                ;       GET NEXT BUFFER CHARACTER FROM CONSOLE
0BD7 E5                 PUSH    H       ;SAVE FOR REUSE LOCALLY
0BD8 21600F             LXI     H,CURLEN
0BDB 7E                 MOV     A,M
0BDC B7                 ORA     A       ;ZERO?
0BDD 3E0D               MVI     A,CR
0BDF CAEE0B             JZ      GNCRET  ;RETURN WITH CR IF EXHAUSTED
0BE2 35                 DCR     M       ;CURLEN=CURLEN-1
0BE3 2A5D0F             LHLD    NEXTCOM
0BE6 7E                 MOV     A,M     ;GET NEXT CHARACTER
0BE7 23                 INX     H       ;NEXTCOM=NEXTCOM+1
```



```
0BE8 225D0F             SHLD    NEXTCOM ;UPDATED
0BEB CDCE0B             CALL    TRANS
0BEE E1         GNCRET. POP     H       ;RESTORE ENVIRONMENT
0BEF C9                 RET
                ;
                PNIB.   ;PRINT NIBBLE IN LO ACCUM
0BF0 FE0A               CPI     10
0BF2 D2FA0B             JNC     PNIBH   ;JUMP IF A-F
0BF5 C630               ADI     '0'
0BF7 C3C10B             JMP     PCHAR   ;RET THRU PCHAR
0BFA C637       PNIBH.  ADI     'A'-10
0BFC C3C10B             JMP     PCHAR
                ;
0BFF F5         PBYTE.  PUSH    PSW     ;SAVE A COPY FOR LO NIBBLE
0C00 1F                 RAR
0C01 1F                 RAR
0C02 1F                 RAR
0C03 1F                 RAR
0C04 E60F               ANI     0FH     ;MASK HO NIBBLE TO LO NIBBLE
0C06 CDF80B             CALL    PNIB
0C09 F1                 POP     PSW     ;RECALL BYTE
0C0A E60F               ANI     0FH
0C0C C3F80B             JMP     PNIB
                ;
                CRLF.   ;CARRIAGE RETURN LINE FEED
0C0F 3E0D               MVI     A,CR
0C11 CDC10B             CALL    PCHAR
0C14 3E0A               MVI     A,LF
0C16 C3C10B             JMP     PCHAR
                ;
                BREAK.  ;CHECK FOR BREAK KEY
0C19 C5                 PUSH    B
0C1A D5                 PUSH    D
0C1B E5                 PUSH    H
0C1C 0E0B               MVI     C,CHKIO
0C1E CDA206             CALL    TRAPAD
0C21 E601               ANI     1B
0C23 E1                 POP     H
0C24 D1                 POP     D
0C25 C1                 POP     B
0C26 C9                 RET
                ;
                PADDX.  ;SAME AS PADDR. EXCEPT PRINT VALUE IN D,E
0C27 EB                 XCHG
                ;
                PADDR.  ;PRINT THE ADDRESS VALUE IN H,L
0C28 7C                 MOV     A,H
0C29 CDFF0B             CALL    PBYTE
0C2C 7D                 MOV     A,L
0C2D C3FF0B             JMP     PBYTE
                ;
                PGRAPH. ;PRINT GRAPHIC CHARACTER IN REG-A OR '.' IF NOT
0C30 FE7F               CPI     7FH
0C32 D23A0C             JNC     PPERIOD
0C35 FE20               CPI     ' '
0C37 D2C10B             JNC     PCHAR
```

```
                PPERIOD,
0C3A 3E2E               MVI     A,'.'
0C3C C3C18B             JMP     PCHAR
                ;
                DISCOM, ;COMPARE H,L AGAINST DISMAX.  CARRY SET IF HL > DISMAX AND
0C3F EB                 XCHG
0C40 2A598F             LHLD    DISMAX
0C43 7D                 MOV     A,L
0C44 93                 SUB     E
0C45 6F                 MOV     L,A     ;REPLACE FOR ZERO TESTS LATER
0C46 7C                 MOV     A,H
0C47 9A                 SBB     D
0C48 EB                 XCHG
0C49 C9                 RET
                ;
                DELIM,  ;CHECK FOR DELIMITER CHARACTER
0C4A FE0D               CPI     CR
0C4C C8                 RZ
0C4D FE2C               CPI     ','
0C4F C8                 RZ
0C50 FE20               CPI     ' '
0C52 C9                 RET
                ;
                HEXCON, ;CONVERT ACCUMULATOR TO PURE BINARY FROM EXTERNAL HEX
0C53 D630               SUI     '0'
0C55 FE0A               CPI     10
0C57 D8                 RC              ;MUST BE 0-9
0C58 C6F9               ADI     ('0'-'A'+10) AND 0FFH
0C5A FE10               CPI     16
0C5C D8                 RC              ;MUST BE 0-15
0C5D C3A58B             JMP     CERROR  ;BAD HEX DIGIT
                ;
                GETVAL, ;GET NEXT EXPRESSION VALUE TO H,L (POINTER IN D,E ASSUMED)
0C60 EB                 XCHG
0C61 5E                 MOV     E,M
0C62 23                 INX     H
0C63 56                 MOV     D,M
0C64 23                 INX     H
0C65 EB                 XCHG
0C66 C9                 RET
                ;
                GETEXP, ;GET HEX VALUE TO D,E
0C67 EB                 XCHG
0C68 210000             LXI     H,0
                GETEXP0,
0C6B CD538C             CALL    HEXCON
0C6E 29                 DAD     H       ;*2
0C6F 29                 DAD     H       ;*4
0C70 29                 DAD     H       ;*8
0C71 29                 DAD     H       ;*16
0C72 B5                 ORA     L       ;HL=HL+HEX
0C73 6F                 MOV     L,A
0C74 CDD78B             CALL    GNC
0C77 CD4A8C             CALL    DELIM   ;DELIMITER?
0C7A C26B8C             JNZ     GETEXP0
0C7D EB                 XCHG
```



