```
1  ;***************************************************************************
2  ;*
3  ;* Title: subroutine_based_display.asm
4  ;* Author: Jason Chen
5  ;* Version: 1
6  ;* Last updated: 10/25/2022
7  ;* Target: AVR128DB48
8  ;*
9  ;* DESCRIPTION
10 ;*          Design Task 4:
11 ;*          This program polls the flag associated with the pushbutton. This flag
12 ;*          is connected to PE0. If the flag is set, the contents of the array
13 ;*          bcd_entries is shifted left and the BCD digit set on the least
14 ;*          significant 4 bits of PORTC_IN are stored in the least significant
15 ;*          byte of the bcd_entries array. Then the corresponding segment values
16 ;*          for each digit in bcd_entries display are written into the
     led_display.
17 ;*
18 ;*          Note: entry of a non-BCD value is ignored.
19 ;*
20 ;* This program also continually multiplexes the display so that the digits
21 ;* entered are constantly seen on the display. Before any digits are
22 ;* entered the display displays 0000.
23 ;*
24 ;* VERSION HISTORY
25 ;* 1.0 Original version
26 ;***************************************************************************
27
28 .dseg
29 bcd_entries: .byte 4
30 led_display: .byte 4
31 digit_num: .byte 1
32
33 .cseg
34 initialize:
35     ldi r16, 0xFF           ; load r16 with all 1s
36     out VPORTD_DIR, r16     ; VPORTD - all pins configured as output
37     ldi r16, 0xF0
38     out VPORTA_DIR, r16     ; VPORTA - pins 4 - 7 configured as output
39     ldi r16, 0x00
40     out VPORTC_DIR, r16     ; VPORTC - all pins configured as input
41     cbi VPORTE_DIR, 0       ; PE0 configured as input
42     sbi VPORTE_DIR, 1       ; PE1 configured as output
43     sbi VPORTE_OUT, 1       ; PE1 is 1, ensure flip flop is uncleared
44
45 enable_pullups_inven:
46     ldi XH, HIGH(PORTC_PIN0CTRL)    ; X points to PORTC_PIN0CTRL
47     ldi XL, LOW(PORTC_PIN0CTRL)
48     ldi r17, 8                      ; loop control variable, 8 step counter
49
50 pin_config:             ; configures PORTC_PINnCTRL
51     ld r16, X           ; load value of PORTC_PINnCTRL
```

```
52      ori r16, 0x88       ; enable input bits invert and pullup resistors
53      st X+, r16          ; store results at PORTC_PINnCTRL address
54      dec r17             ; decrement lcv
55      brne pin_config     ; repeats 7 times
56
57  main_loop:
58      rcall multiplex_display
59      rcall mux_digit_delay
60      rcall poll_digit_entry
61      rjmp main_loop
62
63
64
65
66
67  ;***********************************************************************
68  ;*
69  ;* "multiplex_display" - Multiplex the Four Digit LED Display
70  ;*
71  ;* DESCRIPTION
72  ;*          Updates a single digit of the display and increments the
73  ;*          digit_num to the digit position to be displayed next.
74  ;* Author:          Jason Chen
75  ;* Version:         1
76  ;* Last Updated:    10/24/2022
77  ;* Target:          AVR128DB48
78  ;* Number of words:
79  ;* Number of cycles:
80  ;* Low registers modified:  none
81  ;* High registers modified: none
82  ;*
83  ;* Parameters:
84  ;*      led_display: a four byte array that holds the segment values
85  ;*          for each digit of the display. led_display[0] holds the
86  ;*          segment patter for digit 0 (the rightmost digit) and so on.
87  ;*
88  ;*      digit_num: byte variable, the least significant two bits are the
89  ;*          index of the last digit displayed.
90  ;*
91  ;* Returns: Outputs segment pattern and turns on digit driver for the next
92  ;*          position in the display to be turned ON.
93  ;*
94  ;* Notes:   The segments are controlled by PORTD - (dp, a through g), the
95  ;*          digit drivers are controlled by PORTA (PA7 - PA4, digit 0 - 3).
