/Volumes/DongyunLee/ESE280 Lab/Lab8/task1.asm

```
;
; dog_lcd_test_avr128.asm
;
; Created: 10/9/2023 2:14:29 PM
; Author : kshort
;


;*********************************************************************
;**********       BASIC DOG LCD TEST PROGRAM          **********
;*********************************************************************
;
;DOG_LCD_BasicTest.asm
;  Simple test application to verify DOG LCD is properly
;  wired.  This test writes simple test messages to each
;  line of the display.
;
;Version - 2.0 For DOGM163W LCD operated at 3.3V
;

     .CSEG

     ; interrupt vector table, with several 'safety' stubs
     rjmp RESET      ;Reset/Cold start vector
     reti            ;External Intr0 vector
     reti            ;External Intr1 vector



;*************************************************************************
;************* M A I N   A P P L I C A T I O N   C O D E  *************
;*************************************************************************

RESET:

    sbi VPORTA_DIR, 7       ; set PA7 = output.
    sbi VPORTA_OUT, 7       ; set /SS of DOG LCD = 1 (Deselected)

    rcall init_lcd_dog    ; init display, using SPI serial interface
    rcall clr_dsp_buffs   ; clear all three SRAM memory buffer lines

   rcall update_lcd_dog     ;display data in memory buffer on LCD

   rcall test_lcd

    ;breakpoint followin instr. to see blanked LCD and messages in buffer
   rcall update_lcd_dog     ;breakpoint here to see blanked LCD

   // after 4 seconds
   rcall 4s_delay

   rcall clr_dsp_buffs   ; clear all three SRAM memory buffer lines

   rcall update_lcd_dog     ;display data in memory buffer on LCD

    end_loop:       ;infinite loop, program's task is complete
    rjmp end_loop
```

```
58
59
60
61
62
63  ;---------------------------- SUBROUTINES ----------------------------
64
65
66  ;===================================
67  .include "lcd_dog_asm_driver_avr128.inc"  ; LCD DOG init/update procedures.
68  ;===================================
69
70
71  ;************************
72  ;NAME:      clr_dsp_buffs
73  ;FUNCTION:  Initializes dsp_buffers 1, 2, and 3 with blanks (0x20)
74  ;ASSUMES:   Three CONTIGUOUS 16-byte dram based buffers named
75  ;           dsp_buff_1, dsp_buff_2, dsp_buff_3.
76  ;RETURNS:   nothing.
77  ;MODIFIES:  r25,r26, Z-ptr
78  ;CALLS:     none
79  ;CALLED BY: main application and diagnostics
80  ;*********************************************************************
81  clr_dsp_buffs:
82       ldi R25, 48               ; load total length of both buffer.
83       ldi R26, ' '              ; load blank/space into R26.
84       ldi ZH, high (dsp_buff_1) ; Load ZH and ZL as a pointer to 1st
85       ldi ZL, low (dsp_buff_1)  ; byte of buffer for line 1.
86
87      ;set DDRAM address to 1st position of first line.
88  store_bytes:
89       st  Z+, R26       ; store ' ' into 1st/next buffer byte and
90                         ; auto inc ptr to next location.
91       dec  R25          ;
92       brne store_bytes  ; cont until r25=0, all bytes written.
93       ret
94
95
96  ;*********************************************************************
97  ; test_lcd
98
99  test_lcd:
100     ldi XH, high (dsp_buff_1) ; Load ZH and ZL as a pointer to 1st
101     ldi XL, low (dsp_buff_1)  ; byte of buffer for line 1.
102     ldi r16, 0x30
103     ldi r17, 48
104
105     loop:
106         st X+, r16
107         inc r16
108
109         cpi r16, 0x39
110         breq jump_ascii
111
112         cpi r16, 0x7A
113         breq jump_ascii_2
114
115         dec r17
116         brne loop
117         ret
```

```
118
119     jump_ascii:
120         ldi r16, 0x61
121         rjmp loop
122
123     jump_ascii_2:
124         ldi r16, 0x41
125         rjmp loop
126
127
128
129
130 4s_delay:
131     ldi r22, 160  ; Set R22 to introduce a delay of ~160 * 30uS = 4.8ms
132     ldi r23, 250  ; Set R23 to repeat the above delay 250 times for ~4 seconds
133
134      4s_delay_loop:
135           rcall v_delay  ; Call the v_delay subroutine with the specified delay
136           dec r23        ; Decrement the outer loop counter
137           brne 4s_delay_loop  ; Continue the loop until r23 reaches zero
138           ret
139
140 ;********************
141 ;NAME:      load_msg
142 ;FUNCTION:  Loads a predefined string msg into a specified diplay
143 ;           buffer.
144 ;ASSUMES:   Z = offset of message to be loaded. Msg format is
145 ;           defined below.
146 ;RETURNS:   nothing.
147 ;MODIFIES:  r16, Y, Z
148 ;CALLS:     nothing
149 ;CALLED BY:
150 ;*************************************************************************
151 ; Message structure:
152 ;   label:  .db <buff num>, <text string/message>, <end of string>
153 ;
154 ; Message examples (also see Messages at the end of this file/module):
155 ;   msg_1: .db 1,"First Message ", 0   ; loads msg into buff 1, eom=0
156 ;   msg_2: .db 1,"Another message ", 0 ; loads msg into buff 1, eom=0
157 ;
158 ; Notes:
159 ;   a) The 1st number indicates which buffer to load (either 1, 2, or 3).
160 ;   b) The last number (zero) is an 'end of string' indicator.
161 ;   c) Y = ptr to disp_buffer
162 ;      Z = ptr to message (passed to subroutine)
163 ;*************************************************************************
164 load_msg:
165      ldi YH, high (dsp_buff_1) ; Load YH and YL as a pointer to 1st
166      ldi YL, low (dsp_buff_1)  ; byte of dsp_buff_1 (Note - assuming
167                                ; (dsp_buff_1 for now).
168      lpm R16, Z+               ; get dsply buff number (1st byte of msg).
169      cpi r16, 1                ; if equal to '1', ptr already setup.
170      breq get_msg_byte         ; jump and start message load.
171      adiw YH:YL, 16            ; else set ptr to dsp buff 2.
172      cpi r16, 2                ; if equal to '2', ptr now setup.
173      breq get_msg_byte         ; jump and start message load.
174      adiw YH:YL, 16            ; else set ptr to dsp buff 2.
175
176 get_msg_byte:
177      lpm R16, Z+               ; get next byte of msg and see if '0'.
```

```
178       cpi R16, 0                ; if equal to '0', end of message reached.
179       breq msg_loaded           ; jump and stop message loading operation.
180       st Y+, R16                ; else, store next byte of msg in buffer.
181       rjmp get_msg_byte         ; jump back and continue...
182  msg_loaded:
183       ret
184
185  ;***** END OF FILE ******
186
```