```
    .include ".\Hardware.asm"

CmdLineLength = DISPCHARS

; set which bit if the tick counter to use for flashing the cursor
; D7=1 second, D6=1/2 second, d5=1/4 second, d4=1/8 second, etc...
TickMask = 0x40


    .area   RAM    (REL)                ;RAM area is relative

LastTick:   .ds     1                   ;last system tick value - used for

CmdIdx:     .db     0                   ;index into current line for curso
CmdPtr:     .dw     0                   ;pointer to current cursor byte
CmdPtrLast: .dw     0                   ;pointer to last command line ente
PromptLen:  .db     0                   ;length of prompt on current line

LineBuffer: .ds     DISPCHARS*DISPLINES
LB_EOB:     .ds     1                   ;MUST be immediately following Lin

    .area   CODE   (REL)                ;program area CODE is relative

    .globl InitCmdLine, CmdLineScan
    .globl CmdNewLine, CmdWRChar, CmdWRPrompt, CmdWRString, CmdBackSpace
    .globl CmdWRA, ClrLineBuffer

    ;--------------------------------------
    ;initialization logic
    ;--------------------------------------

InitCmdLine:                            ;One time initialization
    xor     a                           ;
    ld      (CmdIdx),a                  ;setup index
    ld      (PromptLen),a               ;clear prompt length
    ld      hl,#LineBuffer              ;
    ld      (CmdPtr),hl                 ;setup pointer
    ld      (CmdPtrLast),hl             ;setup last command line pointer

.if     USE_PU_LED_CODES
    ld      a,#LEDCLI1
    out     (LEDPORT), a
.endif
              ;
    call    ClrLineBuffer               ;clear line buffer
```

```
 .if     USE_PU_LED_CODES
    ld      a,#LEDCLI2
    out     (LEDPORT), a
 .endif

    call    DispUpdate                  ;update the display

 .if     USE_PU_LED_CODES
    ld      a,#LEDCLI3
    out     (LEDPORT), a
 .endif

    ret

    ;--------------------------------------
    ;main logic
    ;--------------------------------------


 ;-------------------------------------------------------------------------
 ;DispUpdate
 ;
 ;update display to reflect the current buffer state
 ;
 ;Assumes buffer cursor is in the first column (after NewLine)
 ;
 ;HL, BC, DE, A destroyed
 ;-------------------------------------------------------------------------

 DispUpdate:                            ;

    ;**********************************
    ; Force the video cursor to the
    ; home position
    ;**********************************

    ld      a,#DISPHOME                 ;
    call    DisplayA                    ;move the cursor to the home posit

 .if     USE_PU_LED_CODES
    ld      a,#LEDCLU1
    out     (LEDPORT), a
 .endif
```

```
    ;**********************************
    ; Fill new line buffer area with
    ; spaces
    ;**********************************

    ld      hl,(CmdPtr)                 ;get pointer to current cursor pos
    ld      b,#DISPCHARS                ;get #of characters in the line
DULoop1:                                ;Repeat
    ld      a,#0x20                     ;  get space fill character
    ld      (hl),a                      ;  clear out next character in cur
    call    IncBufPtr                   ;  adjust HL to point to next buff
    djnz    DULoop1                     ;Until (current line is all blanks

.if     USE_PU_LED_CODES
    ld      a,#LEDCLU2
    out     (LEDPORT), a
.endif

    ;**********************************
    ; Update screen with buffer contents
    ;**********************************

    ld      bc,#DISPCHARS*DISPLINES     ;get total #of characters on scree
                                        ;
DULoop2:                                ;Repeat
    ld      a,(hl)                      ;  get next character on 'next' li
    call    DisplayA                    ;  write it to the display
                                        ;
    call    IncBufPtr                   ;  adjust HL to point to next buff
                                        ;
    dec     bc                          ;
    ld      a,b                         ;
    or      c                           ;
    jr      nz,DULoop2                  ;Until (all characters have been u

.if     USE_PU_LED_CODES
    ld      a,#LEDCLU3
    out     (LEDPORT), a
.endif

    ;**********************************
    ; Move video cursor back to column 1
    ; on last line
    ;**********************************
```

```
    ld      b,#DISPCHARS                ;get #of characters in the line
DULoop3:                                ;Repeat
    call    DispBS                      ;  issue backspace sequence to vid
    djnz    DULoop3                     ;Until (video cursor is at the sta

