SMP 80.0

FLAGS AND LINKS

0	F780	CURLIN	CURRENT CURSOR LINE	$0-23_{10}$
+1	81	CURCOL	" " COLUMN	$0-79_{10}$
+2	82	RVFLG	REVERSE VIDEO FLAG	
+3	83	TXTFLG	04 = ASCII 00 = GRAPHIC 08 = EXTENDED	
+4	84	INSRTF	CHARACTER INSERTION FLAG ↑T → FF	
+5	85	ESCFLG	NON ZERO INDICATES ESCAPE SEQUENCE	
+6	86	ESCSTOR	STORAGE FOR ESCAPE SEQUENCE CHARACTERS	
+7	87	CTLNK	CONTROL TABLE LINKAGE ADDRESS	
+8	88		HI BYTE MUST BE NON ZERO	
+9	89	ESCLNK	ESCAPE TABLE LINKAGE	
+10	8A			
+11	8B	COMLNK	COMMAND TABLE LINKAGE	
+12	8C			
+13	8D	IO		
+14	8E			
+15	8F			
+16	F790	CURPTR	CURRENT CURSOR ADDRESS	
+17	91	PRTCTFL	PROTECTED FIELD FLAG. FF = PROTECT	
+18	92	BLANK	SHOULD = 20H	
+19	93	INPUT	TEMPORARY STORAGE FOR BYTE INPUT	
+20	94	MODEFLG	VIO MODE BYTE	
+21	95	TAB	TAB POSITIONS	
+22	96			
+23	97			
+24	98			
+25	99			
+26	9A			
+27	9B			
+28	9C			
+29	9D			
+30	9E			
+31	9F	COLS	COLUMNS SELECTED FOR DISPLAY	
+32	F7A0	MUST BE 0		
+33	A1	LINES	LINES SELECTED FOR DISPLAY	
+34	A2	NUMCHAR	CHARACTERS IN DISPLAY	
+35	A3		12/40 = 1E0 24/40 12/80 = 3C0 24/80 = 780	
+36	A4		cf TAB P.17 THISFLG	
+37	A5			
+38	A6	OFFSCRN	ADDRESS OF FIRST POSITION OFF SCREEN	
+39	A7		12/40 = F1E0 12/80, 24/40 = F3C0 24/80 = F780	
	F7FF	VIOCOM	VIO COMMAND PORT	
	F7FE	STACK	DEFAULT STACK POINTER	

```
F800    JMP     INITVIO     INITIALIZE VIO
F803    JMP     CHAROUT     CHARACTER OUTPUT ROUTINE
F806    JMP     INITMON     INITIALIZE MONITOR

                TEST        TEST VIO FUNCTIONS

F809 TEST   LXI     SP, STACK
            CALL    INITVIO
            CALL    CHARIN      GET INPUT WITH ECHO
F814        JMP     TEST
```

INITVIO

REQUIRE AN EQUATE OF INITIAL WITH DESIRED INITIAL VIO MODE BYTE

```
F815 INITVIO    PUSH    H
                PUSH    D
                PUSH    B
                PUSH    PSW
                LXI     H, CURLIN   POINT TO CURSOR LINE
                MVI     B, 1FH      PREPARE TO INITIALIZE FLAGS
                XRA     A           ZERO ACCUMULATOR
INIT1           MOV     M, A
                INX     H           POINT TO NEXT
                DCR     B           DECREMENT COUNT
                JNZ     INIT1       DO ANOTER FOR 1F(H) BYTES (31D)
                LXI     H, SCANBEG
                SHLD    CURPTR      CURSOR ADDRESS TO POSITION 1
                LXI     H, 780H     NUMBER OF CHARACTERS TO INITIAL
                SHLD    NUMCHAR
                CALL    ERASE
                MVI     A, INITIAL  INITIAL VIO MODE BYTE
                LXI     D, DSPEX    PREPARE FOR INDIRECT CALL
                PUSH    D
INIT7           STA     MODEFLG     STORE MODE BYTE
                STA     VIOCOM      MODE BY TO VIO CONTROL PORT
                CMA                 CHECK FOR # OF COLUMNS INDICATED
                ANI     03          BY MODE BYTE
                RRC                 CARRY = 80 COLUMNS  NO CARRY = 40 COL
```

(2)

```
                LXI  H, 0028H      LOAD 40 COLS
                INC  INIT2
                DAD  H             DOUBLE TO 80 COLS
INIT2           SHLD COLS
                LXI  H, LINES      POINT TO LINES
                MVI  M, 11D        SET TO 11 INITIALLY
                RAR                CHECK LINES IN MODE BYTE
                JNC  INIT3         IF 12 LINE DISPLAY PROCEDE
                MVI  M, 23D        ELSE SET LINES = 23
INIT3           LXI  H, 480D       NUMBER OF CHARACTERS (480D)
                JNC  INIT4         IF 12/40 SELECTED PROCEDE,
                DAD  H             ELSE DOUBLE ~~IT~~ CHARS.
INIT4           ORA  A
                JNZ  INIT5         IF 24/80 NOT SELECTED PROCEDE
                DAD  H             ELSE DOUBLE CHARS.
INIT5           SHLD NUMCHAR
                LXI  D, SCRNBEG    GET ADDRESS POSITION ONE
                DAD  D             ADD TO NUMBER OF CHARS
                SHLD OFFSCRN       = ADDRESS FIRST POSITION OFF SCREEN
                LDA  MODEFLAG      GET MODEBYTE
                ANI  0CH           MASK OFF TEXT MODE BITS
                XRI  0CH
                STA  TXTFLG        STORE IT
INIT6           XRA  A             ZERO ACCUMULATOR
                STA  ESCFLG        SAVE IN ESCFLAG
                RET                PULL RETURN OFF STACK
                                   (DSPEX NORMALLY)
                                   (DSPEX1 FROM ESCAPE)
```

