

```
1
2 AVRASM ver. 2.2.7 E:\ESE_280\MyDocuments$\Atmel Studio\7.0\lab_9
  \PE0_and_PE2_intrs\PE0_and_PE2_intrs\main.asm Tue Nov 03 19:29:58 2020
3
4 E:\ESE_280\MyDocuments$\Atmel Studio\7.0\lab_9\PE0_and_PE2_intrs
  \PE0_and_PE2_intrs\main.asm(9): Including file 'C:/Program Files (x86)\Atmel
  \Studio\7.0\Packs\atmel\ATmega_DFP\1.3.300\avrasm\inc\m4809def.inc'
5 E:\ESE_280\MyDocuments$\Atmel Studio\7.0\lab_9\PE0_and_PE2_intrs
  \PE0_and_PE2_intrs\main.asm(9): Including file 'C:/Program Files (x86)\Atmel
  \Studio\7.0\Packs\atmel\ATmega_DFP\1.3.300\avrasm\inc\m4809def.inc'
6
7
8 ; PE0_and_PE2_intrs.asm
9 ;
10 ; Created: 10/30/2020 9:44:44 PM
11 ; Author : hp
12 ;
13
14 .list
15
16 .dseg
17 002800 PB1_count: .byte 1 ;pushbutton 1 presses.
18 002801 PB2_count: .byte 1 ;pushbutton 2 presses.
19
20
21 .cseg ;start of code segment
22 reset:
23 000000 940c 0048 jmp start ;reset vector executed
  a power on
24
25 .org PORTE_PORT_vect
26 000046 940c 005c jmp porte_isr ;vector for all PORTE
  pin change IRQs
27
28
29 start:
30 ; Configure I/O ports
31 000048 9880 cbi VPORTE_DIR, 0 ;PE0 input- gets output
  from PB1
32 000049 9882 cbi VPORTE_DIR, 2 ;PE2 input- gets output
  from PB2
33
34 00004a e000 ldi r16, 0x00 ;make initial counts 0
35 00004b 9300 2800 sts PB1_count, r16
36 00004d 9300 2801 sts PB2_count, r16
37
38 ;Configure interrupts
39 00004f 9100 0490 lds r16, PORTE_PIN0CTRL ;set ISC for PE0 to
  pos. edge
```

```

40 000051 6002          ori r16, 0x02          ;set ISC for rising  ↗
    edge
41 000052 9300 0490      sts PORTE_PIN0CTRL, r16
42
43 000054 9100 0492      lds r16, PORTE_PIN2CTRL ;set ISC for PE2 to  ↗
    pos. edge
44 000056 6002          ori r16, 0x02          ;set ISC for rising  ↗
    edge
45 000057 9300 0492      sts PORTE_PIN2CTRL, r16
46
47 000059 9478          sei                    ;enable global interrupts
48
49                      main_loop:          ;main program loop
50 00005a 0000          nop
51 00005b cffe          rjmp main_loop
52
53
54                      ;Interrupt service routine for any PORTE pin  ↗
                                change IRQ
55                      porte_ISR:
56 00005c 930f          push r16              ;save r16 then SREG
57 00005d b70f          in r16, CPU_SREG
58 00005e 930f          push r16
59 00005f 94f8          cli                    ;clear global interrupt  ↗
    enable
60
61                      ;Determine which pins of PORTE have IRQs
62 000060 9100 0489      lds r16, PORTE_INTFLAGS ;check for PE0 IRQ  ↗
    flag set
63 000062 fd00          sbrc r16, 0
64 000063 d008          rcall PB1_sub          ;execute subroutine  ↗
    for PE0
65
66 000064 9100 0489      lds r16, PORTE_INTFLAGS ;check for PE2 IRQ  ↗
    flag set
67 000066 fd02          sbrc r16, 2
68 000067 d00d          rcall PB2_sub          ;execute subroutine  ↗
    for PE2
69
70 000068 910f          pop r16              ;restore SREG then r16
71 000069 bf0f          out CPU_SREG, r16
72 00006a 910f          pop r16
73 00006b 9518          reti                    ;return from PORTE pin  ↗
    change ISR
74
75
76                      ;Subroutines called by porte_ISR
77                      PB1_sub:              ;PE0's task to be done
78 00006c 9100 2800      lds r16, PB1_count      ;get current count  ↗

