

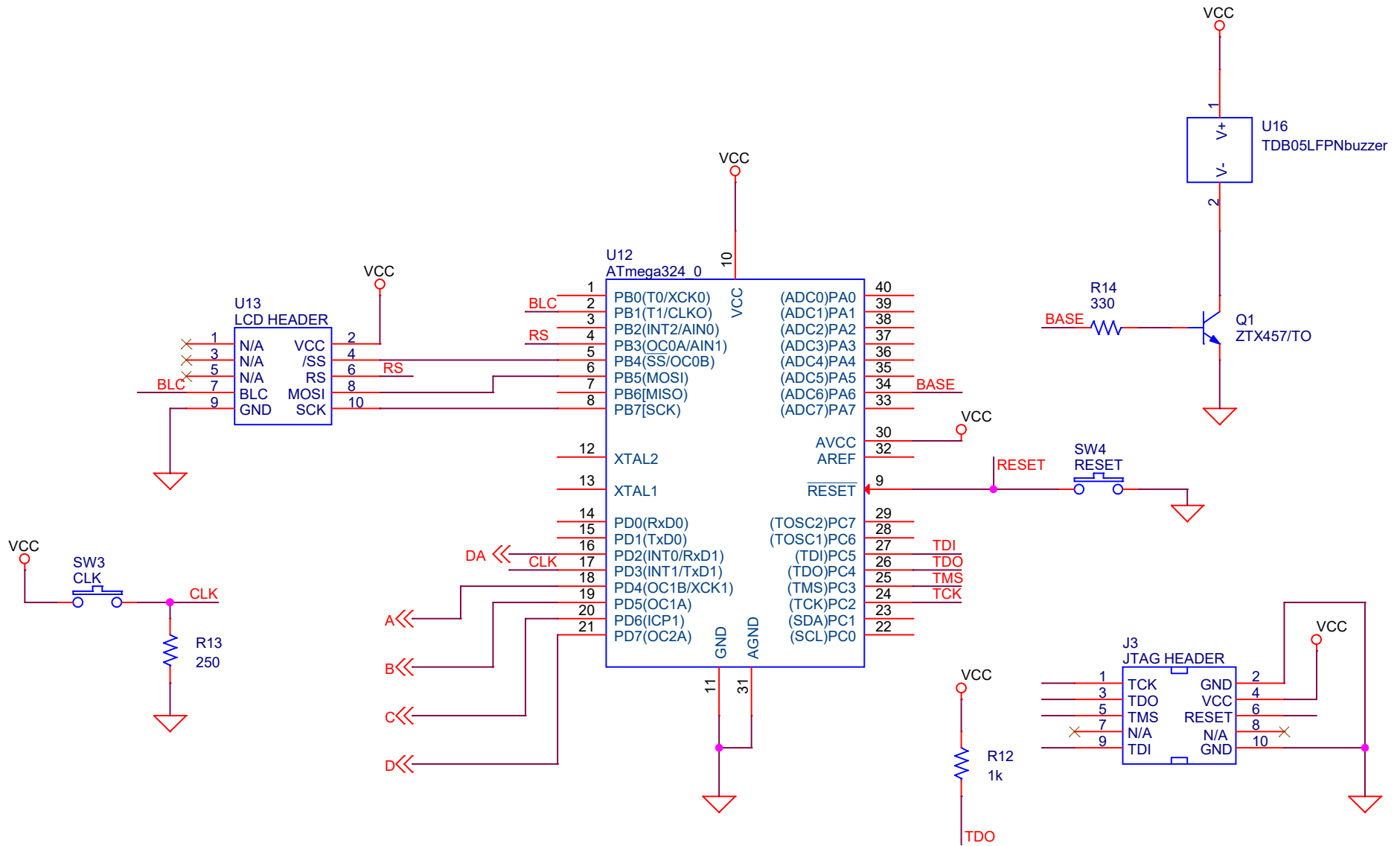
Ma & Varghese

111486364 & 111604890

Prelab #11

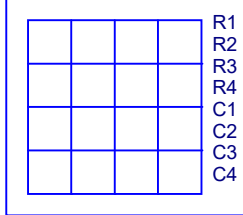
L-05

Bench #10



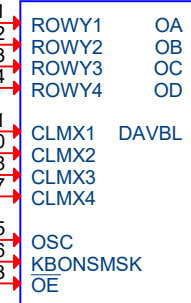
| | | |
|------------------------------|---------------------------|-----------------------------|
| Title | | |
| ESE280-Prelab#10 SCHEMATIC#1 | | Aaron Varghese & Richard Ma |
| Size | Document Number | Rev |
| A | <Doc> | <RevCo> |
| Date: | Sunday, December 01, 2019 | Sheet 1 of 1 |

