```
;
; Test Program to interact with LAVA-10 VGA Board.. John Monahan
;===============================================================================
;
;      V0.1   4/27/2012           ;Initial Program
;      V1.3   5/5/2012            ;Modified labels, param. passing so it can be spliced easily into the 8086 Monitor


LavaStatus    EQU    090H         ;Status Port
LavaData      EQU    091H         ;Data port

                                  ;Equates for display on SD Systems Video Board (Used In CPM Debugging mode only)
SCROLL        EQU    01H          ;Set scroll direction UP.
LF            EQU    0AH
CR            EQU    0DH
BS            EQU    08H          ;Back space
BELL          EQU    07H
SPACE         EQU    20H
QUIT          EQU    11H          ;Turns off any screen enhancements (flashing, underline etc).
NO$ENHANCEMENT      EQU    17H          ;Turns off whatever is on
FAST          EQU    10H          ;High speed scroll
TAB           EQU    09H          ;TAB ACROSS (8 SPACES FOR SD-BOARD)
ESC           EQU    1BH
DEL           EQU    7FH
CLEAR         EQU    1CH          ;SD Systems Video Board, Clear to EOL. (Use 80 spaces if EOL not available
                                  ;on other video cards)

;Propeller Console IO S-100 board or SD SYSTEMS VIDIO BOARD FOR CONSOLE I/O(<---These must configured for your hardware)

KEYSTAT             EQU    0H
KEYIN         EQU    01H          ;Console input port. Normally the Propeller Driven S-100 Console-IO Board
KEYOUT        EQU    01H          ;Console output port. Normally the Propeller Driven S-100 Console-IO Board


;LAVA Commands:-
COPY$MEMORY   EQU    010H
WRITE$MEMORY  EQU    020H
READ$MEMORY   EQU    030H
DRAW$TEXT     EQU    040H
READ$CSR      EQU    036H
WRITE$CSR     EQU    022H

L_CRT_WIDTH   EQU    800          ;Pixels across per line
L_CRT_HEIGHT  EQU    600          ;Pixels (16 per line)
L_BELOW_SCREEN      EQU    601          ;Area of LAVA screen RAM not visible (Use as a clear buffer area, EOL etc)
L_CHARS_PER_LINE    EQU    99           ;99X8 = 792
L_SCREEN_LINES EQU   37           ;37X16 = 592
L_CHAR_WIDTH  EQU    8            ;Character pixel width
L_CHAR_HEIGHTEQU     16           ;Character pixel height


L_WHITE_COLOREQU     0FFFFH
L_BLACK_COLOREQU     00000H
L_BLUE_COLOR  EQU    00F0FH
```

```
L_GREEN_COLOR EQU    008F0H

;-------------------------------------------------------------------------------------------
              CPU 8086            ;No 80286/386 opcodes
              [BITS 16]

 SECTION      .text
              org    0H           ;<-- This must be 0. All addresses relative to this location
                                  ;For debugging/testing this test program will reside in RAM at E000:2000H

       TIMES  2000H  DB     0H    ;At E000:2000H in RAM for testing/debugging (Monitor "Y" command will do
                                  ;this if it is first loaded to 0:2000H with CPM80 SID.
                                  ;Else goto 0:2000H with "G" 8086 Monitor command.
                                  ;Change the org to whatever value you like

START:                            ;Start of this test program
       MOV    AX,CS
       MOV    SS,AX
       MOV    AX,STACK
       MOV    SP,AX
       MOV    BX,SIGN$ON          ;print a welcome message
       CALL   PRINT_STRING
       MOV    AL,00000000B        ;Set to WRITE mode, no strobes etc
       OUT    LavaStatus,AL       ;Send to lava status port (91H)
       CALL   L_CLEAR_BUFFER             ;Clear a buffer LAVA RAM area for Clear line etc
       CALL   L_CLEAR_SCREEN
       CALL   L_HOME              ;Set cursor X,Y to 0,0
       MOV    BX,L_GREEN_COLOR
       CALL   L_SET_COLOR
       CALl   L_SHOW_CURSOR

NEXT$CHAR:
       CALL   CI                  ;Get a character
       MOV    CL,AL
       CMP    CL,ESC
       JZ     ABORT               ;Back to DOS if ESC

       CMP    CL,'@'
       JNZ    NOT_STR_TEST        ;String Test
       CALL   STR_TEST
       JMP    NEXT$CHAR

NOT_STR_TEST:                     ;See menu at bottom of page for "commands"
       CMP    CL,'!'