```

```

    for PB1
79 00006e 9503                inc r16                ;increment count
80 00006f 9300 2800          sts PB1_count, r16    ;store new count
81 000071 e001                ldi r16, PORT_INT0_bm  ;clear IRQ flag for ↗
    PE0
82 000072 9300 0489          sts PORTE_INTFLAGS, r16
83 000074 9508                ret
84
85
86                                PB2_sub:        ;PE2's task to be done
87 000075 9100 2801          lds r16, PB2_count    ;get current count ↗
    for PB2
88 000077 9503                inc r16                ;increment count
89 000078 9300 2801          sts PB2_count, r16    ;store new count
90 00007a e004                ldi r16, PORT_INT2_bm  ;clear IRQ flag for ↗
    PE2
91 00007b 9300 0489          sts PORTE_INTFLAGS, r16
92 00007d 9508                ret
93
94
95
96 RESOURCE USE INFORMATION
97 -----
98
99 Notice:
100 The register and instruction counts are symbol table hit counts,
101 and hence implicitly used resources are not counted, eg, the
102 'lpm' instruction without operands implicitly uses r0 and z,
103 none of which are counted.
104
105 x,y,z are separate entities in the symbol table and are
106 counted separately from r26..r31 here.
107
108 .dseg memory usage only counts static data declared with .byte
109
110 "ATmega4809" register use summary:
111 x : 0 y : 0 z : 0 r0 : 0 r1 : 0 r2 : 0 r3 : 0 r4 : 0
112 r5 : 0 r6 : 0 r7 : 0 r8 : 0 r9 : 0 r10: 0 r11: 0 r12: 0
113 r13: 0 r14: 0 r15: 0 r16: 29 r17: 0 r18: 0 r19: 0 r20: 0
114 r21: 0 r22: 0 r23: 0 r24: 0 r25: 0 r26: 0 r27: 0 r28: 0
115 r29: 0 r30: 0 r31: 0
116 Registers used: 1 out of 35 (2.9%)
117
118 "ATmega4809" instruction use summary:
119 .lds : 0 .sts : 0 adc : 0 add : 0 adiw : 0 and : 0
120 andi : 0 asr : 0 bclr : 0 bld : 0 brbc : 0 brbs : 0
121 brcc : 0 brcs : 0 break : 0 breq : 0 brge : 0 brhc : 0
122 brhs : 0 brid : 0 brie : 0 brlo : 0 brlt : 0 brmi : 0
123 brne : 0 brpl : 0 brsh : 0 brtc : 0 brts : 0 brvc : 0

```

```

124 brvs : 0 bset : 0 bst : 0 call : 0 cbi : 2 cbr : 0
125 clc : 0 clh : 0 cli : 1 cln : 0 clr : 0 cls : 0
126 clt : 0 clv : 0 clz : 0 com : 0 cp : 0 cpc : 0
127 cpi : 0 cpse : 0 dec : 0 des : 0 eor : 0 fmul : 0
128 fmul : 0 fmul : 0 ical : 0 ijmp : 0 in : 1 inc : 2
129 jmp : 2 ld : 0 ldd : 0 ldi : 3 lds : 6 lpm : 0
130 lsl : 0 lsr : 0 mov : 0 movw : 0 mul : 0 muls : 0
131 muls : 0 neg : 0 nop : 1 or : 0 ori : 2 out : 1
132 pop : 2 push : 2 rcall : 2 ret : 2 reti : 1 rjmp : 1
133 rol : 0 ror : 0 sbc : 0 sbci : 0 sbi : 0 sbic : 0
134 sbis : 0 sbiw : 0 sbr : 0 sbrc : 2 sbrs : 0 sec : 0
135 seh : 0 sei : 1 sen : 0 ser : 0 ses : 0 set : 0
136 sev : 0 sez : 0 sleep : 0 spm : 0 st : 0 std : 0
137 sts : 8 sub : 0 subi : 0 swap : 0 tst : 0 wdr : 0