.if     USE_PU_LED_CODES
    ld      a,#LEDCLU4
    out     (LEDPORT), a
.endif

    ret                                 ;

;-----------------------------------------------------------------------
;IncBufPtr
;
;DE - end of buffer address + 1
;HL++ taking buffer wrap into consideration
;
;HL, DE, A destroyed
;-----------------------------------------------------------------------

IncBufPtr:
    ld      de,#LB_EOB                  ;get end of LineBuffer address
                                        ;
    inc     hl                          ;increment line buffer pointer
                                        ;
    ld      a,h                         ;
    cp      d                           ;check for end of buffer
    jr      nz,IBPDone                  ;
    ld      a,l                         ;
    cp      e                           ;
    jr      nz,IBPDone                  ;if (HL went past the end of LineB
    ld      hl,#LineBuffer              ;  reset HL to the start of the Li
IBPDone:                                ;end
    ret

;-----------------------------------------------------------------------
;ClrLineBuffer
;
;DE - end of buffer address + 1
;HL++ taking buffer wrap into consideration
;
;HL, DE, A destroyed
;-----------------------------------------------------------------------
```

```
ClrLineBuffer:
    ld      hl,#LineBuffer              ;reset HL to the start of the Line
    ld      bc,#DISPCHARS*DISPLINES     ;get total #of characters on scree
CLBLoop:                                ;repeat
    ld      a,#0x20                     ;  set fill character to space
    ld      (hl),a                      ;  clear next buffer byte
    inc     hl                          ;  increment pointer
    dec     bc                          ;
    ld      a,b                         ;  decrement loop count
    or      c                           ;
    jr      nz,CLBLoop                  ;Until (all of buffer has been cle
    ret


;-------------------------------------------------------------------------
;CmdNewLine
;
;fill rest of line with spaces
;
;HL, A, B destroyed
;-------------------------------------------------------------------------

CmdNewLine:                             ;
    ld      a,(CmdIdx)                  ;Get the current index
    ld      b,a                         ;
    ld      a,#CmdLineLength            ;
    sub     b                           ;#of spaces needed = line length -
    ld      b,a                         ;
MonPad:                                 ;Repeat
    ld      a,#0x20                     ;  use ' ' as fill character
    push    bc                          ;
    call    CmdWRChar                   ;  write character to command buff
    pop     bc                          ;
    djnz    MonPad                      ;Until (entire line is padded out)
    ret


;-------------------------------------------------------------------------
;CmdWRChar
;
;write a single character to the command line (CmdWRA same but slower - sa
;
;A = character to write
;
;all registers destroyed (except CmdWRA)
```

```
;-----------------------------------------------------------------------

CmdWRA:
    push    bc
    push    de
    push    hl
    call    CmdWRChar
    pop     hl
    pop     de
    pop     bc
    ret


CmdWRChar:                              ;
    ld      hl,(CmdPtr)                 ;
    ld      (hl),a                      ;store the character in the commar
                                        ;
    push    af                          ;
    call    IncBufPtr                   ;adjust HL to point to next buffer
    ld      (CmdPtr),hl                 ;adjust the pointer up
    pop     af                          ;
                                        ;
    call    DisplayA                    ;display the character on the scre
                                        ;
    ld      a,(CmdIdx)                  ;
    inc     a                           ;adjust index up
    ld      (CmdIdx),a                  ;
    cp      #CmdLineLength              ;check for end of command line
    jr      nz,WCNEOL                   ;if (EOL)
    xor     a                           ;
    ld      (CmdIdx),a                  ;  reset index
    call    DispUpdate                  ;  update the display

WCNEOL:                                 ;endif
    ret




;-----------------------------------------------------------------------
;CmdWRString, CmdWRPrompt
;
;write a string/prompt to the command line
;
;Prompt differs in that you can't backspace into a prompt, AND that a comm
```

```
;defined as the line on which a prompt has been placed.  In other words a
;is REQUIRED to gather user input.