       JNZ    NOT_L_CLEAR_EOL
       CALL   L_CLEAR_EOL         ;CLear to End Of Line
       JMP    NEXT$CHAR
NOT_L_CLEAR_EOL:
       CMP    CL,BS               ;Backspace requires special treatment
       JNZ    NOT_L_BACK_SPACE
       CALL   L_BACK_SPACE
       JMP    NEXT$CHAR

NOT_L_BACK_SPACE:
```

```
       CMP    CL,DEL              ;Delete requires special treatment
       JNZ    NOT_L_DEL_SPACE
       CALL   L_DEL_SPACE
       JMP    NEXT$CHAR

NOT_L_DEL_SPACE:
       CMP    CL,CR               ;Delete requires special treatment
       JNZ    NOT_CR
       CALL   L_DO_CR
       JMP    NEXT$CHAR

NOT_CR:       CMP    CL,LF
       JNZ    NOT_LF
       CALL   L_DO_LF
       JMP    NEXT$CHAR

NOT_LF:       CMP    CL,'%'
       JNZ    NOT_L_CLEAR_CURRENT_LINE
       CALL   L_CLEAR_CURRENT_LINE       ;Clear current Line
       JMP    NEXT$CHAR

NOT_L_CLEAR_CURRENT_LINE:
       CMP    CL,'&'
       JNZ    NOT_L_CLEAR_SCREEN
       CALL   L_CLEAR_SCREEN             ;Clear screen. Cursor to 0,0
       JMP    NEXT$CHAR

NOT_L_CLEAR_SCREEN:
       CMP    CL,'('
       JNZ    NOT_SCROLL_UP
       CALL   L_SCROLL_UP_1       ;Scroll up screen 1 line
       JMP    NEXT$CHAR

NOT_SCROLL_UP:
       CMP    CL,'+'
       JNZ    NOT_L_HOME
       CALL   L_HOME              ;Cursor to 0,0
       JMP    NEXT$CHAR

NOT_L_HOME:
       MOV    AH,CL
       CALL   L_TTY_OUT           ;<<<< Send character in [AH] to LAVA board
       JMP    NEXT$CHAR



STR_TEST:
       MOV    BX,L_WHITE_COLOR
       CALL   L_SET_COLOR         ;May as well test this as well
       MOV    CX,2000                    ;Test Rapid sending of characters
       MOV    AH,'1'              ;Start with 1,2,3...
TEST1: CALL   L_TTY_OUT
       LOOP   TEST1
       MOV    BX,L_GREEN_COLOR
```

```
       CALL   L_SET_COLOR
       RET


ABORT: CALL   CRLF
       JMP     word  0F000H:0FFF0H      ;Far Jump to CPU Reset


;------------------ BASIC CRT TREMINAL COMMANDS ---------------------------------------------
;             Only AL register changed unless stated otherewise
;
L_TTY_OUT:                          ;<< CORE FUNCTION >>Write 1 character (in [AH]) to current cursor X,Y position. Update cursor
       CALL   L_HIDE_CURSOR

       MOV    AL,DRAW$TEXT         ;Send Draw Text Command
       CALL   L_PULSE$WR
       MOV    AL,1                 ;Send 1 character only
       CALL   L_PULSE$WR
       MOV    AL,AH
       CALL   L_PULSE$WR           ;Send Ascii
       MOV    AL,0
       CALL   L_PULSE$WR           ;send Ascii X2 (So we have an even number of bytes sent)

       CALL   L_NEXT_POSITION             ;Advance the cursor one position, next line if at EOL, Scroll up if at bottom of screen
       CALL   L_SHOW_CURSOR        ;Show new cursor position

       PUSH   DX
       CALL   L_GET_CURSOR
       MOV    AX,DX
       CALL   AX_HEXOUT
       CALL   SKIP_SPACE
       POP    DX
       RET


L_HOME:                                 ;Cursor to Top left of screen
       PUSH   BX
       CALL   L_HIDE_CURSOR
       XOR    BX,BX
       CALL   L_SET_X
       CALL   L_SET_Y
       CALL   L_SHOW_CURSOR
       POP    BX
       RET



L_SET_CURSOR:                       ;AH = 02h     VIDEO - SET CURSOR POSITION
       PUSH   BX                    ;DH = row (00h is top), DL = column (00h is left)
       XOR    BX,BX
       MOV    BL,DL
       SHL    BX,1                 ;X2
       SHL    BX,1                 ;X4
       SHL    BX,1                 ;X8    (8 Pixels/character)
```

```
        CALL    L_SET_X
        XOR     BX,BX
        MOV     BL,DH
        SHL     BX,1                ;X2
        SHL     BX,1                ;X4
        SHL     BX,1                ;X8
        SHL     BX,1                ;X16    (16 Pixels/character)
        CALL    L_SET_Y
        POP     BX
        RET


                                    ;AH = 03h    VIDEO - GET CURSOR POSITION AND SIZE
