Judah Ben-Eliezer

112352727
11/10//2020

Prelab 10:

Timer/Counter TCA0 Overflow Interrupt

```
 1
 2  AVRASM ver. 2.2.7  E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_10
      \intr_driven_bit_toggle\intr_driven_bit_toggle\main.asm Tue Nov 10 18:21:32
      2020
 3
 4  E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_10\intr_driven_bit_toggle
      \intr_driven_bit_toggle\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_10\intr_driven_bit_toggle
      \intr_driven_bit_toggle\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                                    ; intr_driven_bit_toggle.asm
 9                                    ;
10                                    ; Created: 11/10/2020 4:50:58 PM
11                                    ; Author : Judah Ben-Eliezer
12                                    ;
13
14                                    .list
15
16
17                                    .equ PERIOD_EXAMPLE_VALUE = 325        ;40.06
                        Hz
18
19                                    reset:
20  000000 940c 0010                     jmp start
21
22                                    .org TCA0_OVF_vect
23  00000e 940c 0023                     jmp toggle_pin_ISR
24
25                                    start:
26  000010 9a81                           sbi VPORTE_DIR, 1
27
28                                       ;configure TCA0
29  000011 e000                          ldi r16, TCA_SINGLE_WGMODE_NORMAL_gc
      ;WGMODE normal
30  000012 9300 0a01                     sts TCA0_SINGLE_CTRLB, r16
31
32                                       ;enable overflow interrupt
33  000014 e001                          ldi r16, TCA_SINGLE_OVF_bm
34  000015 9300 0a0a                     sts TCA0_SINGLE_INTCTRL, r16
35
36                                       ;load period low byte then high byte
37  000017 e405                          ldi r16, LOW(PERIOD_EXAMPLE_VALUE)
38  000018 9300 0a26                     sts TCA0_SINGLE_PER, r16
39  00001a e001                          ldi r16, HIGH(PERIOD_EXAMPLE_VALUE)
40  00001b 9300 0a27                     sts TCA0_SINGLE_PER + 1, r16
41
```

```
42                                      ;set clock and start timer
43  00001d e00d                         ldi r16, TCA_SINGLE_CLKSEL_DIV256_gc |
      TCA_SINGLE_ENABLE_bm
44  00001e 9300 0a00                    sts TCA0_SINGLE_CTRLA, r16
45
46  000020 9478                         sei     ;enable global interrupts
47
48                                    main_loop:
49  000021 0000                         nop
50  000022 cffe                         rjmp main_loop
51
52                                    toggle_pin_ISR:
53  000023 930f                         push r16
54  000024 b70f                         in r16, CPU_SREG
55  000025 930f                         push r16
56  000026 931f                         push r17
57
58  000027 e012                         ldi r17, $02    ;toggle PE1
59  000028 b301                         in r16, VPORTE_OUT
60  000029 2701                         eor r16, r17
61  00002a bb01                         out VPORTE_OUT, r16
62
63  00002b e001                         ldi r16, TCA_SINGLE_OVF_bm  ;clear OVF flag
64  00002c 9300 0a0b                    sts TCA0_SINGLE_INTFLAGS, r16
65
66  00002e 911f                         pop r17
67  00002f 910f                         pop r16
68  000030 bf0f                         out CPU_SREG, r16
69  000031 910f                         pop r16
70
71
72
73  RESOURCE USE INFORMATION
74  ------------------------
75
76  Notice:
77  The register and instruction counts are symbol table hit counts,
78  and hence implicitly used resources are not counted, eg, the
79  'lpm' instruction without operands implicitly uses r0 and z,
80  none of which are counted.
81
82  x,y,z are separate entities in the symbol table and are
83  counted separately from r26..r31 here.
