```
1  ;
2  ; ADC_sgnl_conv.asm
3  ;
4  ; Created: 11/20/2021 10:40:04 PM
5  ; Author : Kyle Han
6  ;
7
8
9  .equ PERIOD = 97 ////(4MHz * (1/160))/256)-1
10
11
12 .dseg
13 bcd_entries: .byte 4
14 led_display: .byte 4
15 digit_num: .byte 1
16
17
18 .cseg
19 reset:
20     rjmp start
21 .org TCA0_OVF_vect
22     rjmp multiplex_display
23
24
25 start:
26
27 //This section deals with port configurations
28     //Set all of PORTD to outputs (For the 7seg)
29     ldi r16, 0xFF
30     sts PORTD_DIR, r16
31
32     //Set PA7-PA4 as outputs (For the multiplexer)
33     ldi r16, 0xF0
34     sts PORTA_DIR, r16
35
36     //Configures all of PORTE as inputs
37     ldi r16, 0x00
38     sts PORTE_DIR, r16
39
40 //This section deals with timer TCA0
41 //Sets up TCA0 timer for the interrupt. 40Hz
42     ldi r16, TCA_SINGLE_WGMODE_NORMAL_gc    ;WGMODE normal
43     sts TCA0_SINGLE_CTRLB, r16
44
45     ldi r16, TCA_SINGLE_OVF_bm      ;enable overflow interrupt
46     sts TCA0_SINGLE_INTCTRL, r16
47
48     ;load period low byte then high byte
49     ldi r16, LOW(PERIOD)        ;set the period
50     sts TCA0_SINGLE_PER, r16
51     ldi r16, HIGH(PERIOD)
52     sts TCA0_SINGLE_PER + 1, r16
```

```
53
54
55  //This section deals with Analog-to-Digital-Converter ADC0
56      //Set the controls as a 12 bit resolution, and enable the ADC
57      ldi r16, ADC_RESSEL_12BIT_gc | ADC_ENABLE_bm
58      sts ADC0_CTRLA, r16
59
60      //Config the voltage ref to 2.5V
61      ldi r16, VREF_REFSEL_2V500_gc
62      sts VREF_ADC0REF, r16
63
64      //Sets the prescaler to 64 bits
65      ldi r16, ADC_PRESC_DIV64_gc
66      sts ADC0_CTRLC, r16
67
68      //Configs PE3 as the positive analog input to the ADC
69      //AIN11 = PE3
70      ldi r16, ADC_MUXPOS_AIN11_gc
71      sts ADC0_MUXPOS, r16
72
73
74      //POST the display. Once posted, then start the timer and continue as normal.
75      rcall post_display
76
77      //set clock and start timer
78      ldi r16, TCA_SINGLE_CLKSEL_DIV256_gc | TCA_SINGLE_ENABLE_bm
79      sts TCA0_SINGLE_CTRLA, r16
80
81      //Start ADC0's conversion
82      ldi r16, ADC_STCONV_bm
83      sts ADC0_COMMAND, r16
84      sei
85
86  loop:
87      rcall pollADC
88      rjmp loop
89
90
91  ;***********************************************************************
92  ;*
93  ;* "pollADC" - Poll the ADC's result int flag
94  ;*
95  ;* Description: Polls ADC0's INTFLAGS register to check if a result is ready.
96  ;* If it is ready, it will read in the value, and update bcd_entries/led_display ↵
      accordingly
97  ;* Author: Kyle Han
98  ;* Version: 0.1
99  ;* Last updated: 11/29/2021
100 ;* Target: AVR128DB48
101 ;* Number of words:
102 ;* Number of cycles:
103 ;* Low registers modified: r16-r18
```

```
104 ;* High registers modified:
105 ;*
106 ;* Parameters: None
107 ;*
108 ;* Returns: None
109 ;*
110 ;* Notes: Requires digit_num, bcd_entries, and led_display to be declared
      in .dseg
111 ;*
112 ;***************************************************************************
113 pollADC:
114     //Saving things to the stack
115     push r16
116     push r17
117     push r18
118     push r19
119
120     //Check the flag. If 1, then continue. If 0, then remove the things from the
        stack and return.
121     lds r16, ADC0_INTFLAGS
122     sbrs r16, 0
123     rjmp retADC
124
125     //Past this point, we know that a conversion is ready for us to read in.