```
0C7E C9                 RET
                ;
                SCSTORE,        ;STORE D,E TO H,L AND INCREMENT ADDRESS
0C7F 73                 MOV     M,E
0C80 23                 INX     H
0C81 72                 MOV     M,D
0C82 23                 INX     H
0C83 E5                 PUSH    H
0C84 21508F             LXI     H,EXPLIST
0C87 34                 INR     M       ;COUNT NUMBER OF EXPN'S
0C88 E1                 POP     H
0C89 C9                 RET
                ;
                SCANEXP,        ;SCAN EXPRESSIONS - CARRY SET IF ,B
                ;       ZERO SET IF NO EXPRESSIONS, A SET TO NUMBER OF EXPRESSIONS
                ;       HI ORDER BIT SET IF ,B ALSO
0C8A CDD78B             CALL    GNC
                SCANEX, ;ENTER HERE IF CHARACTER ALREADY SCANNED
0C8D 21508F             LXI     H,EXPLIST
0C90 3600               MVI     M,0     ;ZERO EXPRESSIONS
0C92 23                 INX     H       ;READY TO FILL EXPRESSION LIST
0C93 FE0D               CPI     CR      ;END OF LINE?
0C95 CACF8C             JZ      SCANRET
                ;
                ;       NOT CR, MUST BE DIGIT OR COMMA
0C98 FE2C               CPI     ','
0C9A C2A08C             JNZ     SCANE0
                ;       MARK AS COMMA
0C9D 3E80               MVI     A,80H
0C9F 32508F             STA     EXPLIST
0CA2 110000             LXI     D,0
0CA5 C3AB8C             JMP     SCANE1
                ;
                SCANE0, ;NOT CR OR COMMA
0CA8 CD678C             CALL    GETEXP  ;EXPRESSION TO D,E
0CAB CD7F8C     SCANE1, CALL    SCSTORE ;STORE THE EXPRESSION AND INCREMENT H,L
0CAE FE0D               CPI     CR
0CB0 CACF8C             JZ      SCANRET
0CB3 CDD78B             CALL    GNC
0CB6 CD678C             CALL    GETEXP
0CB9 CD7F8C             CALL    SCSTORE
                ;       SECOND DIGIT SCANNED
0CBC FE0D               CPI     CR
0CBE CACF8C             JZ      SCANRET
0CC1 CDD78B             CALL    GNC
0CC4 CD678C             CALL    GETEXP
0CC7 CD7F8C             CALL    SCSTORE
0CCA FE0D               CPI     CR
0CCC C2A58B             JNZ     CERROR
                SCANRET,
0CCF 11508F             LXI     D,EXPLIST       ;LOOK AT COUNT
0CD2 1A                 LDAX    D               ;LOAD COUNT TO ACC
0CD3 FE81               CPI     81H             ;, WITHOUT B?
0CD5 CAA58B             JZ      CERROR
0CD8 13                 INX     D               ;READY TO EXTRACT EXPN'S
0CD9 B7                 ORA     A       ;ZERO FLAG MAY BE SET
```

```
0CDA 07                 RLC
0CDB 8F                 RRC             ;SET CARRY IF HO BIT SET (,B)
0CDC C9                 RET                     ;WITH FLAGS SET
                ;
                ;
                ;       SUBROUTINES FOR CPU STATE DISPLAY
                FLGSHF. ;SHIFT COMPUTATION FOR FLAG GIVEN BY REG-B
                ;       REG A CONTAINS FLAG UPON EXIT (UNSHIFTED)
                ;       REG C CONTAINS NUMBER OF SHIFTS REQUIRED+1
                ;       REGS D,E CONTAIN ADDRESS OF FLAGS IN TEMPLATE
0CDD E5                 PUSH    H
0CDE 21BD0D             LXI     H,FLGTAB        ;SHIFT TABLE
0CE1 58                 MOV     E,B
0CE2 1600               MVI     D,0
0CE4 19                 DAD     D
0CE5 4E                 MOV     C,M             ;SHIFT COUNT TO C
0CE6 21AD0F             LXI     H,FLOC          ;ADDRESS OF FLAGS
0CE9 7E                 MOV     A,M             ;TO REG A
0CEA EB                 XCHG                    ;SAVE ADDRESS
0CEB E1                 POP     H
0CEC C9                 RET
                ;
                GETFLG. ;GET FLAG GIVEN BY REG-B TO REG-A AND MASK
0CED CDDD0C             CALL    FLGSHF  ;BITS TO SHIFT IN REG C
0CF0 0D         GETFL0. DCR     C
0CF1 CAF80C             JZ      GETFL1
0CF4 1F                 RAR
0CF5 C3F00C             JMP     GETFL0
0CF8 E601       GETFL1. ANI     1B
0CFA C9                 RET
                ;
                GETDBA. ;GET DOUBLE BYTE ADDRESS CORRESPONDING TO REG-A TO HL
0CFB D606               SUI     BVAL    ;NORMALIZE TO 0,1,...
0CFD 21B00D             LXI     H,RINX  ;INDEX TO STACKED VALUES
0D00 5F                 MOV     E,A     ;INDEX TO E
0D01 1600               MVI     D,0     ;DOUBLE PRECISION
0D03 19                 DAD     D       ;INDEXED INTO VECTOR
0D04 5E                 MOV     E,M     ;OFFSET TO E
0D05 16FF               MVI     D,0FFH  ;-1
0D07 21B50F             LXI     H,STACK
0D0A 19                 DAD     D       ;HL HAS BASE ADDRESS
0D0B C9                 RET
                ;
                GETDBL. ;GET DOUBLE BYTE CORRESPONDING TO REG-A TO HL
0D0C CDFB0C             CALL    GETDBA  ;ADDRESS OF ELT IN HL
0D0F 5E                 MOV     E,M     ;LSB
0D10 23                 INX     H
0D11 56                 MOV     D,M     ;MSB
0D12 EB                 XCHG            ;BACK TO HL
0D13 C9                 RET
                ;
                DELT.   ;DISPLAY CPU ELEMENT GIVEN BY COUNT IN REG-B, ADDRESS IN H,L
0D14 7E                 MOV     A,M     ;GET CHARACTER
0D15 CDC10B             CALL    PCHAR   ;PRINT IT
0D18 78                 MOV     A,B     ;GET COUNT
0D19 FE05               CPI     AVAL    ;PAST A?
```