84
85  .dseg memory usage only counts static data declared with .byte
86
87  "ATmega4809" register use summary:
88  x  :   0 y  :   0 z  :   0 r0 :   0 r1 :   0 r2 :   0 r3 :   0 r4 :   0
89  r5 :   0 r6 :   0 r7 :   0 r8 :   0 r9 :   0 r10:   0 r11:   0 r12:   0
```

```
 90  r13:   0 r14:   0 r15:   0 r16:  21 r17:   4 r18:   0 r19:   0 r20:   0
 91  r21:   0 r22:   0 r23:   0 r24:   0 r25:   0 r26:   0 r27:   0 r28:   0
 92  r29:   0 r30:   0 r31:   0
 93  Registers used: 2 out of 35 (5.7%)
 94
 95  "ATmega4809" instruction use summary:
 96  .lds  :   0 .sts  :   0 adc   :   0 add   :   0 adiw  :   0 and   :   0
 97  andi  :   0 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
 98  brcc  :   0 brcs  :   0 break :   0 breq  :   0 brge  :   0 brhc  :   0
 99  brhs  :   0 brid  :   0 brie  :   0 brlo  :   0 brlt  :   0 brmi  :   0
100  brne  :   0 brpl  :   0 brsh  :   0 brtc  :   0 brts  :   0 brvc  :   0
101  brvs  :   0 bset  :   0 bst   :   0 call  :   0 cbi   :   0 cbr   :   0
102  clc   :   0 clh   :   0 cli   :   0 cln   :   0 clr   :   0 cls   :   0
103  clt   :   0 clv   :   0 clz   :   0 com   :   0 cp    :   0 cpc   :   0
104  cpi   :   0 cpse  :   0 dec   :   0 des   :   0 eor   :   1 fmul  :   0
105  fmuls :   0 fmulsu:   0 icall :   0 ijmp  :   0 in    :   2 inc   :   0
106  jmp   :   2 ld    :   0 ldd   :   0 ldi   :   7 lds   :   0 lpm   :   0
107  lsl   :   0 lsr   :   0 mov   :   0 movw  :   0 mul   :   0 muls  :   0
108  mulsu :   0 neg   :   0 nop   :   1 or    :   0 ori   :   0 out   :   2
109  pop   :   3 push  :   3 rcall :   0 ret   :   0 reti  :   1 rjmp  :   1
110  rol   :   0 ror   :   0 sbc   :   0 sbci  :   0 sbi   :   1 sbic  :   0
111  sbis  :   0 sbiw  :   0 sbr   :   0 sbrc  :   0 sbrs  :   0 sec   :   0
112  seh   :   0 sei   :   1 sen   :   0 ser   :   0 ses   :   0 set   :   0
113  sev   :   0 sez   :   0 sleep :   0 spm   :   0 st    :   0 std   :   0
114  sts   :   6 sub   :   0 subi  :   0 swap  :   0 tst   :   0 wdr   :   0
115
116  Instructions used: 13 out of 114 (11.4%)
117
118  "ATmega4809" memory use summary [bytes]:
119  Segment   Begin    End      Code   Data   Used    Size   Use%
120  ----------------------------------------------------------------
121  [.cseg] 0x000000 0x000066     78      0     78   49152   0.2%
122  [.dseg] 0x002800 0x002800      0      0      0    6144   0.0%
123  [.eseg] 0x000000 0x000000      0      0      0     256   0.0%
124
125  Assembly complete, 0 errors, 0 warnings
126
```

```
 1
 2  AVRASM ver. 2.2.7  E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_10
      \bcd_to_hex_mux_intr\bcd_to_hex_mux_intr\main.asm Tue Nov 10 18:42:51 2020
 3
 4  E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_10\bcd_to_hex_mux_intr
      \bcd_to_hex_mux_intr\main.asm(37): 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$\Atmel Studio\7.0\lab_10\bcd_to_hex_mux_intr
      \bcd_to_hex_mux_intr\main.asm(417): warning: Register r14 already defined by
      the .DEF directive
 6  E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_10\bcd_to_hex_mux_intr
      \bcd_to_hex_mux_intr\main.asm(418): warning: Register r15 already defined by
      the .DEF directive
 7  E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_10\bcd_to_hex_mux_intr
      \bcd_to_hex_mux_intr\main.asm(37): Including file 'C:/Program Files (x86)
      \Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.3.300\avrasm\inc\m4809def.inc'
 8
 9
10                                  ;*
11                                  ;* Title: bcd_to_hex_mux_intr
12                                  ;* Author:  Judah Ben-Eliezer
13                                  ;* Version:     1.0
14                                  ;* Last updated:    11/10/2020
15                                  ;* Target:          ;ATmega4809 @3.3MHz
16                                  ;*
17                                  ;* DESCRIPTION
18                                  ;* This program polls the flag associated with
                     pushbutton 1. This flag is
19                                  ;* connected to PE0. If the flag is set, the
                    contents of the array bcd_entries
20                                  ;* is shifted left and the BCD digit set on
                    the least significant 4 bits of
21                                  ;* PORTA_IN are stored in the least
                    significant byte of the bcd_entries array.