CHAROUT

```
FNTE CHAROUT    PUSH H
                PUSH D
                PUSH B
                PUSH PSW
                STA  INPUT         SAVE CHARACTER INPUT TO ROUTINE
                LHLD CURPTR
                LDA  BLANK         BLANK CURSOR
                MOV  M, A
```

```
                LXI   H, INPUT      POINT TO BYTE INPUT
                MOV   A, M          GET IT
                CPI   1BH           ESCAPE ?
                JZ    ESCAPE        YES. GO PROCESS IT
                LDA   ESCFLG        IS AN ESCAPE SEQUENCE IN PROGRESS
                ORA   A
                JNZ   ESCAPE        YES. GO PROCESS.
                MOV   A, M          GET INPUT AGAIN
                CPI   7FH           RUBOUT ?
                JZ    RUBOUT        PROCESS WITH CONTROL ROUTINE
                LDA   TXTFLG        GET TEXT MODE
                ORA   A
                JZ    CHOUT         IF GRAPHICS MODE GO AND DISPLAY
                MOV   A, M          RESTORE INPUT
                CPI   FF            IF FF
                JZ    DSPEX         RETURN WITH DISPLAY UNCHANGED
                ANI   7F            STRIP PARITY BIT
                MOV   M, A          PUT IN INPUT
                SBI   20H           IF CONTROL CHARACTER
                JM    CONTROL       GO PROCESS
                LDA   MODEFLG       ELSE GET VIO MODE FLAG
                MOV   B, A          SAVE IT
                ANI   0CH           MASK OFF TEXT MODE BITS
                CPI   08            IF NOT ASCII
                JNZ   CHOUT         GO AND DISPLAY IT, ELSE
                MOV   A, B          GET MODE BYTE
                ANI   20H           MASK OFF CASE MODE BIT
                JNZ   CHOUT         IF LOWER CASE ON, GO DISPLAY
                MOV   A, M          ELSE GET INPUT
                SBI   61H           IF NOT LOWER CASE,
                JM    CHOUT         GO DISPLAY, ELSE
                SBI   1AH           CHECK IF ALPHABETIC.
                JP    CHOUT         NO, DISPLAY, ELSE
                ADI   5BH           TRANSLATE TO UPPER CASE
                MOV   M, A          AND STORE TRANSLATED BYTE IN INP
CHOUT           CALL  DISPLAY       PUT BYTE ON SCREEN
                CALL  NEWCUR        INCREMENT CURSOR COLUMN
DSPEX1          CALL  NEWPTR        GET NEW CURSOR POINTER
DSPEX           CALL  CURDISP       DISPLAY CURSOR AT NEW LOCATION
```

```
                POP   PSW        RESTORE ENVIRONMENT
                POP   B
                POP   D
                POP   H
F8EL            RET
```

NEWCUR

INCREMENTS CURSOR COLUMN POINTER AND ADJUSTS FOR LINE CHANGE ON SCROLLS AS NEEDED
ENTER: H,L = INDETERMINATE
EXIT: H,L POINTING TO CURLINE

```
F8E7  NEWCUR  CALL   NEWCOL      GET NEW COLUMN POSITION
              CZ     NEWLINE+3   IF LINE HAS CHANGED ADJUST
              LDA    PRTCTFL     ELSE CHECK FIELD PROTECTED FLAG
              XCHG               H,L = NEW CURSOR POINTER (FROM NEWCOL)
              ANA M              CHECK FOR REVERSED VIDEO AT NEW
              JM     NEWCUR      POSITION, IF YES GET NEXT POSITION
F8F5          RET                ELSE RETURN
```

NEWLINE

UPDATES CURSOR LINE POSITION
ENTER:
EXIT:

```
F8F6    NEWLINE  LXI  H, CURLIN   POINT TO CURSOR LINE NO.
                 INR  M           LINE → LINE + 1
                 LDA  LINES       GET LINE FORMAT
                 CMP  M           OFFSCREEN?
                 RP               NO. RETURN ELSE
                 DCR  M           LINE → BOTTOM LINE
                 LDA  MODEFLG     CHECK FOR WRAP AROUND MODE
                 ANI  &C          MASK OF WRAP BIT AND TEST MODE BIT
                 JM   WRAP        GO WRAP AROUND, ELSE
```

```
                CPI     ØC          GRAPHICS MODE?
                JNZ     SCROLL      NO, SCROLL
WRAP            XRA     A           ZERO ACCUMULATOR
                MOV     M,A         CURSOR LINE = 0
                INX     H           POINT TO CUR COL
                MOV     M,A         CURSOR COL = 0
                RET

SCROLL          LHLD    COLS        GET # OF COLUMNS
                PUSH    H           SAVE IT
                XCHG                # COLS → D,E
                LHLD    NUMCHAR     GET # OF CHARACTERS
                MOV     A,L
                SUB     E           SUBTRACT 1 LINE FROM # CHARS
                MOV     C,A         SAVE IN C
                MOV     B,H         B,C = # CHARS LESS 1 LINE
                XCHG                H,L → NUM COLS  DE → NUMCHARS
                LXI     D,SCRNBEG   D,E = POSITION 1
                DAD     D           H,L → POSITION 1 + 1 LINE = POSITION 1, LINE 2
                CALL    UPSCRL
F928            JMP     SCRL
```

CONTROL

REQUIRES EQUATE FOR CTLIMP = START OF CTL CHAR JUMP TABLE

ENTER: H,L POINTING TO ADDRESS OF STORED CONTROL CHARACTER (INPUT)

```
F929 CONTROL    XCHG                DE POINTS TO INPUT
                LHLD    CTLNK       LOAD LINKAGE ADDRESS
                MOV     A,H
                ORA     A           IF H1 BYTE = 0 ~~CHECK OWN TABLE~~
                LDAX    D           ~~ELSE~~ GET CTL-CHARACTER
                JZ      RUBOUT+3    AND CHECK OWN TABLE ELSE
                CALL    PARSE       RETURNS Z=1 IF NO MATCH
RUBOUT          JNZ     CTL2        IF MATCH DO IT, ELSE
```

(6)

```
                RUBOUT+3    LDA   INPUT       GET CTL-CHAR
                            LXI   H, CTLJMP   POINT TO INTERNAL TABLE
                            CALL  PARSE       PARSE IT
                            JNZ   DSPEX       EXIT IF NO MATCH, ELSE
                            LXI   D, DSPEX1   PREPARE FOR INDIRECT CALL ON
                            PUSH  D           RETURN
                            LXI   D, CURCOL   POINT TO CURCOL
F94C                        PCHL              GO AND PROCESS
```

ESCAPE

REQUIRES EQUATE FOR ESCJMP TO ESCAPE JMP TABLE TABLE TERMINATES WITH Ø

```
F94D            ESCAPE  LXI D, DSPEX1   SET UP INDIRECT CALL FOR
                        PUSH D          RETURN
                        XCHG            D,E = ADDRESS OF BYTE INPUT
                        LXI H, ESCFLG   POINT TO ESCAPE FLAG
                        MOV A, M        GET IT
                        INR M           BUMP IT ONE
                        ORA A
                        RZ              IF FLAG WAS Ø, RETURN FOR 2ND
                                        CHARACTER IN SEQUENCE
                        DCR A           ELSE RESTORE FLAG TO ORIGINAL VALUE
                        INX H           POINT ESCSTOR
                        LDAX D          GET BYTE FROM INPUT
                        JNZ ESC1        IF THIRD CHAR OR MORE, JUMP
                        MOV M, A        ELSE SAVE 2ND CHAR IN ESCSTOR
                ESC1    LHLD ESCLNK     GET ADDRESS OF EXTERNAL TABLE
                        MOV A, H        CHECK HI BYTE
                        ORA A           IF ZERO,
                        LDAX D            (RESTORE BYTE INPUT)
                        JZ  ESC2        GO CHECK INTERNAL TABLE
                        CALL PARSE      ELSE SEARCH EXTERNAL TABLE
                        JNZ ESC3        IF GO PROCESS
                ESC2    LDA ESCSTOR     ELSE GET COMMAND
                        ANI DF          CONVERT "=A" TO "1D"
                        LXI H, ESCJMP   ADDRESS OF INTERNAL TABLE
```

```
                        CALL   PARSE       CHECK INTERNAL TABLE, IF NO MATCH
                        JZ     INIT6       GO TO CLEAR ESCFLG WITH DSPEX1 ON STACK
                        LDA    MODEFLG     ELSE
                        LXI    D, INIT7    FOR INDIRECT JUMP
F983                    PCHL               GO TO APPROPRIATE ESCAPE ROUTINE.
```