L_GET_CURSOR:                       ;DH = row (00h is top), DL = column (00h is left)
        PUSH    BX
        CALL    L_GET_X
        SHR     BX,1                ;/2
        SHR     BX,1                ;/4
        SHR     BX,1                ;/8    (8 Pixels/character)
        MOV     DL,BL
        CALL    L_GET_Y
        SHR     BX,1                ;/2
        SHR     BX,1                ;/4
        SHR     BX,1                ;/8
        SHR     BX,1                ;/16   (16 Pixels/character)
        MOV     DH,BL
        POP     BX
        RET



L_SHOW_CURSOR:                              ;Show cursor at current position
        MOV     AL,DRAW$TEXT        ;Send Draw Text Command
        CALL    L_PULSE$WR
        MOV     AL,1                ;Send 1 character only
        CALL    L_PULSE$WR
        MOV     AL,'_'
        CALL    L_PULSE$WR          ;Send Ascii
        MOV     AL,0
        CALL    L_PULSE$WR          ;Send Ascii X2 (So we have an even number of bytes sent)
        RET



L_HIDE_CURSOR:                              ;Hide cursor at current position
        PUSH    BX
        CALL    L_GET_COLOR         ;Normally white on black
        PUSH    BX                  ;Save For below
        MOV     BX,L_BLACK_COLOR
        CALL    L_SET_COLOR
        MOV     AL,DRAW$TEXT        ;Send Draw Text Command
        CALL    L_PULSE$WR
        MOV     AL,1                ;Send 1 character only
        CALL    L_PULSE$WR
```

```
       MOV    AL,'_'
       CALL   L_PULSE$WR          ;Send Ascii
       MOV    AL,0
       CALL   L_PULSE$WR          ;Send Ascii X2 (So we have an even number of bytes sent)
       POP    BX                  ;Get Back origional color
       CALL   L_SET_COLOR
       POP    BX
       RET



L_DO_CR:
       PUSH   BX                  ;Move cursor to start of line
       CALL   L_HIDE_CURSOR
       XOR    BX,BX
       CALL   L_SET_X
       CALL   L_SHOW_CURSOR
       POP    BX
       RET


L_DO_LF:
       PUSH   BX                  ;Move cursor down vertically one line
       CALL   L_HIDE_CURSOR
       CALL   L_GET_Y
       CMP    BX,(L_SCREEN_LINES * L_CHAR_HEIGHT) - L_CHAR_HEIGHT
       JL     L_DO_LF1
       CALL   L_SCROLL_UP_1       ;Scroll up 1 line
       JMP    L_DO_LF2
L_DO_LF1:
       ADD    BX,L_CHAR_HEIGHT
       CALL   L_SET_Y
L_DO_LF2:
       CALL   L_SHOW_CURSOR
       POP    BX
       RET


L_GET_COLOR:                       ;READ Register command, Get Text Color. Data in BX
       MOV    AL,READ$CSR
       CALL   L_PULSE$WR          ;Send
       MOV    AL,0
       CALL   L_PULSE$WR
       MOV    AL,0
       CALL   L_PULSE$WR
       MOV    AL,0H               ;Point to CSR_COLOR
       CALL   L_PULSE$WR
       CALL   L_PULSE$2RD         ;<<< Read 2 byte into [BX]
       RET


L_SET_COLOR:                       ;WRITE Register command, Set Text Color. Data in BX
       MOV    AL,WRITE$CSR
       CALL   L_PULSE$WR          ;Send
```

```
        MOV     AL,0
        CALL    L_PULSE$WR
        MOV     AL,0
        CALL    L_PULSE$WR
        MOV     AL,0H               ;Point to CSR_COLOR
        CALL    L_PULSE$WR
        MOV     AL,BH
        CALL    L_PULSE$WR
        MOV     AL,BL
        CALL    L_PULSE$WR
        RET


L_NEXT_POSITION:                    ;Advance LAVA cursor to next position (Cursor is not displayed here)
        PUSH    BX
        CALL    L_GET_X             ;Get X position in BX
        CMP     BX,(L_CHARS_PER_LINE * L_CHAR_WIDTH) - L_CHAR_WIDTH
        JL      L_SAME_LINE         ;On Same line just update
        CALL    L_DO_LF
        CALL    L_HIDE_CURSOR
        XOR     BX,BX               ;X=0 always to start of next line
        CALL    L_SET_X
        POP     BX
        RET
L_SAME_LINE:                        ;This is the normal situation
        ADD     BX,8
        CALL    L_SET_X
        POP     BX
        RET