22                                  ;* Then the corresponding segment values for
                    each digit in the bcd_entries
23                                  ;* display are written into the led_display.
                    Note: entry of a non-BCD value
24                                  ;* is ignored.
25                                  ;*
26                                  ;* This program also continually multiplexes
                    the display so that the digits
27                                  ;* entered are constantly seen on the display.
                     Before any digits are entered
28                                  ;* the display displays 0000.
29                                  ;*
30                                  ;* This program also polls the flag associated
                     with pushbutton 2. This flag
31                                  ;* is connected to PE2. If the flag is set,
```

```
                        the digits in the bcd_entries
32                                  ;* array are read and passed to the prewritten
                         subroutine BCD2bin16. This
33                                  ;* subroutine performs a BCD to binary
                        conversion. The binary result is
34                                  ;* partitioned into hexadecimal and placed
                        into the array hex_results. The
35                                  ;* contents of the hex_results array is
                        converted to seven segment values
36                                  ;* and placed into the led_display array. The
                        multiplexing then causes
37                                  ;* the hexadecimal equivalent of the BCD value
                         entered to be displayed in
38                                  ;* hexadecimal.
39                                  ;*
40                                  ;* VERSION HISTORY
41                                  ;* 1.0 Original version
42                                  ;*****************************************
                        ******************************
43
44                                  .list
45
46                                  .equ PERIOD_EXAMPLE_VALUE = 325
47
48                                  .dseg
49  002800                          bcd_entries: .byte 4
50  002804                          led_display: .byte 4
51  002808                          digit_num: .byte 1
52  002809                          hex_results: .byte 4
53
54                                  .cseg
55
56                                  .org TCA0_OVF_vect
57  00000e 940c 008a                   jmp mux_display_ISR     ;vector for all
      TCA0_OVF pin change IRQs
58
59                                  .org PORTE_PORT_vect
60  000046 940c 007a                   jmp porte_ISR       ;vector for all PORTE
      pin change IRQs
61
62                                  start:
63  000048 9880                        cbi VPORTE_DIR, 0
64  000049 9882                        cbi VPORTE_DIR, 2
65  00004a e000                        ldi r16, $00
66  00004b b900                        out VPORTA_DIR, r16
67  00004c 9500                        com r16
68  00004d b90c                        out VPORTD_DIR, r16
69  00004e b908                        out VPORTC_DIR, r16
70  00004f e2b8                        ldi XH, HIGH(bcd_entries)
```

```
 71  000050 e0a0                        ldi XL, LOW(bcd_entries)
 72  000051 e2d8                        ldi YH, HIGH(led_display)
 73  000052 e0c4                        ldi YL, LOW(led_display)
 74  000053 9500                        com r16
 75  000054 930d                        st X+, r16
 76  000055 930d                        st X+, r16
 77  000056 930d                        st X+, r16
 78  000057 930c                        st X, r16
 79  000058 e0a0                        ldi XL, LOW(bcd_entries)
 80
 81                                     ;Configure interrupts
 82  000059 9100 0490                   lds r16, PORTE_PIN0CTRL ;set ISC for PE0 to
       pos. edge
 83  00005b 6002                        ori r16, 0x02       ;set ISC for rising
      edge
 84  00005c 9300 0490                   sts PORTE_PIN0CTRL, r16
 85
 86  00005e 9100 0492                   lds r16, PORTE_PIN2CTRL ;set ISC for PE2 to
       pos. edge
 87  000060 6002                        ori r16, 0x02       ;set ISC for rising
      edge
 88  000061 9300 0492                   sts PORTE_PIN2CTRL, r16
 89
 90                                     ;configure TCA0
 91  000063 e000                        ldi r16, TCA_SINGLE_WGMODE_NORMAL_gc
      ;WGMODE normal
 92  000064 9300 0a01                   sts TCA0_SINGLE_CTRLB, r16
 93
 94                                     ;enable overflow interrupt
 95  000066 e001                        ldi r16, TCA_SINGLE_OVF_bm
 96  000067 9300 0a0a                   sts TCA0_SINGLE_INTCTRL, r16
 97
 98                                     ;load period low byte then high byte
 99  000069 e405                        ldi r16, LOW(PERIOD_EXAMPLE_VALUE)
100  00006a 9300 0a26                   sts TCA0_SINGLE_PER, r16
101  00006c e001                        ldi r16, HIGH(PERIOD_EXAMPLE_VALUE)
