```
;***********************************************************************
;*
;* Title: interrupt_driven_display.asm
;* Author: Jason Chen
;* Version: 1
;* Last updated: 11/13/2022
;* Target: AVR128DB48
;*
;* DESCRIPTION
;*          Design Task 2:
;*          This program polls the flag associated with the pushbutton. This flag
;*          is connected to PORTE. If the flag is set, the contents of the array
;*          bcd_entries is shifted left and the BCD digit set on the least
;*          significant 4 bits of PORTC_IN are stored in the least significant
;*          byte of the bcd_entries array. Then the corresponding segment values
;*          for each digit in bcd_entries display are written into led_display.
;*
;*          Note: entry of a non-BCD value is ignored and the program is entirely
;*          interrupt driven, main loop does not call any subroutines.
;*
;* This program also continually multiplexes the display so that the digits
;* entered are constantly seen on the display. Before any digits are
;* entered the display displays 0000.
;*
;* VERSION HISTORY
;* 1.0 Original version
;***********************************************************************

.equ PERIOD = 390            ; 389.625 to be exact for 40.0Hz

.dseg                        ; start of data segment
bcd_entries: .byte 4
led_display: .byte 4
digit_num: .byte 1

.cseg                        ; start of code segment
reset:
    jmp start

.org TCA0_OVF_vect
    jmp ovf_mux_isr          ; vector for overflow IRQ

.org PORTE_PORT_vect
    jmp porte_isr            ; vector for all PORTE pin change IRQs

start:
; Configure I/O ports
    ldi r16, 0xFF
    out VPORTD_DIR, r16          ; VPORTD - all pins configured as output
    ldi r16, 0xF0
    out VPORTA_DIR, r16          ; PA4 - PA7 configured as output (gate of pnp
      transistor)
```

```
52      ldi r16, 0x00
53      out VPORTC_DIR, r16         ; VPORTC - all pins configured as input
54      cbi VPORTE_DIR, 0           ; PE0 configured as input
55
56  ; Enable pullup resistors and inven for PORTC
57      ldi r16, 0x88               ; inven = bit 7, pullup_enable = bit 3 (1000
          1000)
58      sts PORTC_PINCONFIG, r16    ; write PINCONFIG
59      ldi r17, 0xFF               ; specify which PINnCTRL registers to update
          (all)
60      sts PORTC_PINCTRLUPD, r17   ; update specified PINnCTRL registers
          simultaneously
61
62  ; Configure TCA0
63      ldi r16, TCA_SINGLE_WGMODE_NORMAL_gc    ; WGMODE normal
64      sts TCA0_SINGLE_CTRLB, r16
65
66      ldi r16, TCA_SINGLE_OVF_bm              ; enable overflow interrupt
67      sts TCA0_SINGLE_INTCTRL, r16
68
69      ldi r16, LOW(PERIOD)                    ; set the period
70      sts TCA0_SINGLE_PER, r16
71      ldi r16, HIGH(PERIOD)
72      sts TCA0_SINGLE_PER + 1, r16
73
74      ldi r16, TCA_SINGLE_CLKSEL_DIV256_gc | TCA_SINGLE_ENABLE_bm ; set clock and
          start timer
75      sts TCA0_SINGLE_CTRLA, r16
76
77  ; Configure interrupt request
78      lds r16, PORTE_PIN0CTRL     ; set ISC for PE0 to rising edge
79      ori r16, 0x02               ; ISC = bit 1 for rising edge
80      sts PORTE_PIN0CTRL, r16     ; update PIN0CTRL register (0000 0010)
81      sei
82
83  ; Set pointers for arrays
84      ldi XH, HIGH(bcd_entries)
85      ldi XL, LOW(bcd_entries)    ; X points to bcd_entries[0]
86      ldi YH, HIGH(led_display)
87      ldi YL, LOW(led_display)    ; Y points to led_display[0]
88
89  ; Clear arrays
90      ldi r16, 0                  ; load r16 with 0
91      mov r18, r16
92      rcall hex_to_7seg           ; load r18 with 7 segment bit pattern to show 0
93      ldi r17, 4                  ; loop control variable
94      clear_entries:
95          st X+, r16
96          st Y+, r18
97          dec r17
98          brne clear_entries
99
```

```
100 ; Program loop
101 main_loop:
102     rjmp main_loop
103 
104 ; Interrupt subroutines
105 ovf_mux_isr:
106     push r16                        ; save registers
107     in r16, CPU_SREG
108     push r16
109     rcall multiplex_display         ; multiplex display
110     ldi r16, TCA_SINGLE_OVF_bm      ; clear OVF flag
111     sts TCA0_SINGLE_INTFLAGS, r16
112     pop r16                         ; restor registers
113     out CPU_SREG, r16
114     pop r16
115     reti
116 
117 porte_isr:
118     cli                     ; clear interrupt
119     push r16                ; save registers
120     in r16, CPU_SREG
121     push r16
122     rcall poll_PE0          ; poll PE0
123     pop r16                 ; restore registers
124     out CPU_SREG, r16
125     pop r16
126     sei                     ; enable interrupt
127     reti
128 
129 
130 
131 
132 
133 ;***************************************************************************
134 ;*
135 ;* "poll_PE0" - Poll PE0 for IRQ
136 ;*
137 ;* Description:
138 ;*          Checks PE0 for interrupt request. If IRQ is met, the request is
139 ;*          cleared and then polls digit entry.
