/Volumes/DongyunLee/ESE280 Lab/Lab8/task2.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

    rcall 2s_delay

    rcall shifting

   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  ;---------------------------- SUBROUTINES ----------------------------
63
64
65  ;===================================
66  .include "lcd_dog_asm_driver_avr128.inc"  ; LCD DOG init/update procedures.
67  ;===================================
68
69
70  ;************************
71  ;NAME:      clr_dsp_buffs
72  ;FUNCTION:  Initializes dsp_buffers 1, 2, and 3 with blanks (0x20)
73  ;ASSUMES:   Three CONTIGUOUS 16-byte dram based buffers named
74  ;           dsp_buff_1, dsp_buff_2, dsp_buff_3.
75  ;RETURNS:   nothing.
76  ;MODIFIES:  r25,r26, Z-ptr
77  ;CALLS:     none
78  ;CALLED BY: main application and diagnostics
79  ;*********************************************************************
80  clr_dsp_buffs:
81       ldi R25, 48               ; load total length of both buffer.
82       ldi R26, ' '              ; load blank/space into R26.
83       ldi ZH, high (dsp_buff_1) ; Load ZH and ZL as a pointer to 1st
84       ldi ZL, low (dsp_buff_1)  ; byte of buffer for line 1.
85
86      ;set DDRAM address to 1st position of first line.
87  store_bytes:
88       st  Z+, R26       ; store ' ' into 1st/next buffer byte and
89                         ; auto inc ptr to next location.
90       dec  R25          ;
91       brne store_bytes  ; cont until r25=0, all bytes written.
92       ret
93
94
95  ;*********************************************************************
96  ; test_lcd
97
98  test_lcd:
99      ldi XH, high (dsp_buff_1) ; Load ZH and ZL as a pointer to 1st
100     ldi XL, low (dsp_buff_1)  ; byte of buffer for line 1.
101     ldi r16, 0x30
102     ldi r17, 48
103
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 ; shifting subroutine
131
132 shifting:
133
134     ldi XH, high (dsp_buff_1) ; Load ZH and ZL as a pointer to 1st
135     ldi XL, low (dsp_buff_1)  ; byte of buffer for line 1.
136     ldi r20, 0x30   //r16 is zero 0
137     ldi r19, 48
138     ldi r21, 48
139
140     loop_outside:
141
142         loop_shifting:
143             ld r16, X
144             ; adiw XH:XL, $0001     // increament the pointer but it is done br
    the next line
145             ld r17, X+
146
147             sdiw XH:XL, $0001   ; decrement the pointer
148
149             st X+, r17
150
151             dec r19
152             brne push_zero
153             rjmp loop_shifting
154
155         push_zero:
156             st X, 0x20
157
158     rcall update_lcd_dog
159     rcall 2s_delay
160
161     ldi XH, high (dsp_buff_1) ; Load ZH and ZL as a pointer to 1st
162     ldi XL, low (dsp_buff_1)  ; byte of buffer for line 1.
163
164     dec r21
165     brne loop_outside
166     ret
167
168
169
170
171 2s_delay:
172     ldi r22, 160  ; Set R22 to introduce a delay of ~160 * 30uS = 4.8ms
173     ldi r23, 125  ; Set R23 to repeat the above delay 250 times for ~2 seconds
174
175      2s_delay_loop:
176           rcall v_delay  ; Call the v_delay subroutine with the specified delay
```

```
177           dec r23         ; Decrement the outer loop counter
178           brne 2s_delay_loop  ; Continue the loop until r23 reaches zero
179           ret
180
181 ;********************
182 ;NAME:      load_msg
183 ;FUNCTION:  Loads a predefined string msg into a specified diplay
184 ;           buffer.
185 ;ASSUMES:   Z = offset of message to be loaded. Msg format is
186 ;           defined below.
187 ;RETURNS:   nothing.
188 ;MODIFIES:  r16, Y, Z
189 ;CALLS:     nothing
190 ;CALLED BY:
191 ;*************************************************************************
192 ; Message structure:
193 ;   label:  .db <buff num>, <text string/message>, <end of string>
194 ;
195 ; Message examples (also see Messages at the end of this file/module):
196 ;   msg_1: .db 1,"First Message ", 0   ; loads msg into buff 1, eom=0
197 ;   msg_2: .db 1,"Another message ", 0 ; loads msg into buff 1, eom=0
198 ;
199 ; Notes:
200 ;   a) The 1st number indicates which buffer to load (either 1, 2, or 3).
201 ;   b) The last number (zero) is an 'end of string' indicator.
202 ;   c) Y = ptr to disp_buffer
203 ;      Z = ptr to message (passed to subroutine)
204 ;*************************************************************************
205 load_msg:
206      ldi YH, high (dsp_buff_1) ; Load YH and YL as a pointer to 1st
207      ldi YL, low (dsp_buff_1)  ; byte of dsp_buff_1 (Note - assuming
208                                ; (dsp_buff_1 for now).
209      lpm R16, Z+               ; get dsply buff number (1st byte of msg).
210      cpi r16, 1                ; if equal to '1', ptr already setup.
211      breq get_msg_byte         ; jump and start message load.
212      adiw YH:YL, 16            ; else set ptr to dsp buff 2.
213      cpi r16, 2                ; if equal to '2', ptr now setup.
214      breq get_msg_byte         ; jump and start message load.
215      adiw YH:YL, 16            ; else set ptr to dsp buff 2.
216
217 get_msg_byte:
218      lpm R16, Z+               ; get next byte of msg and see if '0'.
219      cpi R16, 0                ; if equal to '0', end of message reached.
220      breq msg_loaded           ; jump and stop message loading operation.
221      st Y+, R16                ; else, store next byte of msg in buffer.
222      rjmp get_msg_byte         ; jump back and continue...
223 msg_loaded:
224      ret
225
226 ;***** END OF FILE ******
227
```