

```

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      ↗
17 ;*      led_display.
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 clear_arrays:
46     ldi r16, 0x00          ; load r16 with all 0s
47     ldi r17, 4              ; loop control variable
48     ldi XH, HIGH(bcd_entries)
49     ldi XL, LOW(bcd_entries)
50     ldi YH, HIGH(led_display)
51     ldi YL, LOW(led_display)

```

```

52     clear_entries:
53         st X+, r16            ; set bcd_entries[i] = 0, i++
54         dec r17
55         brne clear_entries    ; repeats 3 times
56     ldi r17, 4
57     clear_display:
58         ld r18, X+            ; load r18 with bcd_entries[i], i++
59         rcall hex_to_7seg      ; convert binary into segment bit pattern
60         st Y+, r18            ; store bit pattern in led_display[j], j++
61         dec r17
62         brne clear_display    ; repeats 3 times
63
64     enable_pullups_inven:
65         ldi XH, HIGH(PORTC_PIN0CTRL) ; X points to PORTC_PIN0CTRL
66         ldi XL, LOW(PORTC_PIN0CTRL)
67         ldi r17, 8            ; loop control variable, 8 step counter
68
69     pin_config:                ; configures PORTC_PINnCTRL
70         ld r16, X              ; load value of PORTC_PINnCTRL
71         ori r16, 0x88          ; enable input bits invert and pullup resistors
72         st X+, r16             ; store results at PORTC_PINnCTRL address
73         dec r17                ; decrement lcv
74         brne pin_config        ; repeats 7 times
75
76     main_loop:
77         rcall multiplex_display
78         rcall mux_digit_delay
79         rcall poll_digit_entry
80         rjmp main_loop
81
82
83
84
85
86 ;*****
87 ;*
88 ;* "multiplex_display" - Multiplex the Four Digit LED Display
89 ;*
90 ;* DESCRIPTION
91 ;*     Updates a single digit of the display and increments the
92 ;*     digit_num to the digit position to be displayed next.
93 ;* Author:      Jason Chen
94 ;* Version:     1
95 ;* Last Updated: 10/24/2022
96 ;* Target:      AVR128DB48
97 ;* Number of words:
98 ;* Number of cycles:
99 ;* Low registers modified: none
100 ;* High registers modified: none
101 ;*
102 ;* Parameters:
103 ;*     led_display: a four byte array that holds the segment values

```

```

104 ;*      for each digit of the display. led_display[0] holds the
105 ;*      segment patten for digit 0 (the rightmost digit) and so on.
106 ;*
107 ;*      digit_num: byte variable, the least significant two bits are the
108 ;*      index of the last digit displayed.
109 ;*
110 ;* Returns: Outputs segment pattern and turns on digit driver for the next
111 ;*      position in the display to be turned ON.
112 ;*
113 ;* Notes:  The segments are controlled by PORTD - (dp, a through g), the
114 ;*      digit drivers are controlled by PORTA (PA7 - PA4, digit 0 - 3).
115 ;*****
116
117 multiplex_display:
118     push r16      ; push contents of r16 - r18 to stack so they are
119     push r17      ; undisturbed
120     push r18
121
122     ldi r16, 0xFF
123     out VPORTD_OUT, r16    ; turn all segments OFF
124     in r16, VPORTA_OUT     ; get current value of VPORTA
125     ori r16, 0xF0
126     out VPORTA_OUT, r16    ; turn all digits OFF
127
128     ldi XH, HIGH(led_display) ; X points to start of led_display array
129     ldi XL, LOW(led_display)
130
131     lds r16, digit_num      ; get current display number
132     inc r16
133     andi r16, 0x03         ; mask for two least significant bits
134     sts digit_num, r16     ; store next digit to be displayed
135
136     add XL, r16            ; add digit number to offset to array pointer
137     brcc PC + 2           ; if no carry, skip next instruction
138     inc XH
139
140     ld r17, X
141     out VPORTD_OUT, r17    ; output to segment display driver port
142     in r17, VPORTA_OUT     ; get current digit driver port value
143     ldi r18, 0x10         ; for next PORTA value via bit shift
144
145     digit_pos:
146         cpi r16, 0        ; if digit number is 0, use pattern in r18
147         breq digit_on
148         lsl r18           ; r18 shifted left if not 0
149         dec r16           ; decrement digit number offset
150         rjmp digit_pos
151
152     digit_on:
153         eor r17, r18      ; complement digit driver position
154         out VPORTA_OUT, r17 ; turn selected digit ON
155