L_CLEAR_CURRENT_LINE:               ;Clear a whole line at current cursor Y position (Any X position on that lne)
        PUSH    BX
        PUSH    CX                  ;No need to hide cursor
        PUSH    SI
        CALL    L_GET_X
        PUSH    BX                  ;Store it for when we return below

        CALL    L_GET_Y
        MOV     SI,BX
        CALL    L_CLEAR_LINE

        POP     BX                  ;Get Back origional Cursor position
        CALL    L_SET_X
        POP     SI
        POP     CX
        POP     BX
        RET


L_CLEAR_SCREEN:                     ;Clear the whole screen. Cursor to 0,0
        PUSH    BX
        PUSH    CX
        PUSH    SI
```

```
       MOV    SI,0
       MOV    CX,L_SCREEN_LINES+1 ;Count of total lines on screen
CLEAR1:       PUSH   CX
       PUSH   SI
       CALL   L_CLEAR_LINE
       POP    SI
       POP    CX
       ADD    SI,L_CHAR_HEIGHT
       LOOP   CLEAR1
       XOR    BX,BX               ;Set cursor position 0,0
       CALL   L_SET_X
       CALL   L_SET_Y
       POP    SI
       POP    CX
       POP    BX
       RET


L_CLEAR_LINE:                     ;Clear line at [SI]. Note BX & DX changed
       MOV    CX,0                ;Count for lines below
L_CLEAR_LINE1:
       MOV    BX,L_BELOW_SCREEN          ;SOURCE: Below bottom of screen (will display as background, see L_CLEAR_BUFFER)
       ADD    BX,CX
       MOV    DX,0                ;X position is 0
       CALL   L_MAKE_24_ADDRESS          ;Generate LAVA 24 bit address DX+BX -> DX+BH

       MOV    AL,COPY$MEMORY             ;Sent COPY MEMORY command
       CALL   L_PULSE$WR
       MOV    AL,BH
       CALL   L_PULSE$WR          ;Source Address 23:16
       MOV    AL,DH
       CALL   L_PULSE$WR          ;Source Address 15:8
       MOV    AL,DL
       CALL   L_PULSE$WR          ;Source Address 7:0

       MOV    AL,03H
       CALL   L_PULSE$WR          ;Next two size bytes
       MOV    AL,0FFH
       CALL   L_PULSE$WR

       MOV    AX,0
       CALL   L_PULSE$WR          ;Filler byte

       MOV    BX,SI               ;DESTINATION:  Get Y position
       ADD    BX,CX
       MOV    DX,0                ;X position is 0
       CALL   L_MAKE_24_ADDRESS          ;Generate LAVA 24 bit address DX+BX -> DX+BH
       MOV    AL,BH
       CALL   L_PULSE$WR          ;Send Address 23:16
       MOV    AL,DH
       CALL   L_PULSE$WR          ;Send Address 15:8
       MOV    AL,DL
       CALL   L_PULSE$WR          ;Send Address 7:0
       INC    CX
```

```
       CMP    CX,L_CHAR_HEIGHT
       JNZ    L_CLEAR_LINE1
       RET


L_CLEAR_EOL:                        ;Clear to EOL (Any X position to end of that line)
       PUSH   BX
       PUSH   CX
       PUSH   DX
       PUSH   SI
       PUSH   DI
       CALL   L_HIDE_CURSOR
       CALL   L_GET_X                      ;GET X POSITION OF CURSOR IN BX (Number of character positions)
       PUSH   BX                   ;Save Cursor for when done

       MOV    SI,BX                ;>>> X Position in SI <<<
       MOV    DX,L_CRT_WIDTH
       SUB    DX,BX                ;>>> Length in DX <<<< of line in delete area left to EOL

       CALL   L_GET_Y                      ;GET Y POSITION OF CURSOR IN BX (Number of character positions)
       MOV    DI,BX                ;>>> Y Position in DI <<<

       MOV    CX,0                 ;Count for lines for below
EOL_LINE1:                          ;Will move a clear memory block into the area
       PUSH   DX                   ;Save data
       MOV    BX,L_BELOW_SCREEN            ;SOURCE: Below bottom of screen (will display as background, see L_CLEAR_BUFFER)
       ADD    BX,CX
       MOV    DX,SI
       CALL   L_MAKE_24_ADDRESS            ;Generate LAVA 24 bit address DX+BX -> DX+BH

       MOV    AL,COPY$MEMORY               ;Sent COPY MEMORY command
       CALL   L_PULSE$WR
       MOV    AL,BH
       CALL   L_PULSE$WR           ;Source Address 23:16
       MOV    AL,DH