VERTAB (CTL-K) Moves Cursor Up 1 Line

ENTER: DE Points To CURCOL DSPEX1 on STACK

```
F984    VERTAB  DCX    D           POINT TO CURLIN
        BCKSP   LDAX   D           GET CURRENT CURLIN
                ORA    A
                RZ                 RETURN IF ALREADY 0
                DCR    A           ELSE LINE -> LINE - 1
        TAB     STAX   D           PUT BACK
F98A            RET
```

CR Perform A Carriage Return

```
F98B    CR      XRA    A
                STA    INSRTF
F991            JMP    TAB
```

PFIELD Toggle Protected Field Flag

```
F992
        PFIELD  LXI    H, PRTCTFL  POINT TO PROTECTED FIELD FLAG
                JMP    TOGGLE
```

(8)

(CTL-T) INSRT TOGGLES CHARACTER INSERTION FLAG

```
F998            INSRT   LXI  H,INSRTF   POINT TO INSERTION FLAG
                TOGGLE  MOV  A,M        GET IT
                        CMA             FLIP IT
                        MOV  M,A        PUT IT BACK
F99E                    RET
```

ERASE (CTL-Z)

ERASES SCREEN, SKIPS PROTECTED FIELDS

```
F99E            ERASE   LHLD NUMCHAR
                        XCHG            D,E = NUMCHARS
                        LXI  H,SCRNBEG  GET POSITION 1 ADDRESS
                ERASE1  LDA  PRTCTFL    CHECK PROTECT FLAG
                        ANI  80H        MASK OF BIT 7
                        ANA  M          CHECK FOR REVERSE VIDEO
                        JM   ERASE2     YES, SKIP THIS ONE
                        MVI  M,20       ELSE BLANK IT
                ERASE2  INX  H          POINT TO NEXT
                        DCX  D          POSITIONS TO DO LESS ONE
                        MOV  A,D        DONE?
                        ORA  E
                        JNZ  ERASE1     NO, DO ANOTHER
                HOMECUR LXI  H,0000     ELSE CURLIN + CURCOL
                        SHLD CURLIN     SET TO 0
F9BE                    RET
```

CLEOF CLEAR TO END OF FIELD (↑U)

ENTER: B,C = # OF POSITIONS TO ERASE

```
FSBF  CLEOF    CALL   CNTPOS    GET POSITIONS TO ERASE IN C
      CLEOF1   LDA    PRTCTFL   GET FIELD PROTECT FLAG
               ORA    A
               JZ     CLEOF2    IF NOT ON, GO ERASE
               MOV    A,M       ELSE CHECK NEXT POSITION
               ORA    A
               JM     CLEOF3    IF VIDEO REVERSED SKIP
      CLEOF2   MVA    A,20H     ELSE ERASE WITH ASCII SPACE
               MOV    M,A
      CLEOF3   INX    H         POINT TO NEXT POSITION
               DCR    C         COUNT -1 COUNT-1
               JNZ    CLEOF1    NOT DONE DO ANOTHER
FSDG           RET               ELSE EXIT
```

RVID REVERSE CHARACTER VIDEO

```
FSDJ  RVID     LXI    H,RVFLG   POINT TO REVERSE VIDEO FLAG
FSDC           JMP    TOGGLE
```

RUBOUT

```
FSDD  RUBOUT   CALL   CNTPOS    GET POSITIONS TO EOF IN C
               LHLD   CURPTR     GET CURRENT CURSOR
               MOV    D,H
               MOV    E,L       STORE IN D,E
               INX    H         POINT TO NEXT
               CALL   UPSCRL    MOVE ALL CHARS LEFT 1 POS.
               MVI    A,20H     GET ASCII BLANK
```

```
                    DCX   D          POINT TO LAST POSITION
                    STAX  D          ERASE IT
FSED                RET              EXIT


                    CNTPOS

FREE    CNTPOS      LDA   PRTCTFL    GET FIELD PROTECT FLAG
                    ANI   80H        STRIP OFF LOW BITS
                    MOV   D,A        SAVE IT
                    LHLD  CURPTR     GET CURRENT CURSOR ADDRESS
                    PUSH  H          SAVE IT
                    LDA   CURCOL     GET CURRENT CURSOR COLUMN
                    MOV   E,A        SAVE IT
                    LXI   B,0000     # OF POSITIONS TO END OF FIELD
        CNTPOS1     LDA   COLS       GET NUMBER OF COLUMNS
                    SUB   E          A= NUMBER OF COLUMS TO END OF LINE
                    INR   E          EF CURRENT COLUMN +1
                    INX   H          CURPTR=CURPTR+1
                    INX   B          POSITIONS → POSITIONS+1
                    DCR   A          COLUMNS REMAINING → COLS.REM-1
                    JZ    CNTPOS2    IF DONE EXIT
                    MOV   A,M        ELSE CHECK IF PROTECTED
                    ANA   D          IF NOT CHECK NEXT
                    JP    CNTPOS1    ELSE EXIT
        CNTPOS2     POP   H          RESTOR CURSOR POINTER
FAID                RET              TO H,L
```

PARSE

RETURNS WITH Z=1 IF NO MATCH ; H,L = ADDRESS FROM TABLE ON MATCH
ENTER WITH H,L = ADDRESS OF TABLE

```
FA11    PARSE     MOV   B,A      SAVE COMMAND IN B
        PARSE1    MOV   A,M      GET COMMAND FROM TABLE
                  LXI   D,CURLIN
                  ORA   A
                  RZ             RETURN IF END OF TABLE
                  CMP   B        ELSE CHECK FOR MATCH
                  JNZ   PARSE2   IF NOT MOVE TO NEXT
                  INX   H        IF YES POINT TO ADDRESS
                  MOV   E,M      PUT LO BYTE IN E
                  INX   H        POINT TO HI BYTE
                  MOV   D,M      PUT HI BYTE IN D
                  XCHG           ADDRESS TO H,L
                  ORA   A        SET Z STATUS TO Ø
                  RET
        PARSE2    INX   H
                  INX   H
                  INX   H        POINT TO NEXT ENTRY
FA28              JMP   PARSE1   GO CHECK IT
```