140 ;*
141 ;* Author:          Jason
142 ;* Version:         1
143 ;* Last updated:    11/7/2022
144 ;* Target:          AVR128DB48
145 ;* Number of words:
146 ;* Number of cycles:
147 ;* Low registers modified: none
148 ;* High registers modified: r16
149 ;*
150 ;* Parameters:  PE0 is checked for flag.
151 ;* Returns:     PE0's flag is cleared.
```

```
152 ;*
153 ;* Notes:
154 ;*
155 ;***************************************************************************
156 poll_PE0:
157     lds r16, PORTE_INTFLAGS     ; Determine if PE0's INTFLAG is set
158     sbrs r16, 0                 ; Check if PE0 IRQ flag is set
159     ret                         ; return to caller (main_loop) if not set
160     rcall clear_irq
161     rcall digit_entry
162     ret
163
164
165
166
167
168 ;***************************************************************************
169 ;*
170 ;* "clear_irq" - Clear Interrupt request
171 ;*
172 ;* Description:
173 ;*          Clears PORTE_INTFLAG register.
174 ;*
175 ;* Author:          Jason
176 ;* Version:         1
177 ;* Last updated:    11/7/2022
178 ;* Target:          AVR128DB48
179 ;* Number of words:
180 ;* Number of cycles:
181 ;* Low registers modified: none
182 ;* High registers modified: none
183 ;*
184 ;* Parameters:  PE0_INTFLAG to be cleared
185 ;* Returns:     PE0_INTFLAG is cleared.
186 ;*
187 ;* Notes:
188 ;*
189 ;***************************************************************************
190 clear_irq:
191     ldi r16, PORT_INT0_bm
192     sts PORTE_INTFLAGS, r16
193     ret
194
195
196
197
198
199 ;***************************************************************************
200 ;*
201 ;* "poll_digit_entry" - Polls Pushbutton for Conditional Digit Entry
202 ;*
203 ;* DESCRIPTION:
```

```
204 ;*          Polls the flag associated with the pushbutton. This flag is
205 ;*          connected to PE0. If the flag is set, the contents of the array
206 ;*          bcd_entries is shifted left and the BCD digit set on the least
207 ;*          significant 4 bits of PORTC_IN are stored in the least significant
208 ;*          byte of the bcd_entries array. Then the corresponding segment
209 ;*          segment values for each digit in the bcd_entries display are
210 ;*          written into the led_display. Note: entry of a non=BCD value must
211 ;*          be ignored.
212 ;*
213 ;* Author:          Jason Chen
214 ;* Version:         1
215 ;* Last updated:    10/25/2022
216 ;* Target:          AVR128DB48
217 ;* Number of words:     44
218 ;* Number of cycles:    134
219 ;* Low registers modified:  none
220 ;* High registers modified: none
221 ;*
222 ;* Parameters:
223 ;*      bcd_entries: a four byte array that holds a series of binary
224 ;*          represented decimals.
225 ;*      led_display: a four byte array that holds the bit pattern to turn ON
226 ;*          the segments dp, a-g to represent the corresponding decimal of
227 ;*          bcd_entries array.