```
0D1B D2250D             JNC     DELT0   ;JMP IF NOT FLAG
                ;
                ;       DISPLAY FLAG
0D1E CDED0C             CALL    GETFLG  ;FLAG TO REG-A
0D21 CDF000             CALL    PNIB
0D24 C9                 RET
                ;
                DELT0.  ;NOT FLAG, DISPLAY = AND DATA
0D25 F5                 PUSH    PSW
0D26 3E3D               MVI     A,'='
0D28 CDC10B             CALL    PCHAR
0D2B F1                 POP     PSW
0D2C C2370D             JNZ     DELT1   ;JUMP IF NOT REG-A
                ;
                ;       REGISTER A, DISPLAY BYTE VALUE
0D2F 21AE0F             LXI     H,ALOC
0D32 7E                 MOV     A,M
0D33 CDFF0B             CALL    PBYTE
0D36 C9                 RET
                ;
                DELT1.  ;DOUBLE BYTE DISPLAY
0D37 CD0C0D             CALL    GETDBL  ;TO H,L
0D3A CD200C             CALL    PADDR   ;PRINTED
0D3D C9                 RET
                ;
                DSTATE. ;DISPLAY CPU STATE
0D3E 21AD0D             LXI     H,RVECT ;REGISTER VECTOR
0D41 0600               MVI     B,0     ;REGISTER COUNT
0D43 CD0F0C             CALL    CRLF
0D46 C5         DSTA0.  PUSH    B
0D47 E5                 PUSH    H
0D48 CD140D             CALL    DELT    ;ELEMENT DISPLAYED
0D4B E1                 POP     H       ;RVECT ADDRESS RESTORED
0D4C C1                 POP     B       ;COUNT RESTORED
0D4D 04                 INR     B       ;NEXT COUNT
0D4E 23                 INX     H       ;NEXT REGISTER
0D4F 78                 MOV     A,B     ;LAST COUNT?
0D50 FE0B               CPI     PVAL+1
0D52 D2600D             JNC     DSTA1   ;JMP IF PAST END
0D55 FE05               CPI     AVAL    ;BLANK AFTER?
0D57 DA460D             JC      DSTA0
                ;       YES, BLANK AND GO AGAIN
0D5A CD0F0B             CALL    BLANK
0D5D C3460D             JMP     DSTA0
                ;
                ;       READY TO SEND DECODED INSTRUCTION
                DSTA1.
0D60 CD0F0B             CALL    BLANK
0D63 CD7F0E             CALL    NBRK    ;COMPUTE BREAKPOINTS IN CASE OF TRACE
0D66 F5                 PUSH    PSW     ;SAVE EXPRESSION COUNT - B,C AND D,E HAVE
0D67 D5                 PUSH    D       ;SAVE BP ADDRESS
0D68 C5                 PUSH    B       ;SAVE AUX BREAKPOINT
0D69 CD0D09             CALL    CHKDIS  ;CHECK TO SEE IF DISASSEMBER IS HERE
0D6C D2800D             JNC     DCHEX   ;DISPLAY HEX IF NOT
                ;       DISASSEMBLE CODE
0D6F 2A0B0F             LHLD    PLOC    ;GET CURRENT PC
```


COPYRIGHT © 1978
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA. 93950
SER. # _____

```
0B72 220C0B             SHLD    DISPC   ;SET DISASSM PC
0B75 211000             LXI     H,DISPG;PAGE MODE = 0FFH TO TRACE
0B78 36FF               MVI     M,0FFH
0B7A CD0600             CALL    DISEN
0B7D C3A90D             JMP     DSTRET
                ;
                DCHEX,  ;DISPLAY HEX
0B80 2B                 DCX     H       ;POINT TO LAST TO WRITE
0B81 22590F             SHLD    DISMAX  ;SAVE FOR COMPARE BELOW
0B84 2AB30F             LHLD    PLOC    ;START ADDRESS OF TRACE
0B87 7E                 MOV     A,M     ;GET OPCODE
0B88 CDFF0B             CALL    PBYTE
0B8B 23                 INX     H       ;READY FOR NEXT BYTE
0B8C CD3F0C             CALL    DISCOM  ;ZERO SET IF ONE BYTE TO PRINT, CARRY IF MO
0B8F DAA90D             JC      DSTRET
0B92 F5                 PUSH    PSW     ;SAVE RESULT OF ZERO TEST
0B93 CDBF0B             CALL    BLANK   ;SEPARATOR
0B96 F1                 POP     PSW     ;RECALL ZERO TEST
0B97 B3                 ORA     E       ;ZERO TEST
0B98 CAA50D             JZ      DSTA2
                ;       DISPLAY DOUBLE BYTE
0B9B 5E                 MOV     E,M
0B9C 23                 INX     H
0B9D 56                 MOV     D,M
0B9E EB                 XCHG
0B9F CD200C             CALL    PADDR   ;PRINT ADDRESS
0BA2 C3A90D             JMP     DSTRET
                ;
                DSTA2,  ;PRINT BYTE VALUE
0BA5 7E                 MOV     A,M
0BA6 CDFF0B             CALL    PBYTE
                DSTRET,
0BA9 C1                 POP     B       ;AUX BREAKPOINT
0BAA D1                 POP     D       ;RESTORE BREAKPOINT
0BAB F1                 POP     PSW     ;RESTORE COUNT
0BAC C9                 RET
                ;
                ;       DATA VECTORS FOR CPU DISPLAY
0BAD 435A4D4549RVECT,   DB      'CZMEIABDHSP'
0BB8 F6         RINX,   DB      (BLOC-STACK) AND 0FFH   ;LOCATION OF BC
0BB9 F4                 DB      (DLOC-STACK) AND 0FFH   ;LOCATION OF DE
0BBA FC                 DB      (HLOC-STACK) AND 0FFH   ;LOCATION OF HL
0BBB FA                 DB      (SLOC-STACK) AND 0FFH   ;LOCATION OF SP
0BBC FE                 DB      (PLOC-STACK) AND 0FFH   ;LOCATION OF PC
                ;       FLGTAB ELEMENTS DETERMINE SHIFT COUNT TO SET/EXTRACT FLAGS
0BBD 0107080305FLGTAB:  DB      1,7,8,3,5       ;CY, ZER, SIGN, PAR, IDCY
                ;
                CLRTRACE,       ;CLEAR THE TRACE FLAG
0BC2 210000             LXI     H,0
0BC5 22470F             SHLD    TRACER
0BC8 C9                 RET
                ;
                BREAKP, ;ARRIVE HERE WHEN PROGRAMMED BREAK OCCURS
0BC9 F3                 DI
0BCA 22B10F             SHLD    HLOC    ;HL SAVED
0BCD E1                 POP     H       ;RECALL RETURN ADDRESS
```