```

138

139 Instructions used: 19 out of 114 (16.7%)

140

141 "ATmega4809" memory use summary [bytes]:

Segment	Begin	End	Code	Data	Used	Size	Use%
[.cseg]	0x000000	0x0000fc	116	0	116	49152	0.2%
[.dseg]	0x002800	0x002802	0	2	2	6144	0.0%
[.eseg]	0x000000	0x000000	0	0	0	256	0.0%

147

148 Assembly complete, 0 errors, 0 warnings

149

```

1  ;*****
2  ;*
3  ;* Title: BCD to Hex
4  ;* Author:  Judah Ben-Eliezer
5  ;* Version: 1.0
6  ;* Last updated:
7  ;* Target:          ;ATmega4809 @3.3MHz
8  ;*
9  ;* DESCRIPTION
10 ;* This program polls the flag associated with pushbutton 1. This flag is
11 ;* connected to PE0. If the flag is set, the contents of the array bcd_entries
12 ;* is shifted left and the BCD digit set on the least significant 4 bits of
13 ;* PORTA_IN are stored in the least significant byte of the bcd_entries array.
14 ;* Then the corresponding segment values for each digit in the bcd_entries
15 ;* display are written into the led_display. Note: entry of a non-BCD value
16 ;* is ignored.
17 ;*
18 ;* This program also continually multiplexes the display so that the digits
19 ;* entered are constantly seen on the display. Before any digits are entered
20 ;* the display displays 0000.
21 ;*
22 ;* This program also polls the flag associated with pushbutton 2. This flag
23 ;* is connected to PE2. If the flag is set, the digits in the bcd_entries
24 ;* array are read and passed to the prewritten subroutine BCD2bin16. This
25 ;* subroutine performs a BCD to binary conversion. The binary result is
26 ;* partitioned into hexadecimal and placed into the array hex_results. The
27 ;* contents of the hex_results array is converted to seven segment values
28 ;* and placed into the led_display array. The multiplexing then causes
29 ;* the hexadecimal equivalent of the BCD value entered to be displayed in
30 ;* hexadecimal.
31 ;*
32 ;* VERSION HISTORY
33 ;* 1.0 Original version
34 ;*****
35
36 .nolist
37 .include "m4809def.inc"
38 .list
39
40 .dseg
41 bcd_entries: .byte 4
42 led_display: .byte 4
43 digit_num: .byte 1
44 hex_results: .byte 4
45
46 .cseg
47 start:
48     cbi VPORTE_DIR, 0
49     ldi r16, $00
50     out VPORTA_DIR, r16
51     com r16
52     out VPORTD_DIR, r16
53     out VPORTC_DIR, r16
54     ldi XH, HIGH(bcd_entries)
55     ldi XL, LOW(bcd_entries)
56     ldi YH, HIGH(led_display)
57     ldi YL, LOW(led_display)
58     com r16
59     st X+, r16
60     inc r16
61     st X+, r16
62     inc r16
63     st X+, r16
64     inc r16
65     st X, r16
66     cbi VPORTE_OUT, 1
67     sbi VPORTE_OUT, 1
68
69 main_loop:

```

```

70     rcall multiplex_display
71     rcall mux_digit_delay
72     rcall poll_digit_entry
73     rcall poll_bcd_hex
74     rjmp main_loop
75
76
77 ;*****
78 ;*
79 ;* "poll_digit_entry" - Polls Pushbutton 1 for Conditional Digit Entry
80 ;*
81 ;* Description:
82 ;* Polls the flag associated with pushbutton 1. This flag is connected to
83 ;* PE0. If the flag is set, the contents of the array bcd_entries is shifted
84 ;* left and the BCD digit set on the least significant 4 bits of PORTA_IN are
85 ;* stored in the least significant byte of the bcd_entries array. Then the
86 ;* corresponding segment values for each digit in the bcd_entries display are
87 ;* written into the led_display. Note: entry of a non-BCD value is ignored.
88 ;* Author:
89 ;* Version:
90 ;* Last updated:
91 ;* Target:
92 ;* Number of words:
93 ;* Number of cycles:
94 ;* Low registers modified:
95 ;* High registers modified:
96 ;*
97 ;* Parameters:
98 ;* Returns:
99 ;*
100 ;* Notes:
101 ;*
102 ;*****
103 poll_digit_entry:
104     ldi XH, HIGH(bcd_entries)
105     ldi XL, LOW(bcd_entries)
106     sbis VPORTE_IN, 0
107     rjmp poll_digit_entry
108     in r16, VPORTA_IN
109     rcall reverse_bits
110     rcall check_for_non_bcd
111     rcall shift_bcd_entries
112     rcall bcd_to_led
113
114 ;*****
115 ;*
116 ;* "poll_bcd_hex" - Polls Pushbutton 2 for Conditional Conversion of BCD to
117 ;* Hex.
118 ;*
119 ;* Description:
120 ;* Polls the flag associated with pushbutton 2. This flag is connected to
121 ;* PE2. If the flag is set, the digits in the bcd_entries array are read
122 ;* and passed to the prewritten subroutine BCD2bin16. This subroutine
123 ;* performs a BCD to binary conversion. The binary result is partitioned
124 ;* into hexadecimal and placed into the array hex_results. The contents of
125 ;* the hex_results array is converted to seven segment values and placed
126 ;* into the led_display array.
127 ;* Author:
128 ;* Version:
129 ;* Last updated:
130 ;* Target:
131 ;* Number of words:
132 ;* Number of cycles:
133 ;* Low registers modified:
134 ;* High registers modified:
135 ;*
136 ;* Parameters:
137 ;* Returns:
138 ;*

```

```

139  ;* Notes:
140  ;*
141  ;*****
142  poll_bcd_hex:
143      sbis VPORTE_IN, 2
144      ret
145      ldi XH, HIGH(bcd_entries)
146      ldi XL, LOW(bcd_entries)
147      ld r18, X+
148      ld r17, X+
149      swap r17
150      ld r19, X+
151      or r17, r19
152      ld r16, X+
153      swap r16
154      ld r19, X+
155      or r16, r19
156      rcall BCD2bin16
157      ldi XH, HIGH(hex_results)
158      ldi XL, LOW(hex_results)
159      ldi r19, $00
160      or r19, r15
161      andi r19, $F0
162      swap r19
163      st X+, r19
164      lds r20, r15
165      lds r21, r14
166      andi r20, $0F
167      st X+, r20
168      ldi r19, $00
169      or r19, r21
170      andi r19, $F0
171      swap r19
172      st X+, r19
173      andi r21, $0F
174      st X, r21
175      ret
176
177
178  ;*****
179  ;*
180  ;* "mux_digit_delay" - title
181  ;*
182  ;* Description: delays 0.1 * r23
183  ;*
184  ;* Author:  Judah Ben-Eliezer
185  ;* Version: 1.0
186  ;* Last updated:
187  ;* Target:
188  ;* Number of words:
189  ;* Number of cycles:
190  ;* Low registers modified:
191  ;* High registers modified:
192  ;*
193  ;* Parameters:
194  ;* Returns:
195  ;*
196  ;* Notes:
197  ;*
198  ;*****
199  mux_digit_delay:
200      ldi r23, $08 ; 0.1 * r23 = delay
201  outer_loop:
202      ldi r24, $06
203  inner_loop:
204      dec r24
205      brne inner_loop
206      dec r23
207      brne outer_loop