       CALL   L_PULSE$WR           ;Source Address 15:8
       MOV    AL,DL
       CALL   L_PULSE$WR           ;Source Address 7:0

       POP    DX                   ;Get back saved length
       MOV    AL,DH
       CALL   L_PULSE$WR           ;Next two size bytes
       MOV    AL,DL
       CALL   L_PULSE$WR
       PUSH   DX                   ;Save length again

       MOV    AX,0
       CALL   L_PULSE$WR           ;Filler byte

       MOV    BX,DI                ;DESTINATION:  Get Y position
       ADD    BX,CX
       MOV    DX,SI                ;X position is 0
       CALL   L_MAKE_24_ADDRESS            ;Generate LAVA 24 bit address DX+BX -> DX+BH
       MOV    AL,BH
```

```
        CALL    L_PULSE$WR          ;Send Address 23:16
        MOV     AL,DH
        CALL    L_PULSE$WR          ;Send Address 15:8
        MOV     AL,DL
        CALL    L_PULSE$WR          ;Send Address 7:0

        POP     DX                  ;balance up stack
        INC     CX
        CMP     CX,L_CHAR_HEIGHT
        JNZ     EOL_LINE1

        POP     BX                  ;Get Back origional Cursor position
        CALL    L_SET_X
        CALL    L_SHOW_CURSOR
        POP     DI
        POP     SI
        POP     DX
        POP     CX
        POP     BX
        RET


L_SCROLL_UP:                        ;IBM BIOS, Scroll up screen a number of lines
        PUSH    AX                  ;[AL] has number of lines to scroll up
        CALL    L_SCROLL_UP_1       ;Scroll up one line
        POP     AX
        DEC     AL
        JNZ     L_SCROLL_UP
        RET

L_SCROLL_UP_1:                          ;Move the whole screen up one line (quickly)
        PUSH    BX
        PUSH    CX
        PUSH    SI
        PUSH    DI
        CALL    L_GET_Y
        PUSH    BX                  ;Store it for when we return below
        CALL    L_HIDE_CURSOR

        MOV     CX,(L_CHAR_HEIGHT * L_SCREEN_LINES) - L_CHAR_HEIGHT  ;(Count for screen scan lines below)
        MOV     SI,L_CHAR_HEIGHT                                ;Source, one line down
        MOV     DI,0                                            ;Destination, top of screen

UP1:    MOV     BX,SI               ;SOURCE
        MOV     DX,0                ;X position is 0
        CALL    L_MAKE_24_ADDRESS   ;Generate LAVA 24 bit address DX+BX -> DX+BH

        MOV     AL,COPY$MEMORY              ;Sent COPY MEMORY command
        CALL    L_PULSE$WR
        MOV     AL,BH
        CALL    L_PULSE$WR          ;Source Address 23:16
        MOV     AL,DH
        CALL    L_PULSE$WR          ;Source Address 15:8
        MOV     AL,DL
```

```
        CALL    L_PULSE$WR          ;Source Address 7:0

        MOV     AL,03H
        CALL    L_PULSE$WR          ;Next two size bytes
        MOV     AL,0FFH
        CALL    L_PULSE$WR

        MOV     AX,0
        CALL    L_PULSE$WR          ;Filler byte

        MOV     BX,DI               ;DESTINATION
        MOV     DX,0                ;X position is 0
        CALL    L_MAKE_24_ADDRESS   ;Generate LAVA 24 bit address DX+BX -> DX+BH
        MOV     AL,BH
        CALL    L_PULSE$WR          ;Send Address 23:16
        MOV     AL,DH
        CALL    L_PULSE$WR          ;Send Address 15:8
        MOV     AL,DL
        CALL    L_PULSE$WR          ;Send Address 7:0

        INC     SI                  ;Next scan line
        INC     DI
        LOOP    UP1

        MOV     SI,(L_CHAR_HEIGHT * L_SCREEN_LINES)
        CALL    L_CLEAR_CURRENT_LINE

        POP     BX                  ;Get Back origional Y Cursor position
        CALL    L_SHOW_CURSOR
        POP     DI
        POP     SI
        POP     CX
        POP     BX
        RET



L_DEL_SPACE:                        ;DEL requires special treatment because LAVA does not
        PUSH    BX                  ;Overwrite characters. ie a space will not delete a character if overwritten
        PUSH    CX
        PUSH    DX
        CALL    L_HIDE_CURSOR