```
0BCE 2B                 DCX     H       ;DECREMENT FOR RESTART
0BCF 22B30F             SHLD    PLOC
                ;       DAD SP BELOW DESTROYS CY, SO SAVE AND RECALL
0BD2 F5                 PUSH    PSW     ;INTO USER'S STACK
0BD3 210200             LXI     H,2     ;BIAS SP BY 2 BECAUSE OF PUSH
0BD6 39                 DAD     SP      ;SP IN HL
0BD7 F1                 POP     PSW     ;RESTORE CY AND FLAGS
0BD8 31B10F             LXI     SP,STACK-4;LOCAL STACK
0BDB E5                 PUSH    H       ;SP SAVED
0BDC F5                 PUSH    PSW
0BDD C5                 PUSH    B
0BDE D5                 PUSH    D
                ;       MACHINE STATE SAVED, CLEAR BREAK POINTS
0BDF 2AB30F             LHLD    PLOC    ;CHECK FOR RST INSTRUCTION
0BE2 7E                 MOV     A,M     ;OPCODE TO A
0BE3 FEFF               CPI     RSTIN
                ;       SAVE CONDITION CODES FOR LATER TEST
0BE5 F5                 PUSH    PSW
                ;       SAVE PLOC FOR LATER INCREMENT OR DECREMENT
0BE6 E5                 PUSH    H
                ;
                ;       CLEAR BREAKPOINTS WHICH ARE PENDING
0BE7 21490F             LXI     H,BREAKS
0BEA 7E                 MOV     A,M
0BEB 3600               MVI     M,0     ;SET TO ZERO BREAKS
0BED B7         CLER0,  ORA     A       ;ANY MORE?
0BEE CAFE0D             JZ      CLER1
0BF1 3D                 DCR     A
0BF2 47                 MOV     B,A     ;SAVE COUNT
0BF3 23                 INX     H       ;ADDRESS OF BREAK
0BF4 5E                 MOV     E,M     ;LOW ADDR
0BF5 23                 INX     H
0BF6 56                 MOV     D,M     ;HIGH ADDR
0BF7 23                 INX     H
0BF8 7E                 MOV     A,M     ;INSTRUCTION
0BF9 12                 STAX    D       ;BACK TO PROGRAM
0BFA 78                 MOV     A,B     ;RESTORE COUNT
0BFB C3ED0D             JMP     CLER0
                ;
                CLER1,  ;CLEARED, CONTINUE TRACING, OR STOP EXECUTION
0BFE E1                 POP     H       ;RESTORE PLOC
0BFF F1                 POP     PSW     ;RESTORE CONDITION FLAGS
0E00 CA220E             JZ      BREAK0  ;BRANCH IF PROGRAMMED INTERRUPT
                ;
                ;       MUST BE FRONT PANEL INTERRUPT, CHECK IF IN BDOS
0E03 23                 INX     H       ;DON'T DECREMENT ON PANEL INTERRUPT
0E04 22B30F             SHLD    PLOC    ;RESTORE TO NEXT LOGICAL INSTRUCTION
0E07 EB                 XCHG            ;TO D,E FOR COMPARE
0E08 21A806             LXI     H,TRAPJMP+1
0E0B 4E                 MOV     C,M     ;LOW BDOS ADDR
0E0C 23                 INX     H
0E0D 46                 MOV     B,M     ;HIGH BDOS ADDR
0E0E CD5708             CALL    BCDE    ;CY IF BDOS>PLOC
0E11 DA220E             JC      BREAK0  ;BRANCH IF PLOC <= BDOS
                ;
                ;       IN THE BDOS, DON'T BREAK UNTIL THE RETURN OCCURS
```

```
0E14 CDC20D             CALL    CLRTRACE;CLEAR TRACE FLAGS
0E17 2A440F             LHLD    RETLOC  ;TRAPPED RETLOC ON ENTRY TO DOS
0E1A EB                 XCHG            ;TO D,E READY FOR BREAKPOINT
0E1B 3E82               MVI     A,82H   ;LOOKS LIKE G,8888
0E1D B7                 ORA     A       ;SETS FLAGS
0E1E 37                 STC             ;SUBSEQUENT TEST FOR CY
0E1F C38508             JMP     GOPR    ;START PROGRAM EXECUTION, WITH BREAKPOINT
                ;
                BREAK0, ;NORMAL BREAKPOINT
0E22 FB                 EI
0E23 2A470F             LHLD    TRACER
0E26 7C                 MOV     A,H
0E27 B5                 ORA     L
0E28 CA400E             JZ      STOPEX
                ;
                ;       TRACE IS ON
0E2B 2B                 DCX     H
0E2C 22470F             SHLD    TRACER
0E2F CD198C             CALL    BREAK   ;BREAK KEY DEPRESSED?
0E32 C2400E             JNZ     STOPEX
0E35 3A460F             LDA     TMODE   ;TRACE MODE T IF 0FFH
0E38 B7                 ORA     A
0E39 C2428E             JNZ     BREAK1
                ;       NOT TRACING, BUT MONITORING, SO SET BREAKPOINTS
0E3C CD7F0E             CALL    NBRK
0E3F C38508             JMP     GOPR
                ;
                BREAK1, ;TRACING AND MONITORING
0E42 CD3E8D             CALL    DSTATE  ;STATE DISPLAYED, CHECK FOR BREAKPOINTS
0E45 C38508             JMP     GOPR    ;STARTS EXECUTION
                ;
                STOPEX,
0E48 CDC20D             CALL    CLRTRACE        ;TRACE FLAGS GO TO ZERO
0E4B 3E2A               MVI     A,'*'
0E4D CDC18B             CALL    PCHAR
0E50 2AB30F             LHLD    PLOC
                ;       CHECK TO ENSURE DISASSEMBLER IS PRESENT
0E53 CD8D89             CALL    CHKDIS
0E56 D25C0E             JNC     STOP0
0E59 220C00             SHLD    DISPC
0E5C CD200C     STOP0,  CALL    PADDR
0E5F 2AB10F             LHLD    HLOC
0E62 22570F             SHLD    DISLOC
0E65 C3FE86             JMP     START
                ;
                CAT,    ;DETERMINE OPCODE CATEGORY - CODE IN REGISTER B
                ;       D,E CONTAIN DOUBLE PRECISION CATEGORY NUMBER ON RETURN
0E69 110D00             LXI     D,OPMAX ;D=0,E=OPMAX
0E6B 21290F             LXI     H,OPLIST
0E6E 7E         CAT0,   MOV     A,M             ;MASK TO A
0E6F A0                 ANA     B       ;MASK OPCODE FROM B
0E70 23                 INX     H       ;READY FOR COMPARE
0E71 BE                 CMP     M       ;SAME AFTER MASK?
0E72 23                 INX     H       ;READY FOR NEXT COMPARE
0E73 CA7B0E             JZ      CAT1    ;EXIT IF COMPARED OK
0E76 14                 INR     D       ;UP COUNT IF NOT MATCHED
```