DELIN

DELETES LINE AT CURRENT CURSOR POSITION

```
FA29    DELIN     CALL  LINADR   B,C=# POSITIONS TO MOVE  D,E=POSITION ONE
                                 CURRENT LINE  H,L = # COLS IN DISPLAY
                  PUSH  H        SAVE # COLS
                  DAD   D        H,L → POS. ONE CURRENT LIN + 1 LINE
                  CALL  UPSCRL   MOVE LINES UP 1 ONE LINE
```

SCRL

```
FA31    SCRL    POP   B       GET # COLS IN C
                XCHG          DE => LAST SOURCE+1   H,L = LAST DEST +1
                JMP   SCRL2
```

INSRTLN LINE INSERTION ENTRY (TE)

```
FA36    INSRTLN CALL  LINADR
                PUSH  H       SAVE # COLS
                DAD   D       H,L POINTS TO CURRENT LINE + 1 LINE
                DAD   B       H,L POINT TO LAST POSITION IN DISPLAY +1 POS
                XCHG          DE POINTS TO ABOVE
                DAD   B       H,L POINTS TO LAST POSITION IN DISPLAY +1
                                POSITION - 1 LINE
                DCX   H       H,L GOES TO END LAST LINE - 1 LINE
                DCX   D       D,E => END LAST LINE
                CALL  DWNSCRL MOVE THEM
                POP   B       RESTORE # OF COLS TO B, C
                INX   H       H,L => POSITION ONE OF LINE TO INSERT
```

SCRL1 BLANKS C CHARACTERS BEGINNING AT H,L MOVING UP

```
FA45    SCRL2   MVI   M,20H   GET ASCII BLANK TO SCREEN
                INX   H       POINT TO NEXT
                DCR   C       # COLS -1
                JNZ   SCRL1   NOT DONE DO MORE
FA4C            RET           ELSE EXIT.
```

LINADR

ENTER: DE POINTING TO CURCOL RETURN: B,C = # SCREEN POSITIONS FROM CURRENT LINE PLUS ONE (COL Ø) TO SCREEN END; D,E = ADDRESS OF CURRENT LINE (COL Ø); H,L = # COLS IN DISPLAY

```
FA4D    LINADR    XRA    A        ZERO ACCUMULATOR
                  STAX   D        STORE IN CURCOL
                  CALL   CNTEND   GET POSITIONS TO SCREEN END
                  XCHG            PUT IN D.E
                  LHLD   COLS     GET COLUMNS IN DISPLAY
                  MOV    A,L
          LINADR1 DCX    B        POSITIONS LEFT → P. LEFT - 1
                  DCR    A        COLS → COLS - 1 (SUB ONE LINE FM
                  JNZ    LINADR1  BC, IF NOT DONE DO ANOTHER
FA5C              RET             THEN EXIT
```

CNTEND

ENTER; UPDATED CURLIN + CURCOL EXIT; BC = #POS. FROM NEW CURSOR POSITION TO END OF SCREEN, NEW CURPTR IN H,L

```
FA5D    CNTEND    CALL   NEWPTR   UPDATE CURPTR
                  PUSH   H        SAVE IT ON STACK
                  XCHG            AND IN DE
                  LHLD   OFFSCRN  GET FIRST ADDRESS OFF
                  MOV    A,D
                  CMA
                  MOV    D,A
                  MOV    A,E
                  CMA
                  MOV    E,A
                  INX    D        FORM TWO'S COMPLEMENT OF
                                  NEW CURSOR ADDRESS
```

```
                DAD   D     SUBTRACT CURSOR ADDRESS FROM SCREEN END
                PUSH  H     PUT IT ON STACK
                POP   B     BRING TO BC
                POP   H     NEW CURSOR ADDRESS TO HL
FA70            RET
```

VIO MODE ADJUST ROUTINES

CONVERTS CURRENT VIO MODE BYTE TO NEW MODE BYTE

ENTER: DE = INIT7 A = CURRENT VIO MODE BYTE

```
FA71  ESCE    ANI  F3H (1111 0011)   TURN OFF TEXT MODE BITS
              ORI  04  (0000 0100)   TURN ON EXTENDED TEXT BIT
              XCHG
              PCHL                   INDIRECT CALL TO INIT7

      ESCG    ORI  0CH (0000 1100)   TURN ON BOTH TEXT BITS
              XCHG                   AS ABOVE AND ALL FOLLOWING
              PCHL

      ESCT    ANI  F3H (1111 0011)   TURN OFF TEXT BITS
              ORI  08H (0000 1000)   TURN ON ASCII BIT
              XCHG
              PCHL

      ESCS    XRI  80H (10000000)    TOGGLE BIT 7 (SCROLL/WRAP)
              XCHG
              PCHL
```

```
ESCU    XRI 20H (00100000)   TOGGLE UPPER/LOWER CASE BIT
        XCHG
        PCHL

ESCV    XRI 10H (00010000)   TOGGLE VIDEO BIT
        XCHG
        PCHL

ESCL    XRI 02 (00000010)    TOGGLE LINES BIT
        XCHG
        PCHL

ESCC    XRI 01 (00000001)    TOGGLE COLUMNS BIT
        XCHG
FA94    PCHL
```

FA95

CURDISP

DISPLAYS CURSOR AT CURRENT CURPTR BY REVERSING VIDEO AT THAT LOCATION OR IF IN GRAPHICS MODE SENDS CODE 7FH

```
CURDISP LHLD  CURPTR   GET CURSOR ADDRESS
        MOV   A,M      GET EXISTING CHARACTER
        STA   BLANK    SAVE IT
        ORI   80H      TURN ON REVERSE VIDEO BIT
        MOV   M,A      PUT ON SCREEN
        LDA   TXTFLG   CHECK IF IN GRAPHICS MODE
        ORA   A
        RNZ            IF NOT RETURN
        MOV   M,A      ELSE USE 7FH FOR CURSOR
FAAC    RET
```

ESCAI (ESC/CTL-I)

```
FAA7  ESCAI     LXI  H,TAB
                MVI  B,0AH
      ESCAIA    XRA  A
                MOV  M,A
                INX  H
                DCR  B
                JNZ  ESCAIA
                JMP  INIT6
```

ESCI (ESC-I)

```
FABC  ESCI      CALL TABS
                XRI  80H
      ESCIA     RRC
                DCR  B
                JNZ  ESCIA
                STAX D
                JMP  INIT6
```

TABS

```
FAC4  TABS      LDA     CURCOL
                MOV     H,A
                INR     H
                LXI     D,TAB
      TABS1     MVI     C,08
      TABS2     DCR     H
                JZ      TABS3
                DCR     C
                JNZ     TABS2
                INX     D
                JMP     TABS1
      TABS3     LDAX    D
                MOV     B,C
                RLC
      TABS4     DCR     C
                JNZ     TABS4
FAE1            RET
```