;
;HL = string to write
;
;all registers destroyed
;-------------------------------------------------------------------------

CmdWRPrompt:
    call    StrLen                      ;calculate string length
    ld      (PromptLen),a               ;update prompt length
    ld      de,(CmdPtr)                 ;get the current command pointer
    ld      (CmdPtrLast),de             ;store it as the last command poin

CmdWRString:                            ;
    ld      b,#DISPCHARS*2              ;limit string to two monitor lines
CWSLoop:                                ;Repeat
    ld      a,(hl)                      ;  get next string character
    or      a                           ;
    jr      z,WRCDone                   ;  *abort on terminator
    push    bc                          ;
    push    hl                          ;
    call    CmdWRChar                   ;  write character to command line
    pop     hl                          ;
    pop     bc                          ;
    inc     hl                          ;  point to next character
    djnz    CWSLoop                     ;Until (end of string OR safety li
WRCDone:
    ret

;-------------------------------------------------------------------------
;CmdBackSpace
;
;remove the last character from cmd line
;
;HL, A destroyed
;-------------------------------------------------------------------------

CmdBackSpace:
    ld      a,(CmdIdx)                  ;check for empty line taking promp
    ld      l,a                         ;
    ld      a,(PromptLen)               ;
    cp      l                           ;
    jr      z,CBSDone                   ;if (cursor not at start of comman
    ld      a,#0x20                     ;  ensure the current cursor posit
```

```
    call    DisplayA                    ;    (and not displaying the curso
    call    DispBS                      ;  issue backspace sequence to vid
    call    DispBS                      ;  issue backspace sequence to vid
    ld      a,#0x20                     ;
    call    DisplayA                    ;  write a space character over th
    call    DispBS                      ;  issue backspace sequence to vid
    ld      a,(CmdIdx)                  ;
    dec     a                           ;  adjust command index down for b
    ld      (CmdIdx),a                  ;
    ld      hl,(CmdPtr)                 ;
    dec     hl                          ;  adjust command pointer back for
    ld      a,#0x20                     ;
    ld      (hl),a                      ;  clear out 'backspace'd' buffer
    ld      (CmdPtr),hl                 ;
CBSDone:                                ;endif
    ret



;-------------------------------------------------------------------------
;CmdLineScan
;
;Process command line
;
;returns A !0 - command line complete
;        HL - pointer to command line
;             (if A != 0)
;
;all registers destroyed
;-------------------------------------------------------------------------

CmdLineScan:
    call    ReadKybd                    ;check for keyboard input
    or      a                           ;
    jr      z,CmdDone                   ;if (a key was pressed on the keyb
    cp      #0x0d                       ;  check for carriage return
    jr      z,CmdCR                     ;  if (!carriage return) then
    cp      #0x08                       ;    check for backspace character
    jr      z,CmdBS                     ;    if (!backspace) then
    call    CmdWRChar                   ;      write character to command
    ld      a,(CmdIdx)                  ;
    cp      #CmdLineLength              ;      check for end of command bu
    jr      z,CmdCR                     ;      *simulate carriage return i
    ld      a,#0                        ;      indicate command is not com
```

```
    jr      CmdDone                     ;
CmdBS:                                  ;    else
    call    CmdBackSpace                ;      backspace sequence
    ld      a,#0                        ;      indicate command is not com
    jr      CmdDone                     ;    endif
CmdCR:                                  ;  else
    call    CmdNewLine                  ;    fill out the rest of the line
    call    DispUpdate                  ;    update the display

    ld      hl,(CmdPtrLast)             ;    get the last command line poi
    xor     a                           ;
    ld      (PromptLen),a               ;    reset prompt length
    ld      a,#1                        ;    indicate command line is read
                                        ;  endif
CmdDone:                                ;endif
    push    af                          ;save command ready status
    push    hl                          ;save command line pointer

    ;****************************
    ;Implement Flashing Cursor
    ;****************************

    ld      a,(TickCtr)                 ;get system tick counter
    ld      b,a                         ;
    ld      a,(LastTick)                ;
    xor     b                           ;check to see if flash bit changed
    and     #TickMask                   ;
    jr      z,FlashSkip                 ;if (time to update cursor)
    ld      a,b                         ;  restore tick counter
    ld      (LastTick),a                ;  update last tick
    and     #TickMask                   ;  test flash bit state
    jr      nz,FlashH                   ;  if (bit is low)
    ld      a,#0x20                     ;    setup to write a space charac
    jr      FlashDone                   ;
FlashH:                                 ;  else
    ld      a,#0x5f                     ;    setup to write an underline c
FlashDone:                              ;  endif
    call    DisplayA                    ;  update cursor character
    call    DispBS                      ;  issue backspace sequence to vid
FlashSkip:                              ;endif

    pop     hl                          ;restore command line pointer
    pop     af                          ;restore command ready status
    ret
```