

```

1  ;*****
2  ;*
3  ;* Title: subroutine_isr_based_display.asm
4  ;* Author: Jason Chen
5  ;* Version: 1
6  ;* Last updated: 11/08/2022
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 PORTE. 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                                ; start of data segment
29 bcd_entries: .byte 4
30 led_display: .byte 4
31 digit_num: .byte 1
32
33 .cseg                                ; start of code segment
34 reset:
35     jmp start
36
37 .org PORTE_PORT_vect
38     jmp porte_isr                    ;vector for all PORTE pin change IRQs
39
40 start:
41 ; Configure I/O ports
42     ldi r16, 0xFF
43     out VPORTD_DIR, r16              ; VPORTD - all pins configured as output
44     ldi r16, 0xF0
45     out VPORTA_DIR, r16              ; PA4 - PA7 configured as output (gate of pnp      ↗
46                                     transistor)
47     ldi r16, 0x00
48     out VPORTC_DIR, r16              ; VPORTC - all pins configured as input
49     cbi VPORTE_DIR, 0                ; PE0 configured as input
50 ; Enable pullup resistors and inven for PORTC

```

```

51 /* ldi r16, 0x88          ; inven = bit 7, pullup_enable = bit 3 (1000
    1000)
52     sts PORTC_PINCONFIG, r16    ; write PINCONFIG
53     ldi r17, 0xFF              ; specify which PINnCTRL registers to update
    (all)
54     sts PORTC_PINCTRLUPD, r17   ; update specified PINnCTRL registers
    simultaneously*/
55
56 ; Configure interrupt request
57     lds r16, PORTE_PIN0CTRL      ; set ISC for PE0 to rising edge
58     ori r16, 0x02                ; ISC = bit 1 for rising edge
59     sts PORTE_PIN0CTRL, r16     ; update PIN0CTRL register (0000 0010)
60     sei
61
62 ; Set pointers for arrays
63     ldi XH, HIGH(bcd_entries)
64     ldi XL, LOW(bcd_entries)    ; X points to bcd_entries[0]
65     ldi YH, HIGH(led_display)
66     ldi YL, LOW(led_display)    ; Y points to led_display[0]
67
68 ; Clear arrays
69     ldi r16, 0                  ; load r16 with 0
70     mov r18, r16
71     rcall hex_to_7seg           ; load r18 with 7 segment bit pattern to show 0
72     ldi r17, 4                  ; loop control variable
73     clear_entries:
74         st X+, r16
75         st Y+, r18
76         dec r17
77         brne clear_entries
78
79 ; Program loop
80 main_loop:
81     rcall multiplex_display
82     rcall mux_digit_delay
83     nop
84     rjmp main_loop
85
86
87
88
89
90 porte_isr:
91     cli
92     push r16
93     in r16, CPU_SREG
94     push r16
95     rcall poll_PE0
96     pop r16
97     out CPU_SREG, r16
98     pop r16
99     sei

```

```

100     reti
101
102
103
104
105
106 ;*****
107 ;*
108 ;* "poll_PE0" - Poll PE0 for IRQ
109 ;*
110 ;* Description:
111 ;*     Checks PE0 for interrupt request. If IRQ is met, the request is
112 ;*     cleared and then polls digit entry.
113 ;*
114 ;* Author:      Jason
115 ;* Version:     1
116 ;* Last updated: 11/7/2022
117 ;* Target:      AVR128DB48
118 ;* Number of words:
119 ;* Number of cycles:
120 ;* Low registers modified: none
121 ;* High registers modified: r16
122 ;*
123 ;* Parameters:  PE0 is checked for flag.
124 ;* Returns:     PE0's flag is cleared.
125 ;*
126 ;* Notes:
127 ;*
128 ;*****
129 poll_PE0:
130     lds r16, PORTE_INTFLAGS    ; Determine if PE0's INTFLAG is set
131     sbrc r16, 0                ; Check if PE0 IRQ flag is set
132     ret                        ; return to caller (main_loop) if not set
133     rcall clear_irq
134     rcall digit_entry
135     ret
136
137
138
139
140
141 ;*****
142 ;*
143 ;* "clear_irq" - Clear Interrupt request
144 ;*
145 ;* Description:
146 ;*     Clears PORTE_INTFLAG register.
147 ;*
148 ;* Author:      Jason
149 ;* Version:     1
150 ;* Last updated: 11/7/2022
151 ;* Target:      AVR128DB48