102  00006d 9300 0a27                   sts TCA0_SINGLE_PER + 1, r16
103
104                                     ;set clock and start timer
105  00006f e00d                        ldi r16, TCA_SINGLE_CLKSEL_DIV256_gc |
      TCA_SINGLE_ENABLE_bm
106  000070 9300 0a00                   sts TCA0_SINGLE_CTRLA, r16
107
108  000072 9100 0a0a                   lds r16, TCA0_SPLIT_INTCTRL
109  000074 6001                        ori r16, 0x01
110  000075 9300 0a0a                   sts TCA0_SPLIT_INTCTRL, r16
111
112  000077 9478                        sei         ;enable global interrupts
113
```

```
                                    main_loop:
000078 0000                            nop
000079 cffe                            rjmp main_loop

                                    ;Interrupt service routine for any PORTE pin change IRQ
                                    porte_ISR:
00007a 930f                            push r16        ;save r16 then SREG
00007b b70f                            in r16, CPU_SREG
00007c 930f                            push r16
00007d 94f8                            cli             ;clear global interrupt enable

                                       ;Determine which pins of PORTE have IRQs
00007e 9100 0489                       lds r16, PORTE_INTFLAGS ;check for PE0 IRQ flag set
000080 fd00                            sbrc r16, 0
000081 d013                            rcall PB1_sub           ;execute subroutine for PE0

000082 9100 0489                       lds r16, PORTE_INTFLAGS ;check for PE2 IRQ flag set
000084 fd02                            sbrc r16, 2
000085 d019                            rcall PB2_sub           ;execute subroutine for PE2


000086 910f                            pop r16         ;restore SREG then r16
000087 bf0f                            out CPU_SREG, r16
000088 910f                            pop r16
000089 9518                            reti            ;return from PORTE pin change ISR

                                    ;Interrupt service routine for any overflow of counter
                                    mux_display_ISR:
00008a 930f                            push r16
00008b b70f                            in r16, CPU_SREG
00008c 930f                            push r16

00008d d06d                            rcall multiplex_display     ;multiplexes display

00008e e001                            ldi r16, TCA_SINGLE_OVF_bm  ;clear OVF flag
00008f 9300 0a0b                       sts TCA0_SINGLE_INTFLAGS, r16

000091 910f                            pop r16         ;restore SREG then r16
000092 bf0f                            out CPU_SREG, r16
000093 910f                            pop r16
```

```
154  000094 9518                        reti
155
156                                 pb1_sub:
157  000095 d045                        rcall reverse_bits
158  000096 301a                        cpi r17, $0A
159  000097 f430                        brsh non_bcd
160  000098 d048                        rcall shift_bcd_entries
161  000099 d050                        rcall bcd_to_led
162  00009a e001                        ldi r16, PORT_INT0_bm   ;clear IRQ flag for
       PE0
163  00009b 9300 0489                   sts PORTE_INTFLAGS, r16
164  00009d 9508                        ret
165
166                                 non_bcd:
167  00009e 9518                        reti
168
169                                 pb2_sub:
170  00009f e2b8                        ldi XH, HIGH(bcd_entries)
171  0000a0 e0a0                        ldi XL, LOW(bcd_entries)
172  0000a1 e020                        ldi r18, $00
173  0000a2 911d                        ld r17, X+
174  0000a3 9512                        swap r17
175  0000a4 913d                        ld r19, X+
176  0000a5 2b13                        or r17, r19
177  0000a6 910d                        ld r16, X+
178  0000a7 9502                        swap r16
179  0000a8 913c                        ld r19, X
180  0000a9 2b03                        or r16, r19
181  0000aa d07d                        rcall BCD2bin16
182
183                                     ;load_to_hex_results
184  0000ab e2b8                        ldi XH, HIGH(hex_results)
185  0000ac e0a9                        ldi XL, LOW(hex_results)
186  0000ad e030                        ldi r19, $00
187  0000ae 293f                        or r19, tbinH
188  0000af 7f30                        andi r19, $F0
189  0000b0 9532                        swap r19
190  0000b1 933d                        st X+, r19
191  0000b2 e030                        ldi r19, $00
192  0000b3 293f                        or r19, tbinH
193  0000b4 703f                        andi r19, $0F
194  0000b5 933d                        st X+, r19
195  0000b6 e030                        ldi r19, $00
196  0000b7 293e                        or r19, tbinL
197  0000b8 7f30                        andi r19, $F0
198  0000b9 9532                        swap r19
199  0000ba 933d                        st X+, r19
200  0000bb e030                        ldi r19, $00
201  0000bc 293e                        or r19, tbinL
```

```
202  0000bd 703f                         andi r19, $0F
203  0000be 933d                         st X+, r19
204
205                                      ;load to led_display
