

```

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
```