```
0E77 1D                 DCR     E       ;FINISHED?
0E78 C26E8E             JNZ     CAT0
0E7B 5A         CAT1,   MOV     E,D     ;E IS CATEGORY NUMBER
0E7C 1600               MVI     D,0     ;DOUBLE PRECISION
0E7E C9                 RET
                ;
                NBRK,   ;FIND NEXT BREAK POINT ADDRESS
                ;       UPON RETURN, REGISTER A IS SETUP AS IF USER TYPED G,B1,B2
                ;       G,B1 DEPENDING UPON OPERATOR CATEGORY.  B,C CONTAINS SECON
                ;       D,E CONTAINS PRIMARY BP.  HL ADDRESS NEXT OPCODE BYTE
0E7F 2AB30F             LHLD    PLOC
0E82 46                 MOV     B,M     ;GET OPERATOR
0E83 23                 INX     H       ;HL ADDRESS BYTE FOLLOWING OPCODE
0E84 E5                 PUSH    H       ;SAVE IT FOR LATER
0E85 CD688E             CALL    CAT     ;DETERMINE OPERATOR CATEGORY
0E88 21430F             LXI     H,CATNO ;SAVE CATEGORY NUMBER
0E8B 73                 MOV     M,E
0E8C 21968E             LXI     H,CATTAB;CATEGORY TABLE BASE
0E8F 19                 DAD     D       ;INXED
0E90 19                 DAD     D       ;INXED*2
0E91 5E                 MOV     E,M     ;LOW BYTE TO E
0E92 23                 INX     H
0E93 56                 MOV     D,M     ;HIGH BYTE TO D
0E94 EB                 XCHG
0E95 E9                 PCHL            ;JUMP INTO TABLE
0E96 B28E       CATTAB, DW      JMPOP   ;JUMP OPERATOR
0E98 DA8E               DW      CCOP    ;JUMP CONDITIONAL
0E9A B20E               DW      JMPOP   ;CALL OPERATOR (TREATED AS JMP)
0E9C DA8E               DW      CCOP    ;CALL CONDITIONAL
0E9E B88E               DW      RETOP   ;RETURN FROM SUBROUTINE
0EA0 EC0E               DW      RSTOP   ;RESTART
0EA2 FE8E               DW      PCOP    ;PCHL
0EA4 200F               DW      IMOP    ;SINGLE PRECISION IMMEDIATE (2 BYTE)
0EA6 200F               DW      IMOP    ;ADI ... CPI
0EA8 1D0F               DW      DIMOP   ;DOUBLE PRECISION IMMEDIATE (3 BYTES)
0EAA 1D0F               DW      DIMOP   ;LHLD ... STA
0EAC 130F               DW      RCOND   ;RETURN CONDITIONAL
0EAE 200F               DW      IMOP    ;IN/OUT
                ;       NEXT DW MUST BE THE LAST IN THE SEQUENCE
0EB0 0E0F               DW      SIMOP   ;SIMPLE OPERATOR (1 BYTE)
                ;
                JMPOP,  ;GET OPERAND FIELD, CHECK FOR BDOS
0EB2 CDC08E             CALL    GETOPA  ;GET OPERAND ADDRESS TO D,E AND COMPARE WI
0EB5 C2230F             JNZ     ENDOP   ;TREAT AS SIMPLE OPERATOR IF NOT BDOS
                ;       OTHERWISE, TREAT AS A RETURN INSTRUCTION
0EB8 CDD30E     RETOP,  CALL    GETSP   ;ADDRESS AT STACKTOP TO D,E
0EBB C3230F             JMP     ENDOP   ;TREAT AS SIMPLE OPERATOR
                ;
                CBDOS,  ;COMPARE D,E WITH BDOS ADDRESS, RETURN ZERO FLAG IF EQUAL
0EBE 3AA8B6             LDA     TRAPJMP+1
0EC1 BB                 CMP     E
0EC2 C0                 RNZ
0EC3 3AA9B6             LDA     TRAPJMP+2
0EC6 BA                 CMP     D
0EC7 C9                 RET
                ;
```

```
                GETOPA: ;GET OPERAND ADDRESS AND COMPARE WITH BDOS
0EC8 C1                 POP     B       ;GET RETURN ADDRESS
0EC9 E1                 POP     H       ;GET OPERAND ADDRESS
0ECA 5E                 MOV     E,M
0ECB 23                 INX     H
0ECC 56                 MOV     D,M
0ECD 23                 INX     H
0ECE E5                 PUSH    H       ;UPDATED PC INTO STACK
0ECF C5                 PUSH    B       ;RETURN ADDRESS TO STACK
0ED0 C38E0E             JMP     CBDOS   ;RETURN THROUGH CBDOS WITH ZERO FLAG SET
                ;
                GETSP:  ;GET RETURN ADDRESS FROM USER'S STACK TO D,E
0ED3 2AAF0F             LHLD    SLOC
0ED6 5E                 MOV     E,M
0ED7 23                 INX     H
0ED8 56                 MOV     D,M
0ED9 C9                 RET
                ;
                CCOP:   ;CALL CONDITIONAL OPERATOR
0EDA CDC80E             CALL    GETOPA  ;GET OPERAND ADDRESS TO D,E, COMPARE WITH B
0EDD CAE70E             JZ      CCOP1
                ;       NOT THE BDOS, BREAK AT OPERAND ADDRESS AND NEXT ADDRESS
0EE0 C1                 POP     B       ;NEXT ADDRESS TO B,C
0EE1 C5                 PUSH    B       ;BACK TO STACK
0EE2 3E02               MVI     A,2     ;TWO BREAKPOINTS
0EE4 C3250F             JMP     RETCAT  ;RETURN FROM NBRK
                ;
                CCOP1:  ;BREAK ADDRESS AT NEXT LOCATION ONLY, WAIT FOR RETURN FROM B
0EE7 D1                 POP     D
0EE8 D5                 PUSH    D       ;BACK TO STACK
0EE9 C3238F             JMP     ENDOP   ;ONE BREAKPOINT ADDRESS
                ;
                RSTOP:  ;RESTART INSTRUCTION - CHECK FOR RST 7
0EEC 78                 MOV     A,B
0EED FEFF               CPI     RSTIN   ;RESTART INSTRUCTION USED FOR SOFT INT
0EEF C2F60E             JNZ     RST0
                ;
                ;       SOFT RST, NO BREAK POINT SINCE IT WILL OCCUR IMMEDIATELY
0EF2 AF                 XRA     A
0EF3 C3270F             JMP     RETCAT1 ;ZERO ACCUMULATOR
0EF6 E638       RST0:   ANI     111000B ;GET RESTART NUMBER
0EF8 5F                 MOV     E,A
0EF9 1600               MVI     D,0     ;DOUBLE PRECISION BREAKPOINT TO D,E
0EFB C3238F             JMP     ENDOP
                ;
                PCOP:   ;PCHL
0EFE 2A810F             LHLD    HLOC
0F01 EB                 XCHG    ;HL VALUE TO D,E FOR BREAKPOINT
0F02 CDBE0E             CALL    CBDOS   ;BDOS VALUE?
0F05 C2230F             JNZ     ENDOP
                ;       PCHL TO BDOS, USE RETURN ADDRESS
0F08 C3B00E             JMP     RETOP
                ;
0F0B C3238F             JMP     ENDOP
                ;
                SIMOP:  ;SIMPLE OPERATOR, USE STACKED PC
```