206  0000bf e2b8                         ldi XH, HIGH(led_display)
207  0000c0 e0a4                         ldi XL, LOW(led_display)
208  0000c1 e020                         ldi r18, $00
209  0000c2 292f                         or r18, tbinH
210  0000c3 7f20                         andi r18, $F0
211  0000c4 9522                         swap r18
212  0000c5 d06e                         rcall hex_to_7seg
213  0000c6 932d                         st X+, r18
214  0000c7 e020                         ldi r18, $00
215  0000c8 292f                         or r18, tbinH
216  0000c9 702f                         andi r18, $0F
217  0000ca d069                         rcall hex_to_7seg
218  0000cb 932d                         st X+, r18
219  0000cc e020                         ldi r18, $00
220  0000cd 292e                         or r18, tbinL
221  0000ce 7f20                         andi r18, $F0
222  0000cf 9522                         swap r18
223  0000d0 d063                         rcall hex_to_7seg
224  0000d1 932d                         st X+, r18
225  0000d2 e020                         ldi r18, $00
226  0000d3 292e                         or r18, tbinL
227  0000d4 702f                         andi r18, $0F
228  0000d5 d05e                         rcall hex_to_7seg
229  0000d6 932d                         st X+, r18
230  0000d7 e004                         ldi r16, PORT_INT2_bm   ;clear IRQ flag for
       PE2
231  0000d8 9300 0489                    sts PORTE_INTFLAGS, r16
232  0000da 9508                         ret
233
234                                      ;*****************************************
                     ******************************
235                                      ;*
236                                      ;* "reverse_bits" - Reverse Bits
237                                      ;*
238                                      ;* Description: Reverses the bit positions in
                     a byte passed in. Bit 0
239                                      ;* becomes bit 7, bit 6 becomes bit 1, and so
                     on.
240                                      ;*
241                                      ;* Author:                 Judah Ben-Eliezer
242                                      ;* Version:                    1.0
243                                      ;* Last updated:           101120
244                                      ;* Target:                 ATmega4809
245                                      ;* Number of words:            8
246                                      ;* Number of cycles:
```

```
247                                 ;* Low registers modified:  r16, r17, r18
248                                 ;* High registers modified:     none
249                                 ;*
250                                 ;* Parameters: r16: byte to be reversed.
251                                 ;* Returns: r16: reversed byte
252                                 ;*
253                                 ;* Notes:
254                                 ;*
255                                 ;*****************************************
                            ****************************

256
257                                 reverse_bits:
258 0000db e028                        ldi r18, $08
259                                 loop_8:
260 0000dc 9507                        ror r16
261 0000dd 1f11                        rol r17
262 0000de 952a                        dec r18
263 0000df f7e1                        brne loop_8
264 0000e0 9508                        ret
265
266                                 shift_bcd_entries:
267 0000e1 e023                        ldi r18, $03
268                                 shift_loop:
269 0000e2 e2b8                        ldi XH, HIGH(bcd_entries)
270 0000e3 e0a0                        ldi XL, LOW(bcd_entries)
271 0000e4 952a                        dec r18
272 0000e5 0fa2                        add XL, r18
273 0000e6 913d                        ld r19, X+
274 0000e7 933c                        st X, r19
275 0000e8 f7c9                        brne shift_loop
276 0000e9 9508                        ret
277
278                                 ;*****************************************
                            ****************************
279                                 ;*
280                                 ;* "bcd_to_led" - title
281                                 ;*
282                                 ;* Description:    Converts bcd input to 7seg
                            output, from bcd_entries array to led_display array
283                                 ;*
284                                 ;* Author:
285                                 ;* Version:
286                                 ;* Last updated:
287                                 ;* Target:
288                                 ;* Number of words:
289                                 ;* Number of cycles:
290                                 ;* Low registers modified:
291                                 ;* High registers modified:
292                                 ;*
```

```
293                                 ;* Parameters:
294                                 ;* Returns:
295                                 ;*