126     lds r19, ADC0_RESL
127     lds r17, ADC0_RESH
128
129 //Will copy everything needed into bcd_entries, and create the corresponding
      values in led_display
130     //Entry 0
131     mov r18, r17
132     andi r18, 0xF0
133     //We need a shift right to get this back to bcd (Since that only requires 4
        bits)
134     lsr r18
135     lsr r18
136     lsr r18
137     lsr r18
138     sts bcd_entries, r18
139     rcall hex_to_7seg
140     sts led_display, r18
141
142     //Entry 1
143     mov r18, r17
144     andi r18, 0x0F
145     sts bcd_entries+1, r18
146     rcall hex_to_7seg
147     sts led_display+1, r18
148
149     //Entry 2
150     mov r18, r19
151     andi r18, 0xF0
```

```
152     lsr r18
153     lsr r18
154     lsr r18
155     lsr r18
156     sts bcd_entries+2, r18
157     rcall hex_to_7seg
158     sts led_display+2, r18
159
160     //Entry 3
161     mov r18, r19
162     andi r18, 0x0F
163     sts bcd_entries+3, r18
164     rcall hex_to_7seg
165     sts led_display+3, r18
166
167 //Returns from the ADC subroutine. Saves things and make sure the registers are left unmodified.
168 retADC:
169     pop r19
170     pop r18
171     pop r17
172     pop r16
173
174     //Restart ADC0's conversion
175     ldi r16, ADC_STCONV_bm
176     sts ADC0_COMMAND, r16
177     ret
178
179 ;************************************************************************
180 ;*
181 ;* "hex_to_7seg" - Hexadecimal to Seven Segment Conversion
182 ;*
183 ;* Description: Converts a right justified hexadecimal digit to the seven
184 ;* segment pattern required to display it. Pattern is right justified a
185 ;* through g. Pattern uses 0s to turn segments on ON.
186 ;*
187 ;* Author:          Ken Short
188 ;* Version:         0.1
189 ;* Last updated:        101221
190 ;* Target:          AVR128DB48
191 ;* Number of words:
192 ;* Number of cycles:
193 ;* Low registers modified:
194 ;* High registers modified:
195 ;*
196 ;* Parameters: r18: hex digit to be converted
197 ;* Returns: r18: seven segment pattern. 0 turns segment ON
198 ;*
199 ;* Notes:
200 ;*
201 ;************************************************************************
202 hex_to_7seg:
```

```
203     push ZH
204     push ZL
205     ldi ZH, HIGH(hextable * 2)  ;set Z to point to start of table
206     ldi ZL, LOW(hextable * 2)
207     ldi r16, $00                ;add offset to Z pointer
208     andi r18, 0x0F              ;mask for low nibble
209     add ZL, r18
210     adc ZH, r16
211     lpm r18, Z                  ;load byte from table pointed to by Z
212     pop ZL
213     pop ZH
214     ret
215
216     ;Table of segment values to display digits 0 - F
217     ;!!! seven values must be added
218 hextable: .db $01, $4F, $12, $06, $4C, $24, $20, $0F, $00, $04, $08, $60, $31,
      $42, $30, $38
219
220
221 ;***********************************************************************
222 ;*
223 ;* Multiplex_display
224 ;*
225 ;* Description: reads everything in the memory "array" and converts it to a 7seg
      interpetatoin.
226 ;*
227 ;*
228 ;*
229 ;* Author:          Kyle Han
230 ;* Version:         0.1
231 ;* Last updated:        11032021
232 ;* Target:          AVR128DB48
233 ;* Number of words:
234 ;* Number of cycles:
235 ;* Low registers modified:
236 ;* High registers modified:
237 ;*
238 ;* Parameters: A pointer called array, and pointer X available
239 ;* Returns: Everything inside pointer X
240 ;*
241 ;* Notes:
242 ;*
243 ;***********************************************************************
244
245
246 multiplex_display:
247
248 //Turns off the whole display by outputting 1s to PORTA. Then, check with
      digit_num to see which display should be lit
249 turn_off:
250
251     //First, we must push all the registers we are using to the stack. This is so
```

```
           that the original values are restored later on
252     push r16
253     push r17
254     push r18
255     in r16, CPU_SREG
256     push r16
257     push XL
258     push XH
259
260     //Handles the interrupt and resets the timer
261     ldi r16, TCA_SINGLE_OVF_bm  ;clear OVF flag