```

```

156     pop r18             ; repopulate r16 - r18 with original contents from stack
157     pop r17
158     pop r16
159     ret
160
161
162
163
164
165 ;*****
166 ;*
167 ;* "poll_digit_entry" - Polls Pushbutton for Conditional Digit Entry
168 ;*
169 ;* DESCRIPTION:
170 ;*     Polls the flag associated with the pushbutton. This flag is
171 ;*     connected to PE0. If the flag is set, the contents of the array
172 ;*     bcd_entries is shifted left and the BCD digit set on the least
173 ;*     significant 4 bits of PORTC_IN are stored in the least significant
174 ;*     byte of the bcd_entries array. Then the corresponding segment
175 ;*     segment values for each digit in the bcd_entries display are
176 ;*     written into the led_display. Note: entry of a non=BCD value must
177 ;*     be ignored.
178 ;*
179 ;* Author:             Jason Chen
180 ;* Version:            1
181 ;* Last updated:       10/25/2022
182 ;* Target:             AVR128DB48
183 ;* Number of words:    44
184 ;* Number of cycles:    134
185 ;* Low registers modified: none
186 ;* High registers modified: none
187 ;*
188 ;* Parameters:
189 ;*     bcd_entries: a four byte array that holds a series of binary
190 ;*     represented decimals.
191 ;*     led_display: a four byte array that holds the bit pattern to turn ON
192 ;*     the segments dp, a-g to represent the corresponding decimal of
193 ;*     bcd_entries array.
194 ;*
195 ;* Returns: Outputs the led_display array containing the bit pattern to be
196 ;*     displayed associated to the digit position
197 ;*
198 ;* Notes:
199 ;*****
200
201 poll_digit_entry:
202     sbis VPORTE_IN, 0      ; check if the button has been pressed
203     ret                   ; returns to caller if not pressed
204
205     cbi VPORTE_OUT, 1      ; clear the flip flop
206     sbi VPORTE_OUT, 1      ; unclear the flip flop
207

```

```

208     push r16          ; depopulate registers by pushing to stack
209     push r17
210     push r18
211
212     ldi XH, HIGH(bcd_entries) ; X points to bcd_entries[0]
213     ldi XL, LOW(bcd_entries)
214
215     in r16, VPORTC_IN
216     rcall reverse_bits      ; reverse bits for 90 degree board rotation
217
218     mov r18, r16
219     ldi r17, 6
220     add r16, r17
221     brhc PC + 2
222     rjmp repopulate
223     mov r16, r18
224
225     ldi r17, 4            ; loop control variable, 4 step counter
226
227     left_shift_digits:
228         ld r18, X          ; save contents of bcd_entries[i]
229         st X+, r16          ; assign r16 into bcd_entries[i], i++
230         mov r16, r18        ; preparing r16 for next step in loop
231         dec r17             ; decrement lcv
232         brne left_shift_digits ; repeats 3 times
233
234     ldi XH, HIGH(bcd_entries) ; X points to bcd_entries[0]
235     ldi XL, LOW(bcd_entries)
236     ldi YH, HIGH(led_display) ; Y points to led_display[0]
237     ldi YL, LOW(led_display)
238     ldi r17, 4            ; loop control variable
239
240     load_bit_pattern:
241         ld r18, X+          ; load r18 with bcd_entries[i], i++
242         rcall hex_to_7seg    ; convert binary into segment bit pattern
243         st Y+, r18          ; store bit pattern in led_display[j], j++
244         dec r17
245         brne load_bit_pattern ; repeats 3 times
246
247     repopulate:
248         pop r18             ; repopulate r16 - r18 with original contents from stack
249         pop r17
250         pop r16
251         ret
252
253
254
255
256
257 ;*****
258 ;*
259 ;* "hex_to_7seg" - Hexadecimal to Seven Segment Conversion