296                                 ;* Notes:
297                                 ;*
298                                 ;*****************************************
                    ******************************

299
300                                 bcd_to_led:
301  0000ea e0a0                       ldi XL, LOW(bcd_entries)
302  0000eb e2b8                       ldi XH, HIGH(bcd_entries)
303  0000ec 931c                       st X, r17
304  0000ed e044                       ldi r20, $04
305                                 conversion_loop:
306  0000ee 954a                       dec r20
307  0000ef e2b8                       ldi XH, HIGH(bcd_entries)
308  0000f0 e0a0                       ldi XL, LOW(bcd_entries)
309  0000f1 e2d8                       ldi YH, HIGH(led_display)
310  0000f2 e0c4                       ldi YL, LOW(led_display)
311  0000f3 0fa4                       add XL, r20
312  0000f4 0fc4                       add YL, r20
313  0000f5 912c                       ld r18, X
314  0000f6 d03d                       rcall hex_to_7seg
315  0000f7 8328                       st Y, r18
316  0000f8 3040                       cpi r20, $00
317  0000f9 f7a1                       brne conversion_loop
318  0000fa 9508                       ret
319
320
321                                 ;*****************************************
                    ******************************
322                                 ;*
323                                 ;* "multiplex_display" - title
324                                 ;*
325                                 ;* Description:    outputs values from
                    led_display array to 7 segment display on PORTD driven by
                    highest two bits of PORTC
326                                 ;*
327                                 ;* Author: Judah Ben-Eliezer
328                                 ;* Version:    1.0
329                                 ;* Last updated:   11/10/2020
330                                 ;* Target: ATmega4809
331                                 ;* Number of words:
332                                 ;* Number of cycles:
333                                 ;* Low registers modified:
334                                 ;* High registers modified:
335                                 ;*
336                                 ;* Parameters:
337                                 ;* Returns:
```

```
338                                 ;*
339                                 ;* Notes:
340                                 ;*
341                                 ;*****************************************
                        ****************************

342
343                                 multiplex_display:
344  0000fb e064                       ldi r22, $04
345  0000fc e070                       ldi r23, $00
346  0000fd 9370 2808                  sts digit_num, r23
347                                 loop_4:
348  0000ff e0c4                       ldi YL, LOW(led_display)
349  000100 9110 2808                  lds r17, digit_num
350  000102 9140 2808                  lds r20, digit_num
351  000104 7013                       andi r17, $03
352  000105 0fc1                       add YL, r17
353  000106 8128                       ld r18, Y
354  000107 e850                       ldi r21, $80
355  000108 9543                       inc r20
356                                 loop:
357  000109 9556                       lsr r21
358  00010a 954a                       dec r20
359  00010b f7e9                       brne loop
360  00010c 0f55                       lsl r21
361  00010d 9550                       com r21
362  00010e b959                       out VPORTC_OUT, r21
363  00010f b92d                       out VPORTD_OUT, r18
364  000110 9513                       inc r17
365  000111 9310 2808                  sts digit_num, r17
366  000113 956a                       dec r22
367  000114 f751                       brne loop_4
368  000115 9508                       ret
369
370                                 ;*****************************************
                        ****************************
371                                 ;*
372                                 ;* "BCD2bin16" - BCD to 16-Bit Binary
                        Conversion
373                                 ;*
374                                 ;* This subroutine converts a 5-digit packed
                        BCD number represented by
375                                 ;* 3 bytes (fBCD2:fBCD1:fBCD0) to a 16-bit
                        number (tbinH:tbinL).
376                                 ;* MSD of the 5-digit number must be placed in
                         the lowermost nibble of fBCD2.
377                                 ;*
378                                 ;* Let "abcde" denote the 5-digit number. The
                        conversion is done by
379                                 ;* computing the formula: 10(10(10(10a+b)+c)
```

```
                        +d)+e.
380                             ;* The subroutine "mul10a"/"mul10b" does the
                        multiply-and-add operation
381                             ;* which is repeated four times during the
                        computation.