```
0F0E D1                 POP     D
0F0F D5                 PUSH    D
0F10 C3238F             JMP     ENDOP
                ;
                RCOND:  ;RETURN CONDITIONAL
0F13 CDD30E             CALL    GETSP   ;GET RETURN ADDRESS FROM STACK
0F16 C1                 POP     B       ;B,C ALTERNATE LOCATION
0F17 C5                 PUSH    B       ;REPLACE IT
0F18 3E02               MVI     A,2
0F1A C3250F             JMP     RETCAT  ;TO SET FLAGS AND RETURN
                ;
                DIMOP:  ;DOUBLE PRECISION IMMEDIATE OPERATOR
0F1D D1                 POP     D
0F1E 13                 INX     D       ;INCREMENTED ONCE, DROP THRU FOR ANOTHER
0F1F D5                 PUSH    D       ;COPY BACK
                ;
                IMOP:   ;SINGLE PRECISION IMMEDIATE
0F20 D1                 POP     D
0F21 13                 INX     D
0F22 D5                 PUSH    D
                ;
                ENDOP:  ;END OPERATOR SCAN
0F23 3E01               MVI     A,1     ;SINGLE BREAKPOINT
                RETCAT: ;RETURN FROM NBRK
0F25 3C                 INR     A       ;COUNT UP FOR CALLER
0F26 37                 STC
                RETCAT1:
0F27 E1                 POP     H       ;RECALL NEXT ADDRESS
0F28 C9                 RET
                ;
                ;
                ;       OPCODE CATEGORY TABLES
0F29 FFC3       OPLIST: DB      1111$1111B,     1100$0011B      ;0 JMP
0F2B C7C2               DB      1100$0111B,     1100$0010B      ;1 JCOND
0F2D FFCD               DB      1111$1111B,     1100$1101B      ;2 CALL
0F2F C7C4               DB      1100$0111B,     1100$0100B      ;3 CCOND
0F31 FFC9               DB      1111$1111B,     1100$1001B      ;4 RET
0F33 C7C7               DB      1100$0111B,     1100$0111B      ;5 RST 0..7
0F35 FFE9               DB      1111$1111B,     1110$1001B      ;6 PCHL
0F37 C706               DB      1100$0111B,     0000$0110B      ;7 MVI
0F39 C7C6               DB      1100$0111B,     1100$0110B      ;8 ADI...CPI
0F3B CF01               DB      1100$1111B,     0000$0001B      ;9 LXI
0F3D E722               DB      1110$0111B,     0010$0010B      ;10 LHLD SHLD LDA
0F3F C7C0               DB      1100$0111B,     1100$0000B      ;11 RCOND
0F41 F7D3               DB      1111$0111B,     1101$0011B      ;IN OUT
000D =          OPMAX   EQU     ($-OPLIST)/2
                ;
0F43            CATNO:  DS      1       ;CATEGORY NUMBER SAVED IN NBRK
0F44            RETLOC: DS      2       ;RETURN ADDRESS TO USER FROM BDOS
0F46            TMODE:  DS      1       ;TRACE MODE
0F47            TRACER: DS      2       ;TRACE COUNT
0F49            BREAKS: DS      7       ;#BREAKS/BKPT1/DAT1/BKPT2/DAT2
0F50            EXPLIST:DS      7       ;COUNT+(EXP1)(EXP2)(EXP3)
0F57            DISLOC: DS      2       ;DISPLAY LOCATION
0F59            DISMAX: DS      2       ;MAX VALUE FOR CURRENT DISPLAY
```

```
0F5B            TDISP:  DS      2       ;TEMP 16 BIT LOCATION
0F5D            NEXTCOM:DS      2       ;NEXT LOCATION FROM COMMAND BUFFER
0F5F 20         COMLEN: DB      CSIZE   ;MAX COMMAND LENGTH
0F60            CURLEN: DS      1       ;CURRENT COMMAND LENGTH
0F61            COMBUF: DS      CSIZE   ;COMMAND BUFFER
0F81            MLOAD:  DS      2       ;MAX LOAD ADDRESS
0F83                    DS      SSIZE   ;STACK AREA
                STACK:
0FB3 =          PLOC    EQU     STACK-2 ;PC IN TEMPLATE
0FB1 =          HLOC    EQU     STACK-4 ;HL
0FAF =          SLOC    EQU     STACK-6 ;SP
0FAE =          ALOC    EQU     STACK-7 ;A
0FAD =          FLOC    EQU     STACK-8 ;FLAGS
0FA8 =          BLOC    EQU     STACK-10        ;BC
0FA9 =          DLOC    EQU     STACK-12;D,E
                ;
0FB5 00                 NOP             ;FOR RELOCATION BOUNDARY
0FB6                    END
```