U14
KEYPAD



1 ROW1
2 ROW2
3 ROW3
4 ROW4
5 COLUMN1
6 COLUMN2
7 COLUMN3
8 COLUMN4

U15
MM74C922

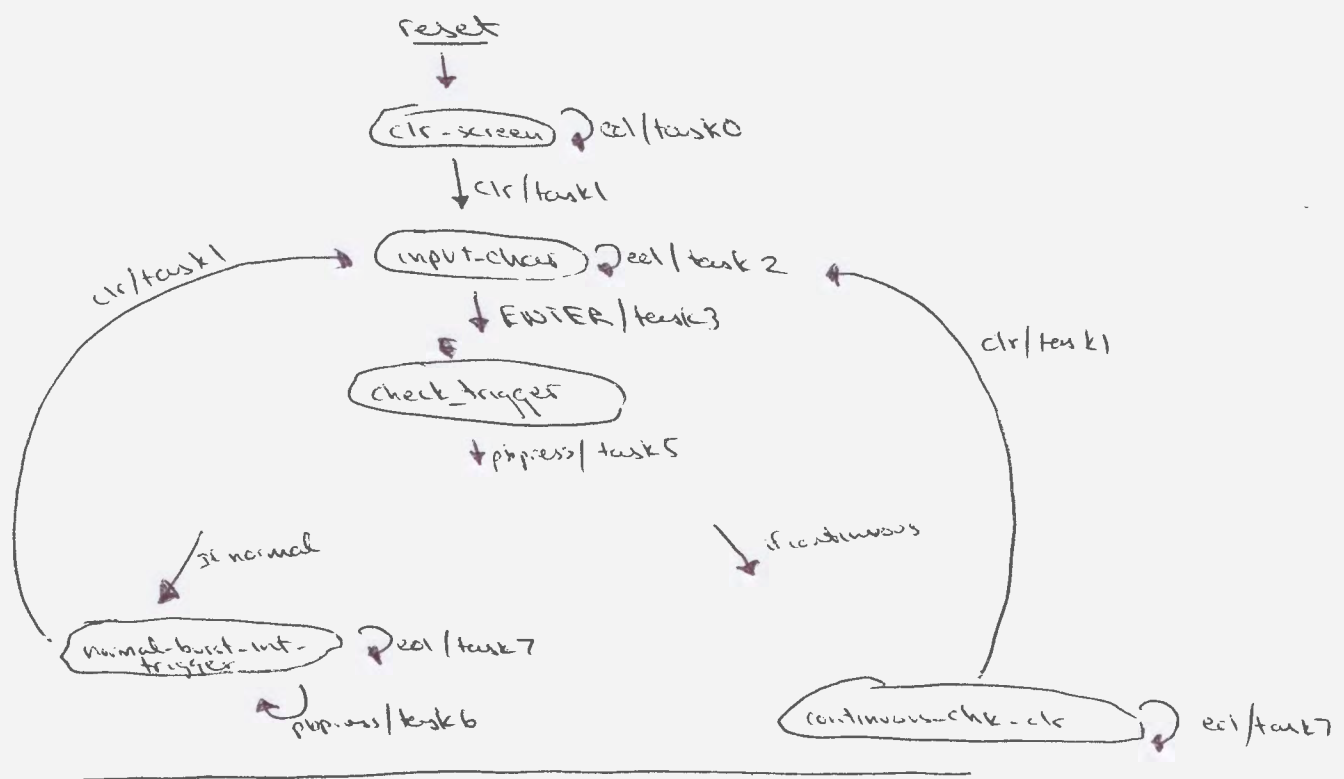


17 <<A
16 <<B
15 <<C
14 <<D
12 <<DA

C3
0.1u
0

C4
1u
0

| | | |
|------------------------------|---------------------------|-----------------------------|
| Title | | |
| ESE280-Prelab#10 SCHEMATIC#2 | | Aaron Varghese & Richard Ma |
| Size A | Document Number <Doc> | Rev <Rev Co |
| Date: | Sunday, December 01, 2019 | Sheet 1 of 1 |