96  ;***********************************************************************
97
98  multiplex_display:
99      push r16        ; push contents of r16 - r18 to stack so they are
100     push r17        ; undisturbed
101     push r18
102
103     ldi r16, 0xFF
```

```
104     out VPORTD_OUT, r16     ; turn all segments OFF
105     in r16, VPORTA_OUT      ; get current value of VPORTA
106     ori r16, 0xF0
107     out VPORTA_OUT, r16     ; turn all digits OFF
108
109     ldi XH, HIGH(led_display)   ; X points to start of led_display array
110     ldi XL, LOW(led_display)
111
112     lds r16, digit_num      ; get current display number
113     inc r16
114     andi r16, 0x03          ; mask for two least significant bits
115     sts digit_num, r16      ; store next digit to be displayed
116
117     add XL, r16         ; add digit number to offset to array pointer
118     brcc PC + 2         ; if no carry, skip next instruction
119     inc XH
120
121     ld r17, X
122     out VPORTD_OUT, r17     ; output to segment display driver port
123     in r17, VPORTA_OUT      ; get current digit driver port value
124     ldi r18, 0x10           ; for next PORTA value via bit shift
125
126     digit_pos:
127         cpi r16, 0          ; if digit number is 0, use pattern in r18
128         breq digit_on
129         lsl r18             ; r18 shifted left if not 0
130         dec r16             ; decrement digit number offset
131         rjmp digit_pos
132
133     digit_on:
134         eor r17, r18        ; complement digit driver position
135         out VPORTA_OUT, r17 ; turn selected digit ON
136
137     pop r18         ; repopulate r16 - r18 with original contents from stack
138     pop r17
139     pop r16
140     ret
141
142
143
144
145
146 ;***************************************************************************
147 ;*
148 ;* "poll_digit_entry" - Polls Pushbutton for Conditional Digit Entry
149 ;*
150 ;* DESCRIPTION:
151 ;*          Polls the flag associated with the pushbutton. This flag is
152 ;*          connected to PE0. If the flag is set, the contents of the array
153 ;*          bcd_entries is shifted left and the BCD digit set on the least
154 ;*          significant 4 bits of PORTC_IN are stored in the least significant
155 ;*          byte of the bcd_entries array. Then the corresponding segment
```

```
156 ;*          segment values for each digit in the bcd_entries display are
157 ;*          written into the led_display. Note: entry of a non=BCD value must
158 ;*          be ignored.
159 ;*
160 ;* Author:          Jason Chen
161 ;* Version:         1
162 ;* Last updated:    10/25/2022
163 ;* Target:          AVR128DB48
164 ;* Number of words:     44
165 ;* Number of cycles:    134
166 ;* Low registers modified:  none
167 ;* High registers modified: none
168 ;*
169 ;* Parameters:
170 ;*      bcd_entries: a four byte array that holds a series of binary
171 ;*          represented decimals.
172 ;*      led_display: a four byte array that holds the bit pattern to turn ON
173 ;*          the segments dp, a-g to represent the corresponding decimal of
174 ;*          bcd_entries array.
175 ;*
176 ;* Returns: Outputs the led_display array containing the bit pattern to be
177 ;*          displayed associated to the digit position
178 ;*
179 ;* Notes:
180 ;*********************************************************************
181
182 poll_digit_entry:
183     sbis VPORTE_IN, 0       ; check if the button has been pressed
184     ret                     ; returns to caller if not pressed
185
186     cbi VPORTE_OUT, 1       ; clear the flip flop
187     sbi VPORTE_OUT, 1       ; unclear the flip flop
188
189     push r16        ; depopulate registers by pushing to stack
190     push r17
191     push r18
192
193     ldi XH, HIGH(bcd_entries)   ; X points to bcd_entries[0]
194     ldi XL, LOW(bcd_entries)
195
196     in r16, VPORTC_IN       ; store input bits from VPORTC_IN in r16
197     rcall reverse_bits      ; reverse bits for 90 degree board rotation
198     ldi r17, 4              ; loop control variable, 4 step counter
199
200     left_shift_digits:
201         ld r18, X               ; save contents of bcd_entries[i]
202         st X+, r16              ; assign r16 into bcd_entries[i], i++
203         mov r16, r18            ; preparing r16 for next step in loop
204         dec r17                 ; decrement lcv
205         brne left_shift_digits  ; repeats 3 times
206
207     ldi XH, HIGH(bcd_entries)   ; X points to bcd_entries[0]
```

```
208     ldi XL, LOW(bcd_entries)
209     ldi YH, HIGH(led_display)   ; Y points to led_display[0]
210     ldi YL, LOW(led_display)
211     ldi r17, 4                  ; loop control variable
212
213     load_bit_pattern:
214         ld r18, X+          ; load r18 with bcd_entries[i], i++
215         rcall hex_to_7seg   ; convert binary into segment bit pattern
216         st Y+, r18          ; store bit pattern in led_display[j], j++
217         dec r17
218         brne load_bit_pattern   ; repeats 3 times
219
220     pop r18     ; repopulate r16 - r18 with original contents from stack
221     pop r17
222     pop r16
223     ret
224
225
226
227
228
229 ;***********************************************************************
230 ;*
231 ;* "hex_to_7seg" - Hexadecimal to Seven Segment Conversion
232 ;*
233 ;* Description:
234 ;*          Converts a right justified hexadecimal digit to the seven
235 ;*          segment pattern required to display it. Pattern is right
236 ;*          justified a through g. Pattern uses 0s to turn segments on ON.
