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

    .globl DisplayA, DisplayString, DisplayDE, DisplayNibble, DispBS, Disp
    .globl SkipWhite

    .include ".\Hardware.asm"

;-------------------------------------------------------------------------
;DispBS/DispFS
;
;Move the cursor back/forward one character
;
;The Digital Group video card requires that
;there was a prior falling edge on D7 for
;this to work.  This is not normally a problem
;as any prior characters written will satisfy
;this requirement.  However this will fail if
;the back/forward is the first command issued
;to the card (very unlikely!)
;
;A is destroyed, no other regs affected
;-------------------------------------------------------------------------
DispBS:
    ld      a,#0x02             ;move cursor back one position
    out     (DISPPORT), a
    ld      a,#0x00
    out     (DISPPORT), a
    ret

DispFS:
    ld      a,#0x01
    out     (DISPPORT), a
    ld      a,#0x00
    out     (DISPPORT), a
    ret


;-------------------------------------------------------------------------
;DisplayA
;
;Display the A reg as an ASCII value
;
;A is destroyed, no other regs affected
;-------------------------------------------------------------------------
DisplayA:
```

```
    or      #DISPWRBIT
    out     (DISPPORT), a
    xor     a
    out     (DISPPORT), a
    ret


;-----------------------------------------------------------------------
;DisplayString
;
;Display the NULL terminated string
;pointed to by HL
;
;A and HL affected
;-----------------------------------------------------------------------
DisplayString:
.if USE_CMD_DISP
    call    CmdWRString
    ret
.else
                                    ;Repeat
    ld      a,(hl)                  ;  get next character in string
    or      a
    ret     z                       ;  *Abort when NULL character foun
    call    DisplayA                ;  display the character
    inc     hl                      ;  point to next character in stri
    jp      DisplayString           ;Until (forever)
.endif

;-----------------------------------------------------------------------
;DisplayDE
;
;Display the hex value in DE
;
;A affected
;-----------------------------------------------------------------------
DisplayDE:
    ld      a, d                    ;Display D15:D12 nibble
    call    HexDispA
    ld      a, e                    ;Display D7:D4 nibble
    call    HexDispA
    ret

;-----------------------------------------------------------------------
;HexDispA
```

```
;
;Display the hex value in A
;
;A affected
;-------------------------------------------------------------------------
HexDispA:
    push    af
    rra
    rra
    rra
    rra
    call    DisplayNibble
    pop     af
    call    DisplayNibble
    ret                                 ;

;-------------------------------------------------------------------------
;DisplayNibble
;
;Display the LSN in 'a' as ASCII hex
;Ex: A=0x0a  ->  'A' displayed
;    A=0x05  ->  '5' displayed
;
;only A is affected
;-------------------------------------------------------------------------
DisplayNibble:

    and     #0x0f
    cp      #10
    jr      nc, DNAlpha                 ;if (A < 10)
    add     #0x30                       ;  offset value to '0'
    jr      DNDone
DNAlpha:                                ;else
    add     #(0x41 - 10)                ;  offset value to 'A'
DNDone:
.if USE_CMD_DISP
    call    CmdWRA
.else
    call    DisplayA
.endif
    ret

;-------------------------------------------------------------------------
;StrLen
;
```

```
;Get length of NULL terminated string
;pointed to by HL
;
;  A - returns length
;
;  NOTE: routine terminates at length 255 to avoid endless loops
;
;A affected
;-------------------------------------------------------------------------
StrLen:                                 ;
    push    hl
    push    bc
    ld      b,#0xff                     ;set search limit to a maximum str
StrLen1:                                ;repeat
    ld      a,(hl)                      ;
    or      a                           ;
    jr      z,StrLen2                   ;  *abort if null terminator found
    inc     hl                          ;  adjust pointer
    djnz    StrLen1                     ;until (terminator found OR search
                                        ;if search limit reached b == 0 wh
                                        ;  after inversion below
StrLen2:                                ;
    ld      a,b                         ;get down count
    cpl                                 ;invert it to get string length
    pop     bc
    pop     hl
    ret


;-------------------------------------------------------------------------
;HexInput
;
;Process ASCII hex characters into an integer value
;
;Input: A - ASCII hex character to process
;       DE - holds the previous binary hex value (set to zero before first
;
;Output: A = 0 scan OK, A != 0 if error encountered
;        DE = value updated to reflect new character
;
;A, DE affected
;-------------------------------------------------------------------------
HexInput:                               ;
    push    bc                          ;
    cp      #0x30                       ;check to see if less than '0'
```

```
    jr      c,HIInvalid             ;*abort if invalid character
    cp      #0x3a                   ;check to see if greater than '9'
    jr      nc,HINotDigit           ;if (this is a digit 0-9)
    sub     #0x30                   ;  adjust digit offset
    jr      HIAdjust                ;
HINotDigit:                         ;else
    cp      #0x41                   ;  check to see if between '9' and
    jr      c,HIInvalid             ;  *abort if invalid
    cp      #0x47                   ;  check to see if less than 'F'+1
    jr      nc,HINotUpper           ;  if (this is an upper case hex d
    sub     #0x41-10                ;    adjust digit offset
    jr      HIAdjust                ;
HINotUpper:                         ;  else
    cp      #0x61                   ;    check to see if between 'F' a
    jr      c,HIInvalid             ;    *abort if invalid
    cp      #0x67                   ;    check to see if less than 'f'
    jr      nc,HIInvalid            ;    if (this is a lower case hex
    sub     #0x61-10                ;      adjust digit offset
                                    ;    endif
                                    ;  endif
HIAdjust:                           ;end
                                    ;now adjust DE value for new nibbl
    rlca                            ;
    rlca                            ;
    rlca                            ;
    rlca                            ;scanned nibble now in d7:d4 of A
                                    ;
    ld      b,#4                    ;setup to shift 4 bits
HIALoop:                            ;repeat
    rlca                            ;  shift next bit of scanned nibbl
    rl      e                       ;  shift C flag into LSb of E, MSb
    rl      d                       ;  shift C flag into LSb of D, MSb
    djnz    HIALoop                 ;until (scanned nibble rotated int
    ld      a,#00                   ;indicate success
    jr      HIDone                  ;
HIInvalid:                          ;if (invalid characters)
    ld      a,#01                   ;  indicate error
HIDone:                             ;endif
    pop     bc                      ;
    ret


;-----------------------------------------------------------------------
;SkipWhite
;
```

```
;Skip over whitespace in a string
;
;Input: HL - ptr to null terminated string
;
;Output:HL - adjusted to next non-whitespace character
;       A - 0 = end of string encountered, !0 = field found
;
;A, HL affected
;-------------------------------------------------------------------------
SkipWhite:                              ;Repeat
    ld      a,(hl)                      ;  get next character
    or      a                           ;  check for end of field null ter
    jr      z,SWDone                    ;  if (!end of field)
    cp      #0x20                       ;    check for space
    jr      z,SWNext                    ;    *skip over whitespace
    cp      #0x09                       ;    check for tab
    jr      z,SWNext                    ;    *skip over whitespace
    jr      SWDone                      ;    *abort if non-whitespace foun
SWNext:                                 ;
    inc     hl                          ;    increment pointer
    jr      SkipWhite                   ;
                                        ;  endif
SWDone:                                 ;Until (end of field found OR non-
    ret
```