```

```

152 ;* Number of words:
153 ;* Number of cycles:
154 ;* Low registers modified: none
155 ;* High registers modified: none
156 ;*
157 ;* Parameters: PE0_INTFLAG to be cleared
158 ;* Returns:    PE0_INTFLAG is cleared.
159 ;*
160 ;* Notes:
161 ;*
162 ;*****
163 clear_irq:
164     ldi r16, PORT_INT0_bm
165     sts PORTE_INTFLAGS, r16
166     ret
167
168
169
170
171
172 ;*****
173 ;*
174 ;* "poll_digit_entry" - Polls Pushbutton for Conditional Digit Entry
175 ;*
176 ;* DESCRIPTION:
177 ;*     Polls the flag associated with the pushbutton. This flag is
178 ;*     connected to PE0. If the flag is set, the contents of the array
179 ;*     bcd_entries is shifted left and the BCD digit set on the least
180 ;*     significant 4 bits of PORTC_IN are stored in the least significant
181 ;*     byte of the bcd_entries array. Then the corresponding segment
182 ;*     segment values for each digit in the bcd_entries display are
183 ;*     written into the led_display. Note: entry of a non=BCD value must
184 ;*     be ignored.
185 ;*
186 ;* Author:      Jason Chen
187 ;* Version:     1
188 ;* Last updated: 10/25/2022
189 ;* Target:      AVR128DB48
190 ;* Number of words: 44
191 ;* Number of cycles: 134
192 ;* Low registers modified: none
193 ;* High registers modified: none
194 ;*
195 ;* Parameters:
196 ;*     bcd_entries: a four byte array that holds a series of binary
197 ;*     represented decimals.
198 ;*     led_display: a four byte array that holds the bit pattern to turn ON
199 ;*     the segments dp, a-g to represent the corresponding decimal of
200 ;*     bcd_entries array.
201 ;*
202 ;* Returns: Outputs the led_display array containing the bit pattern to be
203 ;*     displayed associated to the digit position

```

```

204 ;*
205 ;* Notes:
206 ;*****
207 digit_entry:
208     push r16          ; depopulate registers by pushing to stack
209     push r17
210     push r18
211     push XH
212     push XL
213     push YH
214     push YL
215
216     in r16, VPORTC_IN
217     rcall reverse_bits      ; reverse bits for 90 degree board rotation
218     cpi r16, 10             ; check if r16 >= 10
219     brge repopulate
220
221     ldi XH, HIGH(bcd_entries) ; X points to bcd_entries[0]
222     ldi XL, LOW(bcd_entries)
223     ldi YH, HIGH(led_display) ; Y points to led_display[0]
224     ldi YL, LOW(led_display)
225     mov r18, r16
226     ldi r17, 4              ; loop control variable, 4 step counter
227
228     left_shift_arrays:
229         ld r16, X            ; save contents of bcd_entries[i]
230         st X+, r18            ; bcd_entries[i] = r18, i++
231         rcall hex_to_7seg
232         st Y+, r18            ; led_display[i] = r18, i++
233         mov r18, r16          ; preparing r18 for next step in loop
234         dec r17               ; decrement lcv
235         brne left_shift_arrays ; repeats 3 times
236
237     repopulate:              ; repopulate all registers with
238         pop YL                ; original contents from stack
239         pop YH
240         pop XL
241         pop XH
242         pop r18
243         pop r17
244         pop r16
245     ret
246
247
248
249
250
251 ;*****
252 ;*
253 ;* "multiplex_display" - Multiplex the Four Digit LED Display
254 ;*
255 ;* DESCRIPTION

```

```

256 ;*      Updates a single digit of the display and increments the
257 ;*      digit_num to the digit position to be displayed next.
258 ;* Author:      Jason Chen
259 ;* Version:      1
260 ;* Last Updated: 10/24/2022
261 ;* Target:       AVR128DB48
262 ;* Number of words:
263 ;* Number of cycles:
264 ;* Low registers modified: none
265 ;* High registers modified: none
266 ;*
267 ;* Parameters:
268 ;*      led_display: a four byte array that holds the segment values
269 ;*      for each digit of the display. led_display[0] holds the
270 ;*      segment patten for digit 0 (the rightmost digit) and so on.
271 ;*
272 ;*      digit_num: byte variable, the least significant two bits are the
273 ;*      index of the last digit displayed.
274 ;*
275 ;* Returns: Outputs segment pattern and turns on digit driver for the next
276 ;*      position in the display to be turned ON.
277 ;*
278 ;* Notes:  The segments are controlled by PORTD - (dp, a through g), the
279 ;*      digit drivers are controlled by PORTA (PA7 - PA4, digit 0 - 3).
280 ;*****
281 multiplex_display:
282     push r16          ; depopulate registers by pushing to stack
283     push r17
284     push r18
285
286     ldi r16, 0xFF
287     out VPORTD_OUT, r16    ; turn all segments OFF
288     in r16, VPORTA_OUT     ; get current value of VPORTA
289     ori r16, 0xF0
290     out VPORTA_OUT, r16    ; turn all digits OFF
291
292     ldi XH, HIGH(led_display) ; X points to start of led_display array
293     ldi XL, LOW(led_display)
294
295     lds r16, digit_num      ; get current display number
296     inc r16
297     andi r16, 0x03          ; mask for two least significant bits
298     sts digit_num, r16      ; store next digit to be displayed
299
300     add XL, r16             ; add digit number to offset to array pointer
301     brcc PC + 2             ; if no carry, skip next instruction
302     inc XH
303
304     ld r17, X
305     out VPORTD_OUT, r17     ; output to segment display driver port
306     in r17, VPORTA_OUT      ; get current digit driver port value
307     ldi r18, 0x10          ; for next PORTA value via bit shift