        CALL    L_GET_X
        JMP     L_PUT_SPACE
L_BACK_SPACE:                       ;Back space requires special treatment because LAVA does not
        PUSH    BX                  ;Overwrite characters. ie a space will not delete a character if overwritten
        PUSH    CX
        PUSH    DX
        CALL    L_HIDE_CURSOR
        CALL    L_GET_X
        CMP     BX,8
        JL      L_PUT_SPACE
        SUB     BX,8
        CALL    L_SET_X                   ;Back space one character
```

```
L_PUT_SPACE:                    ;Print a space on Screen at current X,Y position
       CALL   L_GET_COLOR
       PUSH   BX                ;Save for below
       MOV    BX,L_BLACK_COLOR         ;The trick is to overlay with two ASCII characters that fill the complete
       CALL   L_SET_COLOR       ;8X16 pixel area in black (If a different background color, then change)

       MOV    AL,DRAW$TEXT      ;Send Draw Text Command
       CALL   L_PULSE$WR
       MOV    AL,1              ;Send 1 character only
       CALL   L_PULSE$WR
       MOV    AL,03H
       CALL   L_PULSE$WR        ;Send Ascii 'Heart figure"
       MOV    AL,0
       CALL   L_PULSE$WR        ;send Ascii X2 (So we have an even number of bytes sent)

       MOV    AL,DRAW$TEXT      ;Send Draw Text Command
       CALL   L_PULSE$WR
       MOV    AL,1              ;Send 1 character only
       CALL   L_PULSE$WR
       MOV    AL,08H
       CALL   L_PULSE$WR        ;Overlay with ASCII "Circle figure"
       MOV    AL,0
       CALL   L_PULSE$WR        ;send Ascii X2 (So we have an even number of bytes sent)

       POP    BX                ;Get back color
       CALL   L_SET_COLOR
       CALL   L_SHOW_CURSOR
       POP    DX
       POP    CX
       POP    BX
       MOV    AL,BS             ;Return with space in AL
       RET


L_MAKE_24_ADDRESS:              ;Generate LAVA 24 bit memory address from X in DX, and Y in BX
       MOV    AX,DX             ;Get X address, isolate bits 8 & 9
       AND    AH,03H
       SHL    BX,1              ;Shift over Y by 2 bits
       SHL    BX,1
       OR     AH,BL             ;Combine in lower 6 bits of Y coordinate
       MOV    DH,AH
       RET                      ;Return Address 7:0 in DL, 15:8 in DH, and 23:16 in BH


L_GET_X:
       MOV    AL,READ$CSR       ;READ Register command for X Position into BX
       CALL   L_PULSE$WR        ;Send
       MOV    AL,0
       CALL   L_PULSE$WR
       MOV    AL,0
       CALL   L_PULSE$WR
       MOV    AL,01H            ;Point to CSR_FONT_X
       CALL   L_PULSE$WR
       CALL   L_PULSE$2RD       ;<<< Read 2 byte into [BX]
```

```
       RET


L_GET_Y:
       MOV    AL,READ$CSR         ;READ Register command for Y Position into BX
       CALL   L_PULSE$WR          ;Send
       MOV    AL,0
       CALL   L_PULSE$WR
       MOV    AL,0
       CALL   L_PULSE$WR
       MOV    AL,02H              ;Point to CSR_FONT_Y
       CALL   L_PULSE$WR
       CALL   L_PULSE$2RD         ;<<< Read 2 byte into [BX]
       RET


L_SET_X:
       MOV    AL,WRITE$CSR        ;WRITE Register command, X Position, data in BX
       CALL   L_PULSE$WR          ;Send
       MOV    AL,0
       CALL   L_PULSE$WR
       MOV    AL,0
       CALL   L_PULSE$WR
       MOV    AL,01H              ;Point to CSR_FONT_X
       CALL   L_PULSE$WR
       MOV    AL,BH
       CALL   L_PULSE$WR
       MOV    AL,BL
       CALL   L_PULSE$WR
       RET


L_SET_Y:
       MOV    AL,WRITE$CSR        ;WRITE Register command, Y Position, data in BX
       CALL   L_PULSE$WR          ;Send
       MOV    AL,0
       CALL   L_PULSE$WR
       MOV    AL,0
       CALL   L_PULSE$WR
       MOV    AL,02H              ;Point to CSR_FONT_Y
       CALL   L_PULSE$WR
       MOV    AL,BH
       CALL   L_PULSE$WR
       MOV    AL,BL
       CALL   L_PULSE$WR
       RET



                                  ;Clear an area of the LAVA screen RAM for use with Clear Line, EOL etc.