MTAB

```
FFE2
      MTAB      XRA     A
                STA     TABFLG
      MTAB1     CALL    NEWCOL
                JNZ     MTAB
                INR     M
                CMP     M
                JM      WRAP
      MTAB2     LDA     PRTCTFL
```

```
                XCHG
                ANA  M
                MOV  A,M
                LXI  D, TABFLG
                JP   MTAB3
                STAX D
                JMP  MTAB1
MTAB3           LDAX D
                ORA  A
                RM
                CALL TABS
                ORA  A
                RM
FB0B            JMP  MTAB1
```

NEWPTR

TAKES CURSOR LIN + COL (CURLIN + CURCOL) AND PRODUCES NEW CURPTR TAKING INTO ACCOUNT SCREEN FORMAT

FB0C

```
NEWPTR      LHLD   COLS       GET COLUMN FORMAT
            XCHG              SAVE IN E
            LHLD   CURLIN     H = CURCOL   L = CURLIN
            MOV    C,H        CURCOL TO C
            MOV    B,L        CURCOL TO B
            LXI    H,SCRNBEG  SCRN POSITION
            INR    B          LINE → LINE+1
NEWPTR1     DCR    B          LINE → LINE-1
            JZ     NEWPTR2    (FOR AS MANY TIMES AS CURLIN,
            DAD    D          ADD # COLS TO SCREEN POSIT
```

```
                JMP   NEWPTR1
NEWPTR2         DAD   B           THEN ADD CURCOL
                SHLD  CURPTR      SAVE IN CURPTR
FB25            RET
```

NEWCOL

INCREMENTS CURSOR COLUMN. TAKE COLUMN FORMAT INTO ACCOUNT
RETURN: Z = 1 IF LINE HAS CHANGED HL = POINTING TO CURLIN

```
FB26  NEWCOL    LHLD   CURPTR     GET CURSOR ADDRESS
                INX    H          BUMP IT
                SHLD   CURPTR     SAVE
                XCHG
                LXI    H, CURCOL  POINT TO CURCOL
                INR    M          BUMP IT
                LDA    COLS       GET COLUMN FORMAT
                SUB    M          SUBTRACT CURCOL
                RNZ               RETURN IF NOT BEYOND LINE E
                MOV    M, A       ELSE Ø CURCOL
                DCX    H          POINT TO CURLIN
                LDA    LINES      GET LINE FORMAT
FB3C            RET               EXIT
```

CURADJ

ADJUST CURSOR TO ANY POINT ON DISPLAY

CALLED BY ESC/=/Y,X

```
FB3D    CURADJ      LXI   H, ESCFLG     POINT TO ESCAPE SEQUENCE FLAG
                    LXI   D, CURLIN     POINT TO CURSOR LINE
                    LDA   INPUT         GET COMMAND
                    SUI   20H           CONVERT TO * NOTE 1
                    MOV   B,A           SAVE IN B
                    MOV   A,M           GET ESCAPE SEQUENCE FLAG
                    SUI   03            RETURN FOR NEXT CHAR IN SEQUENCE
                    RM                  TILL CHAR #3
                    JNZ   CURADJ2       ON CHAR #4 JUMP
                    LDA   LINES         ELSE GET LINE FORMAT
CURADJ1             STAX  D             LET CURLIN = LAST LINE
                    CMP   B             IF LINE INPUT GREATER THAN
                    RM                  LINES IN FORMAT RETURN
                    MOV   A,B           ELSE LET CURLIN
                    STAX  D             = LINE INPUT
                    RET
CURADJ2             MVI   M,0           LET ESCFLG =0 TERMINATE SEQ.
                    LDA   COLS          GET COLUMN FORMAT
                    INX   D             POINT TO CURCOL
                    DCR   A             MAKE COL= LAST COLUMN IN LINE
FB6A                JMP   CURADJ1       * SEE NOTE 2
```

*1 LINE AND COLUMN CHOSEN ARE INPUT +20H TO BYPASS CTL-CHARACTER FILTERS

*2 ON PASS THRU CURADJ1 FOR COLUMNS SUBSTITUTE COL FOR LINE IN COMMENTS

DISPLAY

PLACES CHARACTER WAITING IN INPUT ON SCREEN. REVERSES VIDEO IF RVFLG IS ON

```
FB63  DISPLAY   LHLD    CURPTR    GET CURSOR ADDRESS
                PUSH    H         SAVE IT
                LDA     INSRTF    CHECK IF CHARACTER INSERT ON
                ORA     A
                JZ      DISP      IF NO DISPLAY
                CALL    CNTPOS    B,C = POSITIONS TO END OF LINE
                DCX     B         DONT COUNT CURSOR
                DAD     B         ADD TO CURSOR ADDRESS
                MOV     D,H       HL = END OF LINE
                MOV     E,L       DE = END OF LINE
                DCX     D         DE = END LINE - 1
                XCHG              H,L = END LINE - 1  DE = END LINE
                CALL    DWNSCRL   OPEN A HOLE FOR INSERTION
DISP            POP     H         RESTORE CURSOR ADDRESS
                LDA     RVFLG     HAS CHAR. REVERSE VIDEO BEEN SET?
                ANI     80H       STRIP OF LOW ORDER BITS
                MOV     B,A       SAVE
                LDA     INPUT     GET BITE INPUT
                ORA     B         SET BIT 7 AS NECESSARY
                MOV     M,A       PLACE RESULT ON SCREEN
FB8C            RET               EXIT
```

DWNSCRL

ENTER: SOURCE END IN H,L DESTINATION END IN DE
POSITIONS TO MOVE IN B,C.

```
FB87  DWNSCRL   MOV   A,C     ALL POSITIONS DONE?
                ORA   B
                RZ            YES. RETURN
                MOV   A,M     GET SOURCE
                STAX  D       PUT IN DESTINATION
                DCX   H       SOURCE -> SOURCE -1
                DCX   D       DEST -> DEST -1
                DCX   B       # BYTES -> BYTES -1
FB91            JMP   DWNSCRL GO BACK FOR MORE
```

[FB92 TO FBE1 JUMP TABLES]

```
CTLJMP    DB    0DH
          DW    CR
          DB    0AH
          DW    NEWLINE
          DB    0BH
          DW    VERTAB
          DB    0CH
          DW    NEWCUR
          DB    08H
          DW    BCKSP
          DB    1EH
          DW    HOMECUR
```

```
        DB      1AH
        DW      ERASE
        DB      15H
        DW      CLEOF
        DB      16H
        DW      RVID
        DB      09H
        DW      MTAB
        DB      7FH
        DW      RUBOUT
        DB      14H
        DW      INSRT
        DB      04
        DW      DELIN
        DB      05
        DW      INSRTLN
        DB      10H
        DW      PFIELD
        DB      1DH
        DW      ESCJMP
        DB      0
```

?

```
ESCJMP  DB      1DH
        DW      CURADJ
        DB      'I'
        DW      ESCI
        DB      09H
        DW      ESCTAB
        DB      'T'
        DW      ESCT
        DB      'E'
        DW      ESCE
```

```
        DB      'G'
        DW      ESCG
        DB      'S'
        DW      ESCS
        DB      'U'
        DW      ESCU
        DB      'V'
        DW      ESCV
        DB      'L'
        DW      ESCL
        DB      'C'
        DW      ESCC
        DB      0
```