237 ;*
238 ;* Author:          Ken Short
239 ;* Version:         0.1
240 ;* Last updated:    10/03/2022
241 ;* Target:          AVR128DB48
242 ;* Number of words:     1
243 ;* Number of cycles:    1
244 ;* Low registers modified: none
245 ;* High registers modified: r16, r18
246 ;*
247 ;* Parameters: r18: hex digit to be converted
248 ;* Returns: r18: seven segment pattern. 0 turns segment ON
249 ;*
250 ;* Notes:
251 ;*
252 ;***********************************************************************
253
254 hex_to_7seg:
255     ldi ZH, HIGH(hextable * 2)  ; set Z to point to start of table
256     ldi ZL, LOW(hextable * 2)
257
258     ldi r16, $00        ; add offset to Z pointer
259     andi r18, 0x0F      ; mask for low nibble
```

```
260     add ZL, r18
261     adc ZH, r16
262     lpm r18, Z          ; load byte from table pointed to by Z
263     ret
264
265 ; Table of segment values to display digits 0 - F
266 ; dp, a - g
267 hextable: .db $01, $4F, $12, $06, $4C, $24, $20, $0F, $00, $04, $08, $60, $31,
      $42, $30, $38
268 ; dp, g - a
269 ;hextable: .db $40, $79, $24, $30, $19, $12, $02, $78, $00, $10, $08, $03, $46,
      $21, $06, $0E
270
271
272
273
274
275 ;***************************************************************************
276 ;*
277 ;* "reverse_bits" - Reverse Bit Order in a Register
278 ;*
279 ;* Description:
280 ;*          Reverse the order of bits register 17, which reads the input
281 ;*          switches, into register 18.
282 ;*
283 ;* Author:          Jason Chen
284 ;* Version:         1
285 ;* Last updated:    10/13/2022
286 ;* Target:          AVR128DB48
287 ;* Number of words:
288 ;* Number of cycles:    8
289 ;* Low registers modified: none
290 ;* High registers modified: r16
291 ;*
292 ;* Parameters: r16 containing original bit order
293 ;*
294 ;* Returns: r16 containing reversed reversed bits
295 ;*
296 ;* Notes:
297 ;*
298 ;***************************************************************************
299
300 reverse_bits:
301     push r17        ; write contents of r17 and r18 to stack
302     push r18
303
304     ldi r18, 0x00
305     ldi r17, 0x08   ; 8 step counter
306
307     bits_loop:
308         lsl r16         ; left shift r16, original register
309         ror r18         ; rotate right r18, reversed register
```

```
310         dec r17
311         cpi r17, 0x00   ; ----- probably can delete
312         brne bits_loop  ; repeats 7 times
313         mov r16, r18    ; copy bit pattern into r16
314
315     pop r18         ; retrieve original contents of r17 and r18 from stack
316     pop r17
317     ret
318
319
320
321
322
323 ;***************************************************************************
324 ;*
325 ;* "mux_digit_delay" - Multiplex Digit Delay / Variable Delay
326 ;*
327 ;* Description: Delays r16 * 1ms (approx.)
328 ;*
329 ;* Author:          Jason Chen
330 ;* Version:         1
331 ;* Last updated:    10/13/2022
332 ;* Target:          AVR128DB48
333 ;* Number of words:     11
334 ;* Number of cycles:
335 ;* Low registers modified:  none
336 ;* High registers modified: none
337 ;*
338 ;* Parameters:
339 ;*
340 ;* Returns:
341 ;*
342 ;* Notes:
343 ;*
344 ;***************************************************************************
345
346 mux_digit_delay:
347     push r16        ; write contents of r16 and r17 to stack
348     push r17
349
350     ldi r16, 1      ; outer loop control variable
351
352     outer_loop:
353         ldi r17, 133    ; inner loop control variable
354
355     inner_loop:
356         dec r17
357         brne inner_loop
358         dec r16
359         brne outer_loop
360
361     pop r17         ; retrieve original contents of r16 and r17 from stack
```

```
362     pop r16
363     ret
```