382                             ;*
383                             ;* Number of words  :30
384                             ;* Number of cycles     :108
385                             ;* Low registers used   :4 (copyL,copyH,mp10L/
                        tbinL,mp10H/tbinH)
386                             ;* High registers used  :4
                        (fBCD0,fBCD1,fBCD2,adder)
387                             ;*
388                             ;*****************************************
                        ****************************

389
390                             ;***** "mul10a"/"mul10b" Subroutine Register
                        Variables
391
392                             .def    copyL   =r12        ;temporary register
393                             .def    copyH   =r13        ;temporary register
394                             .def    mp10L   =r14        ;Low byte of number
                         to be multiplied by 10
395                             .def    mp10H   =r15        ;High byte of
                        number to be multiplied by 10
396                             .def    adder   =r19        ;value to add after
                         multiplication
397
398                             ;***** Code
399
400                             mul10a:     ;***** multiplies "mp10H:mp10L"
                        with 10 and adds "adder" high nibble
401  000116 9532                    swap    adder
402                             mul10b:     ;***** multiplies "mp10H:mp10L"
                        with 10 and adds "adder" low nibble
403  000117 2cce                    mov copyL,mp10L ;make copy
404  000118 2cdf                    mov copyH,mp10H
405  000119 0cee                    lsl mp10L       ;multiply original by 2
406  00011a 1cff                    rol mp10H
407  00011b 0ccc                    lsl copyL       ;multiply copy by 2
408  00011c 1cdd                    rol copyH
409  00011d 0ccc                    lsl copyL       ;multiply copy by 2 (4)
410  00011e 1cdd                    rol copyH
411  00011f 0ccc                    lsl copyL       ;multiply copy by 2 (8)
412  000120 1cdd                    rol copyH
413  000121 0cec                    add mp10L,copyL ;add copy to original
414  000122 1cfd                    adc mp10H,copyH
415  000123 703f                    andi    adder,0x0f  ;mask away upper nibble
       of adder
```

```
000124 0ee3                         add mp10L,adder ;add lower nibble of adder
000125 f408                         brcc    m10_1       ;if carry not cleared
000126 94f3                         inc mp10H       ;   inc high byte
000127 9508                       m10_1: ret

                                  ;***** Main Routine Register Variables

                                  .def    tbinL   =r14        ;Low byte of binary
                      result (same as mp10L)
                                  .def    tbinH   =r15        ;High byte of
                     binary result (same as mp10H)
                                  .def    fBCD0   =r16        ;BCD value digits 1
                      and 0
                                  .def    fBCD1   =r17        ;BCD value digits 2
                      and 3
                                  .def    fBCD2   =r18        ;BCD value digit 5

                                  ;***** Code

                                  BCD2bin16:
000128 702f                         andi    fBCD2,0x0f  ;mask away upper nibble
       of fBCD2
000129 24ff                         clr mp10H
00012a 2ee2                         mov mp10L,fBCD2 ;mp10H:mp10L = a
00012b 2f31                         mov adder,fBCD1
00012c dfe9                         rcall   mul10a      ;mp10H:mp10L = 10a+b
00012d 2f31                         mov adder,fBCD1
00012e dfe8                         rcall   mul10b      ;mp10H:mp10L = 10(10a
      +b)+c
00012f 2f30                         mov adder,fBCD0
000130 dfe5                         rcall   mul10a      ;mp10H:mp10L = 10(10
      (10a+b)+c)+d
000131 2f30                         mov adder,fBCD0
000132 dfe4                         rcall   mul10b      ;mp10H:mp10L = 10(10(10
      (10a+b)+c)+d)+e
000133 9508                         ret


                                  ;*****************************************
                     ****************************
                                  ;*
                                  ;* "hex_to_7seg" - Hexadecimal to Seven
                     Segment Conversion
                                  ;*
                                  ;* Description: Converts a right justified
                     hexadecimal digit to the seven
                                  ;* segment pattern required to display it.
                     Pattern is right justified a
```

```
453                                 ;* through g. Pattern uses 0s to turn segments on ON.
454                                 ;*
455                                 ;* Author:                      Ken Short
456                                 ;* Version:                         1.0
457                                 ;* Last updated:                101620
458                                 ;* Target:                      ATmega4809
459                                 ;* Number of words:                 8
460                                 ;* Number of cycles:            13
461                                 ;* Low registers modified:      none
462                                 ;* High registers modified:         r16, r18, ZL, ZH
463                                 ;*
464                                 ;* Parameters: r18: right justified hex digit, high nibble 0
465                                 ;* Returns: r18: segment values a through g right justified
466                                 ;*
467                                 ;* Notes:
468                                 ;*
469                                 ;******************************************************************
470
471                                 hex_to_7seg:
472 000134 702f                       andi r18, 0x0F              ;clear ms nibble
473 000135 e0f2                        ldi ZH, HIGH(hextable * 2)  ;set Z to point to start of table
474 000136 e7e8                        ldi ZL, LOW(hextable * 2)
475 000137 e000                        ldi r16, $00                ;add offset to Z pointer
476 000138 0fe2                        add ZL, r18
477 000139 1ff0                        adc ZH, r16
478 00013a 9124                        lpm r18, Z                  ;load byte from table pointed to by Z
479 00013b 9508                       ret
480
481                                    ;Table of segment values to display digits 0 - F
482                                    ;!!! seven values must be added - verify all values
483 00013c 4f01
484 00013d 0612
485 00013e 244c
486 00013f 0f20
487 000140 0400
488 000141 6008
489 000142 4231
```

```
490
491
492 RESOURCE USE INFORMATION
493 ------------------------
494
495 Notice:
496 The register and instruction counts are symbol table hit counts,
497 and hence implicitly used resources are not counted, eg, the
498 'lpm' instruction without operands implicitly uses r0 and z,
499 none of which are counted.