```

```

208         ret
209
210
211 ;*****
212 ;*
213 ;* "reverse_bits" - Reverse Bits
214 ;*
215 ;* Description: Reverses the bit positions in a byte passed in. Bit 0
216 ;* becomes bit 7, bit 6 becomes bit 1, and so on.
217 ;*
218 ;* Author:                Judah Ben-Eliezer
219 ;* Version:               1.0
220 ;* Last updated:          101120
221 ;* Target:                ATmega4809
222 ;* Number of words:       8
223 ;* Number of cycles:
224 ;* Low registers modified: r16, r17, r18
225 ;* High registers modified: none
226 ;*
227 ;* Parameters: r16: byte to be reversed.
228 ;* Returns: r16: reversed byte
229 ;*
230 ;* Notes:
231 ;*
232 ;*****
233 reverse_bits:
234     ldi r18, $08
235 loop_8:
236     ror r16
237     rol r17
238     dec r18
239     brne loop_8
240     ret
241
242 check_for_non_bcd:
243     cpi r17, $0A
244     brsh reset
245     ret
246
247 reset:
248     cbi VPORTE_OUT, 1
249     sbi VPORTE_OUT, 1
250     ret
251
252 shift_bcd_entries:
253     ldi r18, $03
254 shift_loop:
255     ldi XH, HIGH(bcd_entries)
256     ldi XL, LOW(bcd_entries)
257     dec r18
258     add XL, r18
259     ld r19, X+
260     st X, r19
261     brne shift_loop
262     ret
263
264 bcd_to_led:
265     ldi XL, LOW(bcd_entries)
266     ldi XH, HIGH(bcd_entries)
267     st X, r17
268     ldi r20, $04
269 conversion_loop:
270     dec r20
271     ldi XH, HIGH(bcd_entries)
272     ldi XL, LOW(bcd_entries)
273     ldi YH, HIGH(led_display)
274     ldi YL, LOW(led_display)
275     add XL, r20
276     add YL, r20

```



```

277     ld r18, X
278     rcall hex_to_7seg
279     st Y, r18
280     cpi r20, $00
281     brne conversion_loop
282     ret
283
284 multiplex_display:
285     ldi r22, $00
286     sts digit_num, r22
287     ldi YL, LOW(led_display)
288     lds r17, digit_num
289     lds r20, digit_num
290     andi r17, $03
291     add XL, r17
292     ld r18, Y
293     ldi r21, $80
294     inc r20
295 loop:
296     lsr r21
297     dec r20
298     brne loop
299     lsl r21
300     com r21
301     out VPORTC_OUT, r21
302     out VPORTD_OUT, r18
303     cbi VPORTE_OUT, 1
304     sbi VPORTE_OUT, 1
305     inc r17
306     sts digit_num, r17
307     ret
308
309 ;*****
310 ;*
311 ;* "BCD2bin16" - BCD to 16-Bit Binary Conversion
312 ;*
313 ;* This subroutine converts a 5-digit packed BCD number represented by
314 ;* 3 bytes (fBCD2:fBCD1:fBCD0) to a 16-bit number (tbinH:tbinL).
315 ;* MSD of the 5-digit number must be placed in the lowermost nibble of fBCD2.
316 ;*
317 ;* Let "abcde" denote the 5-digit number. The conversion is done by
318 ;* computing the formula: 10(10(10(10a+b)+c)+d)+e.
319 ;* The subroutine "mul10a"/"mul10b" does the multiply-and-add operation
320 ;* which is repeated four times during the computation.
321 ;*
322 ;* Number of words :30
323 ;* Number of cycles :108
324 ;* Low registers used :4 (copyL,copyH,mp10L/tbinL,mp10H/tbinH)
325 ;* High registers used :4 (fBCD0,fBCD1,fBCD2,adder)
326 ;*
327 ;*****
328
329 ;***** "mul10a"/"mul10b" Subroutine Register Variables
330
331 .def    copyL    =r12        ;temporary register
332 .def    copyH    =r13        ;temporary register
333 .def    mp10L    =r14        ;Low byte of number to be multiplied by 10
334 .def    mp10H    =r15        ;High byte of number to be multiplied by 10
335 .def    adder    =r19        ;value to add after multiplication
336
337 ;***** Code
338
339 mul10a: ;***** multiplies "mp10H:mp10L" with 10 and adds "adder" high nibble
340     swap    adder
341 mul10b: ;***** multiplies "mp10H:mp10L" with 10 and adds "adder" low nibble
342     mov     copyL,mp10L ;make copy
343     mov     copyH,mp10H
344     lsl     mp10L        ;multiply original by 2
345     rol     mp10H