INITMON

REQUIRES EQUATE FOR STACK TO ESTABLISH STACKPOINTER

```
FBE2    INITMON     LXI     SP, STCK        SET STACK
                    MVI     A, 0AAH
                    OUT     03
                    CMA
                    OUT     03
                    CMA
                    OUT     03
                    MVI     A, 27H          USART MODE BYTE
                    OUT     03              RESET USART
```

```
                CALL    INITVIO
                LXI     H, SGNON        POINT TO SIGN ON MSG
                CALL    MSGOUT          DISPLAY IT
IMON1           LXI     SP, STCK        RESET STACK
                CALL    CRLF            DO CRLF
                MVI     A, '?'          GET PROMPT
                CALL    CHAROUT         DISPLAY IT
                CALL    CHARIN          INPUT WITH ECHO
                MOV     B, A            SAVE IT
                LXI     D, IMON1        SET DEFAULT RETURN
                PUSH    D               ADDRESS
                LHLD    COMLNK          JUMP TABLE ADDRESS
                MOV     A, H            -GET HI BYTE
                ORA     A
                MOV     A, B            RESTORE INPUT
                JZ      IMON2           IF HI BYTE = 0 CHECK INTERNAL TABLE
                CALL    PARSE           ELSE LOOK
                JNZ     IMON3           IF MATCH DO IT
IMON2           MOV     A, B            ELSE RESTORE INPUT
                LXI     H, COMJMP       POINT TO INTERNAL TABLE
                CALL    PARSE           LOOK
                RZ                      GO TO IMON1 ON NO MATCH
                MVI     B, 01           ELSE,
FC28    IMON3   PCHL                    DO IT.
```

[FC29 - FC45 SIGN ON]

```
SGNON     DB      'INSERT SIGN ON MESSAGE', 0
```

JUMP + CALL

FC46

```
          JUMP      POP   D        POP DEFAULT RETURN OFF STACK
FC4A      CALL      CALL  ADRIN    GET ADDRESS IN H,L
                    PCHL           GO
```

EXAM

EXAMINES AND MODIFIES MEMORY

```
FC4B      EXAM      CALL  ADRIN    GET STARTING ADDRESS
                    CALL  CRLF     DO CR + LF
                    CALL  APROUT   PRINT H,L
                    MOV   A,M      GET BYTE AT HL
                    MOV   E,A      SAVE
                    CALL  OHEX     PRINT ACCUM AS TWO HEX DIGIT
                    XCHG           SAVE HL IN DE
                    CALL  ADRIN1   GET NEW BYTE IN L
                    XCHG           L UNCHANGED IF INPUT NON HEX
```

```
        MOV     M,E         STORE NEW INPUT
        DCX     H           POINT TO PREVIOUS ADDRESS
        CPI     0A          WAS LAST CHARACTER A LF?
        RZ                  IF YES RETURN
        CPI     2D          ELSE WAS IT A '-'
        JZ      EXAM+3      IF YES DISPLAY PREVIOUS
        INX     H           ELSE, POINT TO NEXT
        INX     H
FC6C    JMP     EXAM+3
```

DMEM

DISPLAY MEMORY IN BLOCK FROM START TO END

```
FC6D    DMEM    CALL    BLKIN       GET PARAMETERS OF BLOCK
                LDA     MODEFLG     GET VIO MODE
                RRC
                RRC                 CHECK VIO LINE MODE
                MVI     D,0CH       SET FOR 12 LINE DISPLAY
                JC      DMEM1       IF NOT 24 LINES GO AHEAD
                MVI     D,18        ELSE SET FOR 24 LINE DISPLAY
        DMEM1   CALL    CRLF        CR+LF
                IN      03          READ INPUT STATUS
                ANI     02
                RNZ                 RETURN ON ANY INPUT
                MVI     E,08        SET FOR 8 COLUMNS
                LDA     MODEFLG     GET MODE FLAG
                RRC                 CHECK COLUMN MODE
```

```
                JC      DMEM2       IF NOT 80 COLUMN GO AHEAD
                MVI     E,10H       ELSE USE 16 COLUMS
DMEM2           CALL    ADROUT      DISPLAY H,L
                MOV     A,M         GET CONTENTS AT H,L
                CALL    OHEX        DISPLAY AS 2 HEX DIGITS+SP
                INX     H           POINT TO NEXT
                DCX     B           BYTES -> BYTES-1
                MOV     A,B
                ORA     C           IF DONE,
                RZ                  RETURN
                DCR     E           ELSE COLUMNS->COLUMNS-1
                JNZ     DMEM2+3     DO ANOTHER COLUMN
                DCR     D           LINES -> LINES-1, IF NOT DONE
                JNZ     DMEM1       DO ANOTHER LINE
                CALL    CHARIN      ELSE GET INPUT AND
FLAG            JMP     DMEM+3      GO BACK FOR MORE
```

FLAG

OHEX

DISPLAYS CONTENTS OF A AS TWO HEX DIGITS + SPACE

```
FCAD    OHEX        CALL    OUTHEX
                    MVI     A,20H       GET ASCII BLANK
                    CALL    CHAROUT     DISPLAY IT
FCB5                RET                 EXIT
```

ADROUT

DISPLAY CONTENTS OF HL AS 4 HEX DIGITS + SPACE

```
FCBC    ADROUT      MOV     A,H         GET HL BYTE
                    CALL    OUTHEX      PUT IT OUT
                    MOV     A,L         GET LO BYTE
                    CALL    OHEX        PUT IT OUT
FCBE                RET                 EXIT
```

PROTECT

PROTECT RAM 4-A MEMORY

```
PROTECT             INR     B
UNPROTECT           CALL    ADRIN2
                    MOV     A,D
                    ANI     FC
                    ORA     B
                    MOV     D,A
                    MOV     A,H
```

```
                ANI   FC
                ORA   B
PROTECT1        OUT   FE
                CMP   D
                RZ
                ADI   04
FD04            JMP   PROTECT1
```

LD TAPE

LOAD INTEL HEX TAPE

```
FD05    LDTAPE      CALL   CHARIN
                    SBI    3A H
                    JNZ    LDTAPE
                    MOV    D,A
                    CALL   FD04
                    ORA    A
                    RZ
                    MOV    B,A
                    CALL   FD04
                    MOV    H,A
                    CALL   FD04
                    MOV    L,A
                    CALL   FD04
LDTAPE1             CALL   FD04
                    MOV    M,A
                    INX    H
                    DCR    B
                    JNZ    LDTAPE1
```

```
        CALL    FDØY
        JZ      LOTAPE
        MVI     A,43
        CALL    CHAROUT
FD03    RET
```