262     sts TCA0_SINGLE_INTFLAGS, r16
263
264     ldi r16, 0xFF
265     sts PORTA_OUT, r16
266     lds r17, digit_num
267     inc r17
268     //If we are at position 4, return to position 0
269     cpi r17, 0x04
270     brsh overflow
271
272 output:
273     sts digit_num, r17
274     ldi XH, HIGH(led_display)
275     ldi XL, LOW(led_display)
276     add XL, r17
277     ld r18, X
278     //rcall hex_to_7seg
279     sts PORTD_OUT, r18
280
281 //Will check digit_num to decide which display on the 7seg is ebing outputted
282 checking_dig:
283     cpi r17, 0x00
284     breq dig0
285     cpi r17, 0x01
286     breq dig1
287     cpi r17, 0x02
288     breq dig2
289     cpi r17, 0x03
290     breq dig3
291
292     //r18 stores the value of the digit to be displayed
293 dig0:
294     ldi r18, 0x70   //PA7
295     sts PORTA_OUT, r18
296     rjmp restore
297
298 dig1:
299     ldi r18, 0xB0   //PA6
300     sts PORTA_OUT, r18
301     rjmp restore
302
```

```
303 dig2:
304     ldi r18, 0xD0   //PA5
305     sts PORTA_OUT, r18
306     rjmp restore
307
308 dig3:
309     ldi r18, 0xE0 //PA4
310     sts PORTA_OUT, r18
311     rjmp restore
312
313 //Reads off the stack the values that we preserved inside turn_off
314 restore:
315     pop XH
316     pop XL
317     pop r16
318     out CPU_SREG, r16
319     pop r18
320     pop r17
321     pop r16
322
323     reti
324
325 //Sets digit_num back to 0, since we only have 4 digits
326 overflow:
327     ldi r17, 0x00
328     sts digit_num, r17
329     ldi XH, HIGH(led_display)
330     ldi XL, LOW(led_display)
331     rjmp output
332
333
334 ;***************************************************************************
335 ;*
336 ;* "post_display" - Power On Self Test
337 ;*
338 ;* Description: Will POST the 7segment display hooked up to port D, multiplexed
      by PA7-PA4.
339 ;*Individually turns on each segment for a brief moment, totalling 1 second
340 ;* Author: Kyle Han
341 ;* Version: 0.1
342 ;* Last updated: 11/29/2021
343 ;* Target: AVR128DB48
344 ;* Number of words:
345 ;* Number of cycles:
346 ;* Low registers modified: r16-r20
347 ;* High registers modified:
348 ;*
349 ;* Parameters: None
350 ;*
351 ;* Returns: None
352 ;*
353 ;* Notes: Requires digit_num to be declared in .dseg
```

```
354 ;*
355 ;*********************************************************************
356 post_display:
357     //A value of 0 turns on our LED displays
358     ldi r16, 0x00
359     out VPORTD_OUT, r16
360     sts digit_num, r16 ;Store 0x00 to the current digit to be displayed
361 repeatPost:
362     in r17, VPORTA_OUT
363     ori r17, 0xF0
364     out VPORTA_OUT, r17
365     ldi r19, 0b00010000 ;This 1 will turn on a specific digit.
366 leftShiftPOST:
367     cpi r16, 0
368     breq postDispOn
369     lsl r19
370     dec r16
371     rjmp leftShiftPOST
372 postDispOn:
373     in r17, VPORTA_OUT
374     eor r17, r19
375     out VPORTA_OUT, r17
376     ldi r20, 10
377     rcall oneSecDelay
378     //Compares digit_num to 0x04. If greater than or equal to, we've finished ↵
        posting
379     lds r16, digit_num
380     inc r16
381     sts digit_num, r16
382     cpi r16, 0x04
383     brlo repeatPost
384
385     in r17, VPORTA_OUT
386     ori r17, 0xF0
387     out VPORTA_OUT, r17
388     ret
389
390 ;*********************************************************************
391 ;*
392 ;* "oneSecDelay" - Delay the microcontroller by 1 second by occupying CPU time
393 ;*
394 ;* Description: Loops through and occupies CPU time to delay by 1 second
395 ;*
396 ;* Author: Kyle Han
397 ;* Version: 1.0
398 ;* Last updated: 11/29/2021
399 ;* Target: AVR128DB48
400 ;* Number of words:
401 ;* Number of cycles:
402 ;* Low registers modified:r17
403 ;* High registers modified:
404 ;*
```

```
405 ;* Parameters: r20 * .1ms
406 ;*
407 ;* Returns: Nothing
408 ;*
409 ;* Notes:
410 ;*
411 ;***************************************************************************
412 oneSecDelay:
413 outer_loop:
414     ldi r17, 133
415 inner_loop:
416     dec r17
417     brne inner_loop
418     dec r20
419     brne outer_loop
420     ret
```