```

```

260 ;*
261 ;* Description:
262 ;*           Converts a right justified hexadecimal digit to the seven
263 ;*           segment pattern required to display it. Pattern is right
264 ;*           justified a through g. Pattern uses 0s to turn segments on ON.
265 ;*
266 ;* Author:           Ken Short
267 ;* Version:          0.1
268 ;* Last updated:     10/03/2022
269 ;* Target:           AVR128DB48
270 ;* Number of words:   1
271 ;* Number of cycles:  1
272 ;* Low registers modified: none
273 ;* High registers modified: r16, r18
274 ;*
275 ;* Parameters: r18: hex digit to be converted
276 ;* Returns: r18: seven segment pattern. 0 turns segment ON
277 ;*
278 ;* Notes:
279 ;*
280 ;*****
281
282 hex_to_7seg:
283     ldi ZH, HIGH(hextable * 2) ; set Z to point to start of table
284     ldi ZL, LOW(hextable * 2)
285
286     ldi r16, $00 ; add offset to Z pointer
287     andi r18, 0x0F ; mask for low nibble
288     add ZL, r18
289     adc ZH, r16
290     lpm r18, Z ; load byte from table pointed to by Z
291     ret
292
293 ; Table of segment values to display digits 0 - F
294 ; dp, a - g
295 hextable: .db $01, $4F, $12, $06, $4C, $24, $20, $0F, $00, $04;, $08, $60, $31, ➤
296           $42, $30, $38
297 ; dp, g - a
298 ;hextable: .db $40, $79, $24, $30, $19, $12, $02, $78, $00, $10
299
300
301
302
303 ;*****
304 ;*
305 ;* "reverse_bits" - Reverse Bit Order in a Register
306 ;*
307 ;* Description:
308 ;*           Reverse the order of bits register 17, which reads the input
309 ;*           switches, into register 18.
310 ;*

```

```

311 ;* Author:          Jason Chen
312 ;* Version:         1
313 ;* Last updated:    10/13/2022
314 ;* Target:          AVR128DB48
315 ;* Number of words:
316 ;* Number of cycles: 8
317 ;* Low registers modified: none
318 ;* High registers modified: r16
319 ;*
320 ;* Parameters: r16 containing original bit order
321 ;*
322 ;* Returns: r16 containing reversed reversed bits
323 ;*
324 ;* Notes:
325 ;*
326 ;*****
327
328 reverse_bits:
329     push r17          ; write contents of r17 and r18 to stack
330     push r18
331
332     ldi r18, 0x00
333     ldi r17, 0x08     ; 8 step counter
334
335     bits_loop:
336         lsl r16          ; left shift r16, original register
337         ror r18          ; rotate right r18, reversed register
338         dec r17
339         cpi r17, 0x00    ; ----- probably can delete
340         brne bits_loop  ; repeats 7 times
341         mov r16, r18     ; copy bit pattern into r16
342
343     pop r18            ; retrieve original contents of r17 and r18 from stack
344     pop r17
345     ret
346
347
348
349
350
351 ;*****
352 ;*
353 ;* "mux_digit_delay" - Multiplex Digit Delay / Variable Delay
354 ;*
355 ;* Description: Delays r16 * 1ms (approx.)
356 ;*
357 ;* Author:          Jason Chen
358 ;* Version:         1
359 ;* Last updated:    10/13/2022
360 ;* Target:          AVR128DB48
361 ;* Number of words: 11
362 ;* Number of cycles:

```

```
363  ;* Low registers modified:  none
364  ;* High registers modified: none
365  ;*
366  ;* Parameters:
367  ;*
368  ;* Returns:
369  ;*
370  ;* Notes:
371  ;*
372  ;*****
373
374  mux_digit_delay:
375      push r16          ; write contents of r16 and r17 to stack
376      push r17
377
378      ldi r16, 1        ; outer loop control variable
379
380      outer_loop:
381          ldi r17, 133   ; inner loop control variable
382
383          inner_loop:
384              dec r17
385              brne inner_loop
386              dec r16
387              brne outer_loop
388
389      pop r17           ; retrieve original contents of r16 and r17 from stack
390      pop r16
391      ret
```