```
0FAE ALOC       0009 ASSEM      077E ASSM       0005 AVAL       0857 BCDE
1686 BDOS       0005 BDOSE      06AA BEGIN      0BBF BLANK      0FA8 BLOC
0E22 BREAK0     0E42 BREAK1     0686 BREAKA     0C19 BREAK      0DC9 BREAKP
0F49 BREAKS     0006 BVAL       0E68 CAT        0E6E CAT0       0E7B CAT1
0F43 CATNO      0E96 CATTAB     0EBE CBDOS      0EDA CCOP       0EE7 CCOP1
0BA5 CERROR     098D CHKDIS     0A08 CHKIO      0001 CIF        0983 CKMLOAD
0DED CLER0      0DFE CLER1      0DC2 CLRTRACE   0002 COF        0F61 COMBUF
0F5F COMLEN     097B COMLOAD    000D CR         0C0F CRLF       0020 CSIZE
0F60 CURLEN     0000 DBF        0058 DBP        0D80 DCHEX      0000 DEBUG
0BA0 DEF        0C4A DELIM      0D14 DELT       0D25 DELT0      0D37 DELT1
0680 DEMON      081A DEOF       085C DFCB       0F1D DIMOP      070D DISASMOK
0819 DISCH      0823 DISCH0     0C3F DISCOM     0086 DISEN      0003 DISIN
0B6E DISKR      0F57 DISLOC     0F59 DISMAX     07D5 DISP0      07E5 DISP1
07F0 DISP2      07F3 DISP3      0B05 DISP4      000C DISPC      0010 DISPG
07C6 DISPLAY    000E DISPN      0FA9 DLOC       001A DMAF       0D46 DSTA0
0D60 DSTA1      0DA5 DSTA2      0D3E DSTATE     0DA9 DSTRET     0007 DVAL
0F23 ENDOP      0ABF ETRACE     0AEF EXAM0      0AF5 EXAM1      0B02 EXAM2
0B53 EXAM4      0B63 EXAM5      0AE1 EXAMINE    0F50 EXPLIST    0000 FALSE
005C FCB        0020 FCR        0000 FDN        0001 FFN        0009 FFT
005C FILL       0064 FILL0      0926 FLB        0948 FLB1       0CDD FLGSHF
0DBD FLGTAB     0021 FLN        0FAD FLOC       0913 FLP        093A FLP1
0955 FLZ        000F FRC        000C FRL        0BB0 GETBUFF    0CFB GETDBA
0D0C GETDBL     0C67 GETEXP     0C6B GETEXP0    000A GETF       0CF0 GETFL0
0CFB GETFL1     0CED GETFLG     0EC8 GETOPA     0ED3 GETSP      0C60 GETVAL
0BD7 GNC        0BEE GNCRET     000F GOP0       00A1 GOP1       0005 GOPR
0070 GOTO       00DA HEXAR1     0C53 HEXCON     0FB1 HLOC       09DB HREAD
0008 HVAL       0007 IDS        0F20 IMOP       0904 INFCB      0E02 JMPOP
0737 JMPTAB     0797 LASSM      09BA LCOM0      09CD LCOM1      000A LF
000C LIFT       0A69 LMSG       0005 LOF        0B4D LROTATE    0F01 MLOAD
0000 MODBAS     095A MOVE       095D MOVE0      0E7F NBRK       008E NDI
0F5D NEXTCOM    000F OPF        0F29 OPLIST     00BD OPMAX      076B OPN
0C28 PADDR      0C27 PADDX      0BFF PBYTE      0100 PCBASE     08C1 PCHAR
0EFE PCOP       0C30 PGRAPH     0FB3 PLOC       0BF0 PNIB       0BFA PNIBH
0004 POF        0C3A PPERIOD    000C PSIZE      000A PVAL       096A QHEX
0A20 RBYTE      0F13 RCOND      0B79 RDC        0014 RDF        0071 RDI
0A06 RDTYPE     0996 READ       09A6 READN      0A89 RED1       FFFF RELOC
0F25 RETCAT     0F27 RETCAT1    0F44 RETLOC     0E80 RETOP      0003 RIF
09A7 RINIT      0D88 RINX       0A48 RL10       0A54 RL11       0A40 RLIFT
0BA1 RRET       0EF6 RST0       08FF RSTIN      0038 RSTLOC     0007 RSTNUM
0EEC RSTOP      0DAD RVECT      0841 SCAN3      0CA8 SCANE0     0CAB SCANE1
0C0D SCANEX     0C8A SCANEXP    0CCF SCANRET    0C7F SCSTORE    00B2 SETBK
08CD SETBK0     0A7E SETM0      0AB5 SETM1      0A74 SETMEM     0F0E SIMOP
0FAF SLOC       07BD SPAGE      07B8 SPAGE      0100 SPBASE     0032 SSIZE
0FB5 STACK      06FE START      0E5C STOP0      0E48 STOPEX     0009 SVAL
0F50 TDISP      0930 TFT        0F46 TMODE      0AD8 TRAC0      0ABD TRACE
0F47 TRACER     0BCE TRANS      06A2 TRAPAD     06A7 TRAPJMP    FFFF TRUE
0AB9 UNTRACE
```