task 0 → sounds buzzer for all other inputs pressed (not CLR)

task 1 → displays n =

task 2 → inputs char into line and sounds buzzer for char that arent digits or FWIR

task 3 → saves setting

task 4 → sounds buzzer for all inputs besides ph

task 5 → check burst count number

task 6 → reinitialize burst setting

task 7 → sounds buzzer.

```

...2019\ESE 280\Labs\Prelab11\ppg_IV_fsm\ppg_IV_fsm\ppg_IV_fsm\Debug\ppg_IV_fsm.lss 1
1
2 AVRASM ver. 2.2.7 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ ppg_IV_fsm  ↗
  \ppg_IV_fsm\ppg_IV_fsm\main.asm Thu Dec 05 18:44:32 2019
3
4 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm\ppg_IV_f sm  ↗
  \ppg_IV_fsm\main.asm(15): Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP  ↗
  \1.2.150\avrasm\inc\m324adef.inc'
5 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm\ppg_IV_f sm  ↗
  \ppg_IV_fsm\main.asm(146): Including file 'C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 201 9\ESE 280  ↗
  \Labs\Prelab11\ppg_IV_fsm\ppg_IV_fsm\ppg_IV_fsm\lcd_dog_asm_driver_m324a.inc'
6 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm\ppg_IV_f sm  ↗
  \ppg_IV_fsm\main.asm(147): Including file 'C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 201 9\ESE 280  ↗
  \Labs\Prelab11\ppg_IV_fsm\ppg_IV_fsm\ppg_IV_fsm\subroutines.inc'
7 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm\ppg_IV_f sm  ↗
  \ppg_IV_fsm\main.asm(433): warning: Register r14 already defined by the .DEF directive
8 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm\ppg_IV_f sm  ↗
  \ppg_IV_fsm\main.asm(434): warning: Register r15 already defined by the .DEF directive
9 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm\ppg_IV_f sm  ↗
  \ppg_IV_fsm\main.asm(15): Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP  ↗
  \1.2.150\avrasm\inc\m324adef.inc'
10 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm\ppg_IV_f sm  ↗
  \ppg_IV_fsm\main.asm(146): Including file 'C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 201 9\ESE 280  ↗
  \Labs\Prelab11\ppg_IV_fsm\ppg_IV_fsm\ppg_IV_fsm\lcd_dog_asm_driver_m324a.inc'
11 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm\ppg_IV_f sm  ↗
  \ppg_IV_fsm\main.asm(147): Including file 'C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 201 9\ESE 280  ↗
  \Labs\Prelab11\ppg_IV_fsm\ppg_IV_fsm\ppg_IV_fsm\subroutines.inc'
12
13
14 ; ppg_IV_fsm.asm
15 ;
16 ; Created: 11/27/2019 2:28:21 PM
17 ; Author : Aaron
18 ; Version: 1.0
19 ; Target: ATMEGA324A
20 ; Description: The purpose of the program is to

```

```

21          ; implement Lab 10 functionality(Lab 8 function
22          ; with interrupts) except this time it will be
23          ; implemented using the fsm chart
24
25
26          .list
27
28
29          .def pstatel = r24 ;low byte of present state address
30          .def pstateh = r25 ;high byte of present state address
31
32          start:
33          .org 0x0000
34 000000 940c 0005      jmp init
35          .org INT0addr
36 000002 940c 00c7      jmp ISR0
37          .org INT1addr
38 000004 c0d0          rjmp ISR1
39
40          .dseg
41 000100      burst_count_bcd_setting:      .byte 3 ;setting in bcd
42 000103      burst_count_binary_setting:    .byte 1 ;setting in binary
43 000104      normal_flag:                  .byte 1 ;1 is set
44 000105      continuous_flag:              .byte 1 ; 1 is set
45
46          .cseg
47          init:
48          ;initialize the stack pointer
49 000005 ef0f      ldi r16, LOW(RAMEND)
50 000006 bf0d      out SPL,r16
51 000007 e008      ldi r16, HIGH(RAMEND)
52 000008 bf0e      out SPH, r16
53
54          ;setting up PORTB
55 000009 ef0f      ldi r16, $FF          ; set portB = output.