FD04

```
FD04    CALL    CHARIN
        CALL    AHEX
        JC      FEØ7
        ADD     A
        ADD     A
        ADD     A
        ADD     A
        MOV     E,A
        CALL    CHARIN
        CALL    AHEX
        JC      FEØ7
        ADD     E
        MOV     E,A
        ADD     D
        MOV     D,A
FD1F    RET
```

BLKIN

```
FDLD    BLKIN     CALL    ADRIN2      GET START AND END OF BLOCK
                  PUSH    PSW         SAVE LAST CHAR
                  MOV     A,E         SUBTRACT END ADDRESS
                  SUB     L           FROM START, PUT
                  MOV     C,A         RESULT IN BC
                  MOV     A,D
                  SBB     H
                  MOV     B,A
                  INX     B           ADD ONE TO GET TRUE LENGTH
                  POP     PSW         RESTORE LAST INPUT
FDRC              RET
```

MOVE

```
FD2D    MOVE      CALL    ADRIN3      GET START(HL), DEST(DE), LENGTH(BC)
                  CALL    UPSCRL      MOVE BLOCK
                  RET
```

FILL

```
FD3A  FILL    CALL   ADRIN3    GET START(H,L), LENGTH(BC), VALUE(DE)
              MOV    A,E       PLACE VALUE IN A
              MOV    M,A       MOVE FIRST BYTE
              DCX    B         BYTES -> BYTES-1
              MOV    D,H       H,L = SOURCE
              MOV    E,L
              INX    D         DE = DESTINATION
              JMP    MOVE+3
FD3F
```

TEST

```
TEST    CALL   BLKIN     GET START, FINISH, LENGTH
        DCX    B         LENGTH -> LENGTH-1
TEST1   XRA    A         ZERO ACCUMULATOR
        MOV    D,M       SAVE EXISTING CONTENTS
TEST2   MOV    M,A       INSERT TEST VALUE
        CMP    M         READ IT BACK
        JNZ    TEST3     IF ERROR PRINT IT OUT
        DCR    A         ELSE CHECK NEXT VALUE
        JNZ    TEST2     IF NOT DONE TEST
        MOV    M,D       ELSE RESTORE CONTENTS
```

```
                IN   03          CHECK INPUT STATUS
                ANI  02
                RNZ              TERMINATE ON ANY INPUT
                INX  H           POINT TO NEXT BYTE
                DCX  B           BYTES -> BYTES-1
                MOV  A,B
                ORA  C           DONE?
                JNZ  TEST1       NO  DO NEXT
                RET              YES EXIT

TEST3           INX  H           POINT TO NEXT
                MOV  E,A         SAVE A CONTENTS
                CALL DISPADR     DISPLAY HL + ACTUAL VALUE
                MOV  A,E         RESTORE TEST VALUE
FD65            JMP  DISPA1      DISPLAY TEST VALUE
```

IN

```
IN              DCR  B
OUT             CALL ADRIN2      GET PORT AND VALUE
                MOV  A,B
                RLC
                RLC
                RLC
```

```
        XRI   08
        ORI   D3H
        MOV   D,L
        LHLD  IO
        MOV   M,A
        CMP   M
        RNZ
        PUSH  H
        INX   H
        MOV   M,D
        INX   H
        MVI   M,C9
        LXI   H,FD87
        XTHL
        MOV   A,B
        ORA   A
        MOV   A,E
        PCHL
        RNZ
        CALL  OHEX
FD8B    RET
```

READ

```
FD8C    READ    CALL    ADRIN
                SHLD    IO
                RET
```

VERIFY

```
FD93    VERIFY    CALL    ADRIN3    GET START(HL), DEST(DE), LENGTH(BC)
        VER1      LDAX    D         GET SOURCE BITE
                  CMP     M         COMPARE DEST. BYTE
                  INX     H         NEXT DEST
                  INX     D         NEXT SOURCE
                  JZ      VER2      IF SAME PROCEDE ELSE
                  CALL    DISPADR   PRINT ADDRESS + CONTENTS
                  XCHG              DEST TO H,L
                  CALL    DISPADR+3 PRINT DEST ADR+CONTENT
                  XCHG              RESTORE
        VER2      DCX     B         BYTES -> BYTES-1
                  MOV     A,B       DONE?
                  ORA     C
                  RZ                YES. EXIT, ELSE
                  IN      03        CHECK FOR ANY INPUT
                  ANI     02        IF INPUT,
                  RNZ               EXIT, ELSE
                  JMP     VER1      CONTINUE
```

SEARCH

```
FDB1    SEARCH    CALL    ADRIN3    GET START(HL), MODEL(DE), LENGTH(BC)
                  PUSH    H         SAVE START
```

```
        LXI   H,0FFFFH      SET DEFAULT MASK
        CPI   0A            IF LAST INPUT A CR(LF)
        CNZ   ADRIN         GET MASK, ELSE
        XTHL                PUT MASK ON STACK, SOURCE→HL
SRCH1   MOV   A,M           GET SOURCE BYTE
        XTHL                GET MASK
        ANA   H             MASK IT
        CMP   D             COMPARE WITH MODEL
        XTHL                SAVE MASK
        INX   H             POINT TO NEXT
        JNZ   SRCH2         IF NO MATCH, JUMP
        MOV   A,M           ELSE GET NEXT BYTE
        XTHL                GET MASK
        ANA   L             MASK IT
        CMP   E             COMPARE WITH MODEL
        XTHL                GET SOURCE ADDRESS
        CZ    DISPA         DISPLAY ADDRESS AND CONTEN
SRCH2   DCX   B             BYTES→BYTES-1
        MOV   A,B
        ORA   C             IF NOT DONE
        JNZ   SRCH1         DO NEXT ELSE
        POP   B             CLEAR STACK
FDDC    RET                 AND EXIT
```

ADRIN3

INPUTS 3 ADDRESSES

EXIT: FIRST ADDRESS IN HL, SECOND ADDRESS USED TO COMPUTE STRING LENGTH, IN BC, THIRD TO DE

```
ADRIN3  CALL  BLKIN
        JMP   ADRIN21
```

UPSCRL

ENTER: HL = SOURCE BC = # OF BYTES DE = DEST

```
F000  UPSCRL    MOV   A,B
                ORA   C          IF DONE
                RZ               EXIT, ELSE
                MOV   A,M        GET SOURCE BYTE
                STAX  D          MOVE TO DEST.
                DCX   B          BYTES -> BYTES-1
                INX   H          SOURCE -> SOURCE+1
                INX   D          DEST -> DEST+1
F00E            JMP   UPSCRL
```