228 ;*
229 ;* Returns: Outputs the led_display array containing the bit pattern to be
230 ;*          displayed associated to the digit position
231 ;*
232 ;* Notes:
233 ;*********************************************************************
234 digit_entry:
235     push r16        ; depopulate registers by pushing to stack
236     push r17
237     push r18
238     push XH
239     push XL
240     push YH
241     push YL
242
243     in r16, VPORTC_IN
244     rcall reverse_bits          ; reverse bits for 90 degree board rotation
245     cpi r16, 10                 ; check if r16 >= 10
246     brge repopulate
247
248     ldi XH, HIGH(bcd_entries)   ; X points to bcd_entries[0]
249     ldi XL, LOW(bcd_entries)
250     ldi YH, HIGH(led_display)   ; Y points to led_display[0]
251     ldi YL, LOW(led_display)
252     mov r18, r16
253     ldi r17, 4                  ; loop control variable, 4 step counter
254
255     left_shift_arrays:
```

```
256         ld r16, X               ; save contents of bcd_entries[i]
257         st X+, r18              ; bcd_entries[i] = r18, i++
258         rcall hex_to_7seg
259         st Y+, r18              ; led_display[i] = r18, i++
260         mov r18, r16            ; preparing r18 for next step in loop
261         dec r17                 ; decrement lcv
262         brne left_shift_arrays  ; repeats 3 times
263
264     repopulate:         ; repopulate all registers with
265         pop YL          ; original contents from stack
266         pop YH
267         pop XL
268         pop XH
269         pop r18
270         pop r17
271         pop r16
272     ret
273
274
275
276
277
278 ;***********************************************************************
279 ;*
280 ;* "multiplex_display" - Multiplex the Four Digit LED Display
281 ;*
282 ;* DESCRIPTION
283 ;*          Updates a single digit of the display and increments the
284 ;*          digit_num to the digit position to be displayed next.
285 ;* Author:          Jason Chen
286 ;* Version:         1
287 ;* Last Updated:    10/24/2022
288 ;* Target:          AVR128DB48
289 ;* Number of words:
290 ;* Number of cycles:
291 ;* Low registers modified:  none
292 ;* High registers modified: none
293 ;*
294 ;* Parameters:
295 ;*      led_display: a four byte array that holds the segment values
296 ;*          for each digit of the display. led_display[0] holds the
297 ;*          segment patter for digit 0 (the rightmost digit) and so on.
298 ;*
299 ;*      digit_num: byte variable, the least significant two bits are the
300 ;*          index of the last digit displayed.
301 ;*
302 ;* Returns: Outputs segment pattern and turns on digit driver for the next
303 ;*          position in the display to be turned ON.
304 ;*
305 ;* Notes:   The segments are controlled by PORTD - (dp, a through g), the
306 ;*          digit drivers are controlled by PORTA (PA7 - PA4, digit 0 - 3).
307 ;***********************************************************************
```

```
308 multiplex_display:
309     push r16        ; depopulate registers by pushing to stack
310     push r17
311     push r18
312
313     ldi r16, 0xFF
314     out VPORTD_OUT, r16     ; turn all segments OFF
315     in r16, VPORTA_OUT      ; get current value of VPORTA
316     ori r16, 0xF0
317     out VPORTA_OUT, r16     ; turn all digits OFF
318
319     ldi XH, HIGH(led_display)   ; X points to start of led_display array
320     ldi XL, LOW(led_display)
321
322     lds r16, digit_num      ; get current display number
323     inc r16
324     andi r16, 0x03          ; mask for two least significant bits
325     sts digit_num, r16      ; store next digit to be displayed
326
327     add XL, r16             ; add digit number to offset to array pointer
328     brcc PC + 2             ; if no carry, skip next instruction
329     inc XH
330
331     ld r17, X
332     out VPORTD_OUT, r17     ; output to segment display driver port
333     in r17, VPORTA_OUT      ; get current digit driver port value
334     ldi r18, 0x10           ; for next PORTA value via bit shift
335
336     digit_pos:
337         cpi r16, 0          ; if digit number is 0, use pattern in r18
338         breq digit_on
339         lsl r18             ; r18 shifted left if not 0
340         dec r16             ; decrement digit number offset
341         rjmp digit_pos
342
343     digit_on:
344         eor r17, r18        ; complement digit driver position
345         out VPORTA_OUT, r17 ; turn selected digit ON
346
347     pop r18         ; repopulate all registers with
348     pop r17         ; original contents from stack
349     pop r16
350     ret
351
352
353
354
355
356 ;***************************************************************************
357 ;*
358 ;* "reverse_bits" - Reverse Bit Order in a Register
359 ;*
```

```
360 ;* Description:
361 ;*          Reverse the order of bits register 17, which reads the input
362 ;*          switches, into register 18.