```

```

56 00000a b904          out DDRB, r16      ;
57 00000b 9a2c          sbi PORTB, 4        ; set /SS of DOG LCD = 1 (Deselected)
58
59                      ;setting up PORTD
60 00000c e000          ldi r16, $00;PortD is an input port
61 00000d b90a          out DDRD, r16
62
63 00000e e003          ldi r16, $03;initialize pull-up resistors for PD0/1
64 00000f b90b          out PORTD, r16
65
66                      ;initialize the output PA7 for the pulses
67                      ;initialize output PA6 for the buzzer
68 000010 ec00          ldi r16, $C0
69 000011 b901          out DDRA, r16
70
71                      ;put FSM in initial state
72 000012 e487          ldi pstatel, LOW(clear_screen)
73 000013 e091          ldi pstateh, HIGH(clear_screen)
74
75
76 000014 e00f          ldi r16, (1<<ISC00)|(1<<ISC01)|(1<<ISC10)|(1<<ISC11)
77 000015 9300 0069     sts EICRA, r16
78
79 000017 e003          ldi r16, (1<<INT0)|(1<<INT1)
80 000018 bb0d          out EIMSK, r16
81
82 000019 9478          sei
83
84                      main_loop:
85 00001a e350          ldi r21, $30
86 00001b e360          ldi r22, $30
87 00001c e370          ldi r23, $30
88
89 00001d e000          ldi r16, 0
90 00001e 9300 0105     sts continuous_flag, r16

```

```

91 000020 9300 0104      sts normal_flag, r16
92                        checking1:
93                        ;-----
94                        ;check and see if the flag for continuous was set
95 000022 9100 0105      lds r16, continuous_flag
96 000024 3001           cpi r16, 1
97 000025 f0c9           breq zero_burst_setting
98
99                        ;check to see if the flag for normal burst was set
100 000026 9100 0104     lds r16, normal_flag
101 000028 3001           cpi r16, 1
102 000029 f009           breq normal_burst_setting
103                        ;nothing was set, so go to the beginnning
104 00002a f7b9           brne checking1
105
106                        ;if the burst is not set at 0
107                        normal_burst_setting:
108                        ;clear the flag to show task is done
109 00002b e000           ldi r16, 0
110 00002c 9300 0104     sts normal_flag, r16
111                        output_normal_burst_setting:
112 00002e e00a           ldi r16,10
113 00002f 9a17           sbi PORTA, 7
114 000030 d10a           rcall var_delay
115 000031 9817           cbi PORTA,7
116 000032 e00a           ldi r16,10
117 000033 d107           rcall var_delay
118 000034 956a           dec r22
119 000035 f7c1           brne output_normal_burst_setting
120
121                        polling:
122                        ;check and see if setting is reinitialized
123 000036 9100 0104     lds r16, normal_flag
124 000038 3003           cpi r16, 3
125 000039 f389           breq normal_burst_setting