CLOAD

CLOADX = LOAD AND EXECUTE

```
F00E  CLOADX    DCR   B
      CLOAD     PUSH  B
                CALL  ADRIN2
                MOV   A,D
                ORA   E
                JZ    CLOAD1
                PUSH  H
                PUSH  D
                CALL  ADRIN
                PUSH  H
                CALL  ACLOAD
                JNZ   CLOAD_3
```

```
                CALL    ACLOAD
CLOAD2          MVI     A,54H
                JZ      CLOAD3
CLOAD8          CALL    CHAROUT
                JMP     IMON1
CLOAD3          POP     B
                POP     H
                POP     D
                PUSH    B
                PUSH    H
                PUSH    D
                MOV     A,L
                SUB     E
                MOV     A,H
                SBB     D
                MOV     H,A
                DAD     H
                MOV     C,H
                INR     C
CLOAD4          CALL    BCLOAD
                CPI     81H
                JNZ     CLOAD2
                CALL    BCLOAD
                MOV     B,A
                LXI     H,0000
CLOAD5          CALL    CCLOAD
                STAX    D
                INX     D
                DCR     B
                JNZ     CLOAD5
                PUSH    D
                XCHG
```

```
                CALL    DCLOAD
                MOV     H,L
                MOV     L,A
                DAD     D
                MOV     A,H
                ORA     L
                MVI     A,43H
                CNZ     CHAROUT
                JNZ     CLOAD6
                MVI     A,2AH
                CALL    CHAROUT
CLOAD6          POP     D
                DCR     C
                JNZ     CLOAD4
                CALL    CRLF
                MVI     C,03
CLOAD7          POP     H
                CALL    ADROUT
                DCR     C
                JNZ     CLOAD7
                POP     PSW
                RAR
                RC
FE65            PCHL
```

ACLOAD

```
FE66  ACLOAD    CALL    ECLOAD
                CALL    BCLOAD
                CPI     01
                RNZ
```

```
                CALL  BCLOAD
                MVI   C, 5
ACLOAD1         CALL  BCLOAD
                CALL  CHAROUT
                DCR   C
                JNZ   ACLOAD1
                MVI   C, 3
ACLOAD2         CALL  DCLOAD
                XTHL
                PUSH  H
                DCR   C
                JNZ   ACLOAD2
                CALL  DCLOAD
                XRA   A
FE8D            RET
```

ECLOAD

```
FE8E  ECLOAD    CALL  FCLOAD
                MVI   B, 1FH
ECLOAD1         CALL  BCLOAD
                CPI   0E6H
                MVI   A, 49H
                JNZ   CLOAD8
                DCR   B
                JNZ   ECLOAD1
FEA1            RET
```

OSYNC

```
FEA2  OSYNC    PUSH  PSW
               IN    03
               ANI   04
               JZ    OSYNC+1
               POP   PSW
               OUT   00
FEA9           RET
```

GSYNC

```
FEAE  GSYNC    CALL  ADRIN
               MVI   A,10H
               OUT   03
      GSYNC1   MVI   A,E6
               CALL  OSYNC
               IN    03
               ANI   02
               RNZ
FEC1           JMP   GSYNC1
```

BCLOAD

```
FEC2  BCLOAD   IN    03
               RRC
```

```
        RRC
        JC      IMON1
        RRC
        JNC     BCLOAD
        IN      Ø
FECF    RET
```

CCLOAD

```
FED0  CCLOAD    CALL    BCLOAD
                PUSH    B
                MOV     C,A
                MVI     B,Ø
                DAD     B
                POP     B
FED9            RET
```

ALIGN

```
FEDA  ALIGN     CALL    CHARIN
                CALL    FCLOAD
        ALIGN1  LXI     H,SCRNBEG
                LXI     D,1E1H
        ALIGN2  DCX     D
                MOV     A,D
                ORA     E
                JZ      ALIGN1
```

CRLF

```
FF50    CRLF    MVI   A, 0DH
                CALL  CHAROUT
                MVI   A, 0AH
                CALL  CHAROUT
FF5A            RET
```

ADRIN

```
        ADRIN   LXI   H, 0000     ZERO  HL
        ADRIN1  CALL  CHARIN      INPUT BYTE WITH ECHO
                PUSH  PSW         SAVE IT
                CALL  AHEX        CONVERT TO HEX
                JNC   ADRIN2      OR IF NOT A VALD
                POP   PSW         HEX DIGIT, RESTORE
                RET               INPUT AND RETURN
        ADRIN2  DAD   H           ELSE
                DAD   H
                DAD   H
                DAD   H           SHIFT LEFT 4 BITS
                ADD   L
                MOV   L, A        ADD INPUT TO H,L
                POP   PSW         RESTORE
FF73            JMP   ADRIN1
```

AHEX

```
FF74  AHEX    SUI   30H     REMOVE ASCII BIAS
              RC            RETURN IF LESS THAN 0 DEC.
              CPI   0AH     IF 9 DEC OR LESS
              JC    AHEX1   GO PROCESS, ELSE
              SUI   11H     TEST FOR LESS THAN 'A' ASCII
              RC            IF YES RETURN. ELSE
              ADI   0AH     ADD CONVERSION FACTOR
              CPI   10H     CHECK FOR >0FH
      AHEX1   CMC           COMPLEMENT RESULT OF TEST
FF84          RET           EXIT
```

DISPA

```
FF85  DISPA   CALL  DISPADR   CALL + HL OUT + CONTENTS HL O
              MOV   A,M       GET NEXT BYTE
              CALL  OUTHEX    DISPLAY IT
FF8C          RET             EXIT
```

DISPADR

ENTER: HL POINTING TO BYTE FOLLOWING BYTE FOR DISPLAY

```
DISPADR   CALL  CRLF      CRLF
          DCX   H         POINT TO PREVIOUS
          CALL  ADROUT    DISPLAY HL
          MOV   A,M       GET CONTENTS HL
```

```
                CALL   OHEX        DISPLAY IT
                INX    H           RESTORE HL
FF99            RET
```

OUTHEX

DISPLAYS A BYTE AS TWO HEX DIGITS

```
FF9A  OUTHEX    PUSH   PSW
                RRC
                RRC
                RRC
                RRC                MOV HI DIGIT TO BITS 0-3
                CALL   ASCII       CONVERT TO ASCII & DISPLAY
                POP    PSW         RESTORE LOW DIGIT
                CALL   ASCII       DISPLAY IT
FFA6            RET
```

ASCII

CONVERTS BITS 0-3 TO ASCII AND DISPLAYS

```
FFA7  ASCII     ANI    0FH         STRIP HI BITS
                ADI    90H
                DAA
                ACI    40H
                DAA                CONVERT TO ASCII
FFB2            RET
```

MSGOUT

SENDS THE STRING BEGINNING AT H,L TO CHAROUT. STOPS AT A Ø BYTE

```
FF83  MSGOUT-3    CALL  CRLF      CRLF
      MSGOUT      MOV   A,M       GET BYTE
                  ORA   A
                  RZ              TERMINATE IF ZERO, ELSE
                  CALL  CHAROUT   DISPLAY IT
                  INX   H         POINT TO NEXT
FFBF              JMP   MSGOUT
```

COMJMP

REQUIRES EQUATE TO COMJMP (FFC4 IN SMP 80.0)