L_CLEAR_BUFFER:                          ;We will us this for fast LAVA block moves etc.
       MOV    BX,L_BELOW_SCREEN          ;Y position of RAM below bottom of visible screen
CLEARB2:MOV   DX,0                ;X position is 0
CLEARB1:PUSH  BX
```

```
        PUSH    DX
        CALL    L_MAKE_24_ADDRESS          ;Generate LAVA 24 bit address DX+BX -> DX+BH
        MOV     AL,WRITE$MEMORY            ;Sent COPY MEMORY command
        CALL    L_PULSE$WR
        MOV     AL,BH                ;Address 23:16
        CALL    L_PULSE$WR
        MOV     AL,DH                ;Address 15:8
        CALL    L_PULSE$WR
        MOV     AL,DL                ;Address 7:0
        CALL    L_PULSE$WR
        MOV     AL,L_BLACK_COLOR           ;BLACK (0FH = Blue for testing)
        CALL    L_PULSE$WR
        MOV     AL,L_BLACK_COLOR           ;BLACK
        CALL    L_PULSE$WR
        POP     DX
        POP     BX
        INC     DX                   ;Are we at end of line
        CMP     DX,L_CRT_WIDTH+10
        JLE     CLEARB1
        INC     BX                         ;Go to next scan line
        CMP     BX,L_BELOW_SCREEN+L_CHAR_HEIGHT  ;16 scan lines total
        JLE     CLEARB2
        RET


;------------------------ LAVA CORE WRITE ROUTINE -------------------------

                                   ;Note only [AL] altered
L_PULSE$WR:                        ;>>>>> WRITE ONE BYTE OF DATA TO LAVA CHIP, Data in [AL] <<<<<
        OUT     LavaData,AL        ;Send [AL] to Lava data port (91H)
        MOV     AL,00000001B       ;Output enable U10 to LAVA data bus, and set LAVA to WRITE mode
        OUT     LavaStatus,AL      ;Send to lava status port (90H)
        MOV     AL,10000001B       ;Then pulse status port strobe bit LOW (Bit 7 high, pulsed strobe low)
        OUT     LavaStatus,AL      ;Send to lava status port (90H)
L_WR$NOT$RDY:
        IN      AL,LavaStatus      ;Wait until LAVA "Done" signal clears U12A. Then we are done
        AND     AL,80H             ;This will set strobe bit back HIGH. Note still in WRITE LAVA mode
        JZ      L_WR$NOT$RDY
        RET

;------------------------ LAVA CORE READ ROUTINE -------------------------

                                   ;Note only [AL] & [BX] altered
L_PULSE$2RD:                       ;>>>>> READ TWO BYTES OF DATA FROM LAVA CHIP, Data in [BX] <<<<<
        MOV     AL,00001000B       ;Set to LAVA READ MODE, Disable U10 to LAVA data bus
        OUT     LavaStatus,AL      ;Send to lava status port (91H)
        MOV     AL,10001000B       ;Then pulse strobe bit LOW
        OUT     LavaStatus,AL      ;Send to lava status port (91H)
L_RD$NOT$RDY:
        IN      AL,LavaStatus      ;Wait until JAVA "Done" signal clears U12A. Then we are done
        AND     AL,80H             ;This will set strobe bit back HIGH. Note still in READ LAVA mode
        JZ      L_RD$NOT$RDY

        IN      AL,LavaData        ;Data [15:8] from port (90H)
```

```
       MOV    BH,AL               ;Save in BH

       MOV    AL,10001000B        ;Pulse strobe bit LOW
       OUT    LavaStatus,AL       ;Send to lava status port (91H)
L_RD$NOT$RDY1:
       IN     AL,LavaStatus       ;Wait until JAVA "Done" signal clears U12A. Then we are done
       AND    AL,80H              ;This will set strobe bit back HIGH. Note still in READ LAVA mode
       JZ     L_RD$NOT$RDY1
                                  ;Now Second Byte
       IN     AL,LavaData         ;Data [7:0] from port (90H)
       MOV    BL,AL
       RET                        ;Return with data in [BX]