```

```

308
309     digit_pos:
310         cpi r16, 0           ; if digit number is 0, use pattern in r18
311         breq digit_on
312         lsl r18              ; r18 shifted left if not 0
313         dec r16              ; decrement digit number offset
314         rjmp digit_pos
315
316     digit_on:
317         eor r17, r18         ; complement digit driver position
318         out VPORТА_OUT, r17 ; turn selected digit ON
319
320     pop r18                  ; repopulate all registers with
321     pop r17                  ; original contents from stack
322     pop r16
323     ret
324
325
326
327
328
329 ;*****
330 ;*
331 ;* "reverse_bits" - Reverse Bit Order in a Register
332 ;*
333 ;* Description:
334 ;*     Reverse the order of bits register 17, which reads the input
335 ;*     switches, into register 18.
336 ;*
337 ;* Author:         Jason Chen
338 ;* Version:        1
339 ;* Last updated:   10/13/2022
340 ;* Target:         AVR128DB48
341 ;* Number of words:
342 ;* Number of cycles: 8
343 ;* Low registers modified: none
344 ;* High registers modified: r16
345 ;*
346 ;* Parameters: r16 containing original bit order
347 ;*
348 ;* Returns: r16 containing reversed reversed bits
349 ;*
350 ;* Notes:
351 ;*
352 ;*****
353 reverse_bits:
354     push r17                ; write contents of r17 and r18 to stack
355     push r18
356
357     ldi r18, 0x00
358     ldi r17, 0x08          ; 8 step counter
359

```

```

360     bits_loop:
361         lsl r16            ; left shift r16, original register
362         ror r18            ; rotate right r18, reversed register
363         dec r17
364         cpi r17, 0x00      ; ----- probably can delete
365         brne bits_loop    ; repeats 7 times
366         mov r16, r18      ; copy bit pattern into r16
367
368         pop r18            ; retrieve original contents of r17 and r18 from stack
369         pop r17
370         ret
371
372
373
374
375
376 ;*****
377 ;*
378 ;* "mux_digit_delay" - Multiplex Digit Delay / Variable Delay
379 ;*
380 ;* Description: Delays r16 * 1ms (approx.)
381 ;*
382 ;* Author:          Jason Chen
383 ;* Version:         1
384 ;* Last updated:    10/13/2022
385 ;* Target:          AVR128DB48
386 ;* Number of words: 11
387 ;* Number of cycles:
388 ;* Low registers modified: none
389 ;* High registers modified: none
390 ;*
391 ;* Parameters:
392 ;*
393 ;* Returns:
394 ;*
395 ;* Notes:
396 ;*
397 ;*****
398 mux_digit_delay:
399     push r16              ; write contents of r16 and r17 to stack
400     push r17
401
402     ldi r16, 1            ; outer loop control variable
403
404     outer_loop:
405         ldi r17, 133      ; inner loop control variable
406
407     inner_loop:
408         dec r17
409         brne inner_loop
410         dec r16
411         brne outer_loop

```



```

412
413     pop r17          ; retrieve original contents of r16 and r17 from stack
414     pop r16
415     ret
416
417
418
419
420 ;*****
421 ;*
422 ;* "hex_to_7seg" - Hexadecimal to Seven Segment Conversion
423 ;*
424 ;* Description:
425 ;*     Converts a right justified hexadecimal digit to the seven
426 ;*     segment pattern required to display it. Pattern is right
427 ;*     justified a through g. Pattern uses 0s to turn segments on ON.
428 ;*
429 ;* Author:          Ken Short
430 ;* Version:         0.1
431 ;* Last updated:    10/03/2022
432 ;* Target:          AVR128DB48
433 ;* Number of words: 1
434 ;* Number of cycles: 1
435 ;* Low registers modified: none
436 ;* High registers modified: r18
437 ;*
438 ;* Parameters: r18: hex digit to be converted
439 ;* Returns: r18: seven segment pattern. 0 turns segment ON
440 ;*
441 ;* Notes:
442 ;*
443 ;*****
444 hex_to_7seg:
445     push r16
446     ldi ZH, HIGH(hextable * 2) ; set Z to point to start of table
447     ldi ZL, LOW(hextable * 2)
448
449     ldi r16, $00          ; add offset to Z pointer
450     andi r18, 0x0F        ; mask for low nibble
451     add ZL, r18
452     adc ZH, r16
453     lpm r18, Z            ; load byte from table pointed to by Z
454     pop r16
455     ret
456
457 ; Table of segment values to display digits 0 - F
458 ; dp, a - g
459 hextable: .db $01, $4F, $12, $06, $4C, $24, $20, $0F, $00, $04;, $08, $60, $31, ➤
460           $42, $30, $38
461 ; dp, g - a
462 ;hextable: .db $40, $79, $24, $30, $19, $12, $02, $78, $00, $10

```