363 ;*
364 ;* Author:          Jason Chen
365 ;* Version:         1
366 ;* Last updated:    10/13/2022
367 ;* Target:          AVR128DB48
368 ;* Number of words:
369 ;* Number of cycles:    8
370 ;* Low registers modified: none
371 ;* High registers modified: r16
372 ;*
373 ;* Parameters: r16 containing original bit order
374 ;*
375 ;* Returns: r16 containing reversed reversed bits
376 ;*
377 ;* Notes:
378 ;*
379 ;***************************************************************************
380 reverse_bits:
381     push r17        ; write contents of r17 and r18 to stack
382     push r18
383
384     ldi r18, 0x00
385     ldi r17, 0x08   ; 8 step counter
386
387     bits_loop:
388         lsl r16         ; left shift r16, original register
389         ror r18         ; rotate right r18, reversed register
390         dec r17
391         cpi r17, 0x00   ; ----- probably can delete
392         brne bits_loop  ; repeats 7 times
393         mov r16, r18    ; copy bit pattern into r16
394
395     pop r18         ; retrieve original contents of r17 and r18 from stack
396     pop r17
397     ret
398
399
400
401
402
403 ;***************************************************************************
404 ;*
405 ;* "mux_digit_delay" - Multiplex Digit Delay / Variable Delay
406 ;*
407 ;* Description: Delays r16 * 1ms (approx.)
408 ;*
409 ;* Author:          Jason Chen
410 ;* Version:         1
411 ;* Last updated:    10/13/2022
```

```
412 ;* Target:          AVR128DB48
413 ;* Number of words:     11
414 ;* Number of cycles:
415 ;* Low registers modified:  none
416 ;* High registers modified: none
417 ;*
418 ;* Parameters:
419 ;*
420 ;* Returns:
421 ;*
422 ;* Notes:
423 ;*
424 ;***************************************************************************
425 mux_digit_delay:
426     push r16        ; write contents of r16 and r17 to stack
427     push r17
428
429     ldi r16, 1      ; outer loop control variable
430
431     outer_loop:
432         ldi r17, 133    ; inner loop control variable
433
434     inner_loop:
435         dec r17
436         brne inner_loop
437         dec r16
438         brne outer_loop
439
440     pop r17         ; retrieve original contents of r16 and r17 from stack
441     pop r16
442     ret
443
444
445
446
447 ;***************************************************************************
448 ;*
449 ;* "hex_to_7seg" - Hexadecimal to Seven Segment Conversion
450 ;*
451 ;* Description:
452 ;*          Converts a right justified hexadecimal digit to the seven
453 ;*          segment pattern required to display it. Pattern is right
454 ;*          justified a through g. Pattern uses 0s to turn segments on ON.
455 ;*
456 ;* Author:          Ken Short
457 ;* Version:         0.1
458 ;* Last updated:    10/03/2022
459 ;* Target:          AVR128DB48
460 ;* Number of words:     1
461 ;* Number of cycles:    1
462 ;* Low registers modified: none
463 ;* High registers modified: r18
```

```
464 ;*
465 ;* Parameters: r18: hex digit to be converted
466 ;* Returns: r18: seven segment pattern. 0 turns segment ON
467 ;*
468 ;* Notes:
469 ;*
470 ;***********************************************************************
471 hex_to_7seg:
472     push r16
473     ldi ZH, HIGH(hextable * 2)  ; set Z to point to start of table
474     ldi ZL, LOW(hextable * 2)
475
476     ldi r16, $00        ; add offset to Z pointer
477     andi r18, 0x0F      ; mask for low nibble
478     add ZL, r18
479     adc ZH, r16
480     lpm r18, Z          ; load byte from table pointed to by Z
481     pop r16
482     ret
483
484 ; Table of segment values to display digits 0 - F
485 ; dp, a - g
486 hextable: .db $01, $4F, $12, $06, $4C, $24, $20, $0F, $00, $04;, $08, $60, $31,
      $42, $30, $38
487 ; dp, g - a
488 ;hextable: .db $40, $79, $24, $30, $19, $12, $02, $78, $00, $10
```