;------------------------- THIS PROGRAM SUPPORT ROUTINES ----------------------------------------------

CO:                               ;Character in CL
       IN     AL,KEYSTAT          ;PROPELLER CONSOLE (or SD SYSTEMS) VIDIO BOARD PORT
       AND    AL,4H
       JZ     CO
       MOV    AL,CL
       OUT    KEYOUT,AL
       MOV    AL,CL               ;MAKE SURE TO RETURN WITH [AL] CONTAINING CHAR
       RET


CI:                               ;Return with character in AL
       CALL   CSTS                ;Wait until something is there
       JZ     CI
       IN     AL,KEYIN
       AND    AL,7FH
       RET
CSTS:  IN     AL,KEYSTAT
       TEST   AL,02H
       JZ     NONE
       XOR    AL,AL
       DEC    AL
       RET                        ;RETURN WITH 0FFH IN [A] IF SOMETHING
NONE:  XOR    AL,AL
       RET


CRLF:  PUSH   AX                  ;Send CR/LF to console. No registers changed
       PUSH   BX
       PUSH   CX
       PUSH   DX
       MOV    CL,CR
       CALL   CO
       MOV    CL,LF
       CALL   CO
       POP    DX
       POP    CX
       POP    BX
```

```
       POP    AX
       RET

SKIP_SPACE:
       PUSH   AX                  ;Send CR/LF to console. No registers changed
       PUSH   BX
       PUSH   CX
       PUSH   DX
       MOV    CL,' '
       CALL   CO
       POP    DX
       POP    CX
       POP    BX
       POP    AX
       RET

PRINT_STRING:                     ;Use CS over-ride so it will splice into 8086 BIOS easily
       push   cx
print1:       mov    al,[CS:bx]          ;Note this routine does NOT assume DS = CS here.
       inc    bx                  ;By using the CS over-ride we will always have
       cmp    al,'$'              ;a valid pointer to messages at the end of this monitor
       jz     print2
       cmp    AL,0                ;Also terminate with 0's
       JZ     print2
       mov    cl,al
       call   CO
       jmp    print1
print2:       pop    cx
       ret

AX_HEXOUT:
       PUSH   AX
       PUSH   BX
       PUSH   CX
       PUSH   AX
       MOV    Al,AH
       CALL   AL_HEXOUT
       POP    AX
       CALL   AL_HEXOUT
       POP    CX
       POP    BX
       POP    AX
       RET

;      AL_HEXOUT                  ;output the 2 hex digits in [AL]
AL_HEXOUT:                        ;No registers altered (except AL)
       push   cx
       push   ax
       mov    cl,4                ;first isolate low nibble
       shr    al,cl
       call   hexdigout
       pop    ax
       call   hexdigout           ;get upper nibble
       pop    cx
```

```
        ret

hexdigout:
        and     al,0fh              ;convert nibble to ascii
        add     al,90h
        daa
        adc     al,40h
        daa
        mov     cl,al
        call    CO
        ret

SIGN$ON:      DB     CR,LF,'LAVA-10 8086 Test Program Menu 4/28/2012. John Monahan.  (V1.25) '
              DB     CR,LF,'Type characters from keyboard. Type "@" for string test.'
              DB     CR,LF,'Type ! To Clear EOL.             @ For String test.'
              DB     CR,LF,'Type % To Clear Line.            & To Clear Screen'
              DB     CR,LF,'Type ( To scroll screen up.      + To L_HOME cursor'
              DB     CR,LF,'BS, DEL, CR, LF (^J) all work key and DEL work'
              DB     CR,LF,'ESC to abort.'
              DB     CR,LF,LF,'$'

TestString    DB     '  >>> This is a test for rapidly sending characters 1234567890. <<<< $'


        TIMES  100H   DB     0H     ;Space for stack
        STACK  DW 0
```