```
                ;       ASSEMBLY LANGUAGE VERSION OF MEM$MOVE FOR ED SPEEDUP
                ;       VERSION 2.0 OF ED
                ;
13CA =          MEM$MOVE        EQU     13CAH
1D34 =          MOVEFLAG        EQU     1D34H
1D20 =          DIRECTION       EQU     1D20H
1D22 =          FRONT           EQU     1D22H
1D24 =          BACK            EQU     1D24H
1D26 =          FIRST           EQU     1D26H
1D28 =          LAST            EQU     1D28H
1C10 =          BASELINE        EQU     1C10H
1D4D =          MEMORY          EQU     1D4DH
                ;
0001 =          FORWARD         EQU     1
000A =          LF              EQU     0AH
                ;
13CA                    ORG     MEM$MOVE
13CA 21341D             LXI     H,MOVEFLAG
13CD 71                 MOV     M,C     ;1 = MOVE DATA
13CE 114D1D             LXI     D,MEMORY
13D1 2A221D             LHLD    FRONT
13D4 19                 DAD     D       ;MEMORY+FRONT
13D5 E5                 PUSH    H
13D6 2A241D             LHLD    BACK
13D9 19                 DAD     D
13DA E5                 PUSH    H
13DB 3A201D             LDA     DIRECTION
13DE FE01               CPI     FORWARD
13E0 C21214             JNZ     MOVEBACK
13E3 2A281D             LHLD    LAST
13E6 79                 MOV     A,C     ;MOVEFLAG TO A
13E7 1F                 RAR
13E8 DAF113             JC      MOVEFORW
                ;       SET BACK TO LAST
13EB 22241D             SHLD    BACK
13EE E1                 POP     H
13EF E1                 POP     H
13F0 C9                 RET
                ;
                MOVEFORW:
13F1 19                 DAD     D       ;MEMORY+LAST
13F2 44                 MOV     B,H
13F3 4D                 MOV     C,L
13F4 E1                 POP     H
13F5 D1                 POP     D       ;BC=LAST, DE=FRONT, HL=BACK
13F6 7D         MOVEF:  MOV     A,L     ;BACK < LAST?
13F7 91                 SUB     C
13F8 7C                 MOV     A,H
13F9 98                 SBB     B       ;CY IF TRUE
13FA D24314             JNC     EMOVE
13FD 23                 INX     H       ;BACK=BACK+1
13FE 7E                 MOV     A,M     ;CHAR TO A
13FF FE0A               CPI     LF      ;END OF LINE?
1401 C20D14             JNZ     NOTLFF
1404 E5                 PUSH    H
1405 2A101C             LHLD    BASELINE
1408 23                 INX     H       ;BASELINE=BASELINE+1
1409 22101C             SHLD    BASELINE
140C E1                 POP     H
                NOTLFF:
140D 12                 STAX    D       ;TO FRONT
140E 13                 INX     D       ;FRONT=FRONT+1
140F C3F613             JMP     MOVEF
                ;
                MOVEBACK:
1412 2A261D             LHLD    FIRST
1415 19                 DAD     D       ;MEMORY+FIRST
1416 44                 MOV     B,H
1417 4D                 MOV     C,L
1418 E1                 POP     H
1419 D1                 POP     D       ;BC=FIRST, DE=FRONT, HL=LAST
141A 79         MOVEB:  MOV     A,C     ;FIRST > FRONT?
141B 93                 SUB     E
141C 78                 MOV     A,B
141D 9A                 SBB     D       ;CY IF TRUE
141E D24314             JNC     EMOVE
1421 1B                 DCX     D       ;FRONT=FRONT-1
1422 1A                 LDAX    D       ;CHAR TO A
1423 FE0A               CPI     LF
1425 C23114             JNZ     NOTLFB
1428 E5                 PUSH    H
1429 2A101C             LHLD    BASELINE
142C 2B                 DCX     H       ;BASELINE=BASELINE-1
142D 22101C             SHLD    BASELINE
1430 E1                 POP     H
1431 F5         NOTLFB: PUSH    PSW     ;SAVE CHAR
1432 3A341D             LDA     MOVEFLAG
1435 1F                 RAR
1436 D23F14             JNC     NOMOVE
1439 F1                 POP     PSW
143A 77                 MOV     M,A     ;STORE TO BACK
143B 2B                 DCX     H
143C C31A14             JMP     MOVEB
143F F1         NOMOVE: POP     PSW
1440 C31A14             JMP     MOVEB
                ;
1443 D5         EMOVE:  PUSH    D
1444 11B3E2             LXI     D,-MEMORY
1447 19                 DAD     D       ;RELATIVE VALUE OF BACK
1448 22241D             SHLD    BACK
144B E1                 POP     H
144C 19                 DAD     D       ;RELATIVE VALUE OF FRONT
144D 22221D             SHLD    FRONT
1450 C9                 RET
1451                    END
```

```
1D24 BACK       1C10 BASELINE   1D20 DIRECTION  1443 EMOVE      1D26 FIRST
0001 FORWARD    1D22 FRONT      1D28 LAST       000A LF         13CA MEMMOVE
1D4D MEMORY     1412 MOVEBACK   141A MOVEB      1D34 MOVEFLAG   13F1 MOVEFORW
13F6 MOVEF      143F NOMOVE     1431 NOTLFB     140D NOTLFF
```

```
                        ;ASM PATCH OF 8/9,79
                ;
                ;       THIS PATCH FIXES AN ERROR WHICH OCCURS IN STATEMENTS
                ;       INVOLVING STRING CONSTANTS WHCIH INCLUDE LOWER CASE.
                ;       PREVIOUSLY ALL LOWER CASE CHARACTERS WERE TRANSLATED
                ;       TO UPPER CASE AUTOMATICALLY - AFTER MAKING THIS PATCH
                ;       PATCH, ASM WILL NOT PERFORM CHARACTER TRANSLATION
                ;       WITHIN STRING CONSTANTS.
                ;
                ;       THIS PATCH ALSO CHANGES VERSION NUMBER TO 2.0
                ;
0185 =          TOKEN   EQU     0185H   ;CURRENT SCANNER TOKEN
0003 =          STRNG   EQU     03H     ;STRING CONSTANT TOKEN
119E =          TRANS   EQU     119EH   ;SUBROUTINE TO TRANSLATE
11AD =          GNCN    EQU     11ADH   ;LOCATION OF PATCHED CODE
132D =          PATCH   EQU     132DH   ;LOCATION OF FREE AREA
0FA0 =          TITL    EQU     0FA0H   ;ASSEMBLER HEADING MESSAGE
                ;
11B2                    ORG     GNCN+6
11B3 C32D13             JMP     PATCH
132D                    ORG     PATCH
132D F5                 PUSH    PSW
132E 3A8501             LDA     TOKEN
1331 FE03               CPI     STRNG
1333 C49E11             CNZ     TRANS   ;TRANSLATE IF NOT STRING
1336 F1                 POP     PSW
1337 C9                 RET
                ;
0FB5                    ORG     TITL+21
0FB5 322E30     VERS:   DB      '2.0'
                ;
0FB8                    END
```

```
11AD GNCN       132D PATCH      0003 STRNG      0FA0 TITL       0185 TOKEN
119E TRANS      0FB5 VERS
```



```
                ;
                ;       DDT PATCH CHANGES VERSION NUMBER TO CORRESPOND TO
                ;       RELEASE 2.0 OF CP/M.
                ;
0014 =          VERSION EQU     20      ;2.0
0139 =          VERS    EQU     0139H   ;LOCATION OF VERSION NUMBER IN MOVDDT
0139                    ORG     VERS
0139 322F               DB      VERSION/10+'0','.'
013B 30                 DB      VERSION MOD 10 + '0'
013C                    END
```

```
0014 VERSION    0139 VERS
```