500
501 x,y,z are separate entities in the symbol table and are
502 counted separately from r26..r31 here.
503
504 .dseg memory usage only counts static data declared with .byte
505
506 "ATmega4809" register use summary:
507 x  :  20 y  :   2 z  :   1 r0 :   0 r1 :   0 r2 :   0 r3 :   0 r4 :   0
508 r5 :   0 r6 :   0 r7 :   0 r8 :   0 r9 :   0 r10:   0 r11:   0 r12:   5
509 r13:   5 r14:   9 r15:   9 r16:  59 r17:  13 r18:  33 r19:  31 r20:   8
510 r21:   5 r22:   2 r23:   2 r24:   0 r25:   0 r26:  10 r27:   7 r28:   5
511 r29:   2 r30:   2 r31:   2
512 Registers used: 21 out of 35 (60.0%)
513
514 "ATmega4809" instruction use summary:
515 .lds  :   0 .sts  :   0 adc   :   2 add   :   7 adiw  :   0 and   :   0
516 andi  :  12 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
517 brcc  :   1 brcs  :   0 break :   0 breq  :   0 brge  :   0 brhc  :   0
518 brhs  :   0 brid  :   0 brie  :   0 brlo  :   0 brlt  :   0 brmi  :   0
519 brne  :   5 brpl  :   0 brsh  :   1 brtc  :   0 brts  :   0 brvc  :   0
520 brvs  :   0 bset  :   0 bst   :   0 call  :   0 cbi   :   2 cbr   :   0
521 clc   :   0 clh   :   0 cli   :   1 cln   :   0 clr   :   1 cls   :   0
522 clt   :   0 clv   :   0 clz   :   0 com   :   3 cp    :   0 cpc   :   0
523 cpi   :   2 cpse  :   0 dec   :   5 des   :   0 eor   :   0 fmul  :   0
524 fmuls :   0 fmulsu:   0 icall :   0 ijmp  :   0 in    :   2 inc   :   3
525 jmp   :   2 ld    :   7 ldd   :   0 ldi   :  47 lds   :   7 lpm   :   2
526 lsl   :   5 lsr   :   1 mov   :   7 movw  :   0 mul   :   0 muls  :   0
527 mulsu :   0 neg   :   0 nop   :   1 or    :  10 ori   :   3 out   :   7
528 pop   :   4 push  :   4 rcall :  16 ret   :   9 reti  :   3 rjmp  :   1
529 rol   :   5 ror   :   1 sbc   :   0 sbci  :   0 sbi   :   0 sbic  :   0
530 sbis  :   0 sbiw  :   0 sbr   :   0 sbrc  :   2 sbrs  :   0 sec   :   0
531 seh   :   0 sei   :   1 sen   :   0 ser   :   0 ses   :   0 set   :   0
532 sev   :   0 sez   :   0 sleep :   0 spm   :   0 st    :  15 std   :   0
533 sts   :  13 sub   :   0 subi  :   0 swap  :   7 tst   :   0 wdr   :   0
534
535 Instructions used: 39 out of 114 (34.2%)
536
537 "ATmega4809" memory use summary [bytes]:
538 Segment   Begin    End      Code   Data   Used    Size   Use%
```

```
539  ------------------------------------------------------------
540  [.cseg] 0x00001c 0x000288    496     16    512   49152   1.0%
541  [.dseg] 0x002800 0x00280d      0     13     13    6144   0.2%
542  [.eseg] 0x000000 0x000000      0      0      0     256   0.0%
543
544  Assembly complete, 0 errors, 2 warnings
545
```