```

```

346     lsl copyL           ;multiply copy by 2
347     rol copyH
348     lsl copyL           ;multiply copy by 2 (4)
349     rol copyH
350     lsl copyL           ;multiply copy by 2 (8)
351     rol copyH
352     add mp10L,copyL ;add copy to original
353     adc mp10H,copyH
354     andi adder,0x0f ;mask away upper nibble of adder
355     add mp10L,adder ;add lower nibble of adder
356     brcc m10_1         ;if carry not cleared
357     inc mp10H          ; inc high byte
358 m10_1: ret
359
360 ;***** Main Routine Register Variables
361
362 .def     tbinL    =r14           ;Low byte of binary result (same as mp10L)
363 .def     tbinH    =r15           ;High byte of binary result (same as mp10H)
364 .def     fBCD0    =r16           ;BCD value digits 1 and 0
365 .def     fBCD1    =r17           ;BCD value digits 2 and 3
366 .def     fBCD2    =r18           ;BCD value digit 5
367
368 ;***** Code
369
370 BCD2bin16:
371     andi     fBCD2,0x0f ;mask away upper nibble of fBCD2
372     clr mp10H
373     mov mp10L,fBCD2 ;mp10H:mp10L = a
374     mov adder,fBCD1
375     rcall    mul10a      ;mp10H:mp10L = 10a+b
376     mov adder,fBCD1
377     rcall    mul10b      ;mp10H:mp10L = 10(10a+b)+c
378     mov adder,fBCD0
379     rcall    mul10a      ;mp10H:mp10L = 10(10(10a+b)+c)+d
380     mov adder,fBCD0
381     rcall    mul10b      ;mp10H:mp10L = 10(10(10(10a+b)+c)+d)+e
382     ret
383
384
385
386 ;*****
387 ;*
388 ;* "hex_to_7seg" - Hexadecimal to Seven Segment Conversion
389 ;*
390 ;* Description: Converts a right justified hexadecimal digit to the seven
391 ;* segment pattern required to display it. Pattern is right justified a
392 ;* through g. Pattern uses 0s to turn segments on ON.
393 ;*
394 ;* Author: Ken Short
395 ;* Version: 1.0
396 ;* Last updated: 101620
397 ;* Target: ATmega4809
398 ;* Number of words: 8
399 ;* Number of cycles: 13
400 ;* Low registers modified: none
401 ;* High registers modified: r16, r18, ZL, ZH
402 ;*
403 ;* Parameters: r18: right justified hex digit, high nibble 0
404 ;* Returns: r18: segment values a through g right justified
405 ;*
406 ;* Notes:
407 ;*
408 ;*****
409
410 hex_to_7seg:
411     andi r18, 0x0F ;clear ms nibble
412     ldi ZH, HIGH(hextable * 2) ;set Z to point to start of table
413     ldi ZL, LOW(hextable * 2)
414     ldi r16, $00 ;add offset to Z pointer

```

```

415     add ZL, r18
416     adc ZH, r16
417     lpm r18, Z           ;load byte from table pointed to by Z
418     ret
419
420     ;Table of segment values to display digits 0 - F
421     ;!!! seven values must be added - verify all values
422 hextable: .db $01, $4F, $12, $06, $4C, $24, $20, $0F, $00, $04, $08, $60, $31, $42,
    $30, $38

```