```



```
126
127                                     ;check and see if CLR is pressed
128 00003a 9100 0104                   lds r16, normal_flag
129 00003c 3002                       cpi r16, 2
130 00003d f2e1                       breq main_loop
131
132 00003e f7b9                       brne polling
133
134                                     ;if the burst happens to be set at 0
135 zero_burst_setting:
136                                     ;clear the flag to show task is done
137 00003f e000                       ldi r16, 0
138 000040 9300 0105                   sts continuous_flag, r16
139                                     output_zero_burst_setting:
140 000042 e00a                       ldi r16, 10
141 000043 9a17                       sbi PORTA, 7
142 000044 d0f6                       rcall var_delay
143 000045 0000                       nop
144 000046 0000                       nop
145 000047 9817                       cbi PORTA,7
146 000048 e00a                       ldi r16, 10
147 000049 d0f7                       rcall var_delay_2
148                                     ;check and see if CLR is pressed
149 00004a 9100 0105                   lds r16, continuous_flag
150 00004c 3002                       cpi r16, 2
151 00004d f261                       breq main_loop
152
153 00004e cff3                       rjmp output_zero_burst_setting
154
155                                     .list
156
157                                     ;*****
158                                     ;*
159                                     ;*Title: ISR0
160                                     ;* Description: checks what button on the keypad is presseed
```

```
161      ;*
162      ;* Target:ATMEGA324A
163      ;* Number of words: 17 words
164      ;* Number of cycles: 25 cycles
165      ;*
166      ;* High registers modified:
167      ;*
168      ;* Returns: N/A
169      ;*
170      ;*also calls upon the code_to_value subroutine to decode the button press
171      ;* returns the keycode in r18
172      ;*****
173      ISR0:
174      0000c7 930f      push r16
175      0000c8 b70f      in r16, SREG
176      0000c9 930f      push r16
177      0000ca b109      in r16, PIND
178      0000cb 7f00      andi r16, $F0
179      0000cc 9502      swap r16
180      0000cd 2f20      mov r18, r16
181      0000ce d019      rcall code_to_value
182      0000cf 2f02      mov r16, r18
183      0000d0 d097      rcall fsm
184      0000d1 910f      pop r16
185      0000d2 bf0f      out SREG, r16
186      0000d3 910f      pop r16
187      0000d4 9518      reti
188
189      ;*****
190      ;*
191      ;*Title: ISR1
192      ;* Description:activates when the pushbutton is pressed
193      ;*
194      ;* Target:ATMEGA324A
195      ;* Number of words: 22
```

```
196 ;* Number of cycles: 24769
197 ;*
198 ;* Low registers modified: N/A
199 ;* High registers modified: N/A
200
201 ;* Returns: carry flag set or cleared
202 ;*
203 ;*****
204 ISR1:
205 0000d5 930f push r16
206 0000d6 b70f in r16, SREG
207 0000d7 930f push r16
208
209 makedebounce:
210 0000d8 9b4b sbis PIND, 3
211 0000d9 cffe rjmp makedebounce
212 0000da e604 ldi r16, 100
213 0000db d05f rcall var_delay
214 breakdebounce:
215 0000dc 994b sbic PIND, 3
216 0000dd cffe rjmp breakdebounce
217 0000de e604 ldi r16, 100
218 0000df d05b rcall var_delay
219
220 0000e0 e002 ldi r16, (1<<INT1)
221 0000e1 bb0c out EIFR, r16
222 ;using 16 as the pbpress
223 0000e2 e100 ldi r16, $10
224 0000e3 d084 rcall fsm
225
226 0000e4 910f pop r16
227 0000e5 bf0f out SREG, r16
228 0000e6 910f pop r16
229 0000e7 9518 reti
230
```

```

231
232                                     ;*****
233                                     ;*
234                                     ;*Title: code_to_value
235                                     ;* Description:function that decodes the button
236                                     ;*
237                                     ;* Target:ATMEGA324A
238                                     ;* Number of words: 15 words
239                                     ;* Number of cycles: 12 cycles
240                                     ;*
241                                     ;* High registers modified: r16, r18, ZH, ZL
242                                     ;* Parameters:r16
243                                     ;*
244                                     ;* Returns: N/A
245                                     ;*
246                                     ;*****
247 code_to_value://table lookup
248 conversion:
249 0000e8 e0f1 ldi ZH, high(keyconvert*2)
250 0000e9 edee ldi ZL, low(keyconvert*2)
251 0000ea e000 ldi r16, $00
252 0000eb 0fe2 add ZL, r18
253 0000ec 1ff0 adc ZH, r16
254 0000ed 9124 lpm r18, Z
255 0000ee 9508 ret
256 ;table of values used for all of the pushbuttons
257 0000ef 0201
258 0000f0 0f03
259 0000f1 0504
260 0000f2 0e06 keyconvert: .db $01, $02, $03,$0F, $04, $05, $06, $0E
261 0000f3 0807
262 0000f4 0d09
263 0000f5 000a
264 0000f6 0c0b .db $07, $08, $09, $0D, $0A, $00, $0B, $0C
265

```

```
266
267
268
269
270
271
272 ;*****
273 ;*
274 ;*Title: update
275 ;* Description: meant to update the LCD buffer when doing the burst prompt
276 ;*in real time
277 ;*
278 ;* Target:ATMEGA324A
279 ;* Number of words: 5
280 ;* Number of cycles: 31
281 ;*
282 ;* High registers modified: XH, XL
283 ;* Parameters:r16
284 ;*
285 ;* Returns: N/A
286 ;*
287 ;*calls prompt burst count to reinitialize that line for the n =
288 ;*****
289
290 update:
291 0000f7 937d st X+, r23
292 0000f8 936d st X+, r22
293 0000f9 935d st X+, r21
294 0000fa d00e rcall prompt_burst_count
295 0000fb 9508 ret
296
297
298 ;*****
299 ;*
300 ;*Title: check_unneeded_buttons
```

```
301      ;* Description: Checks if unneeded buttons were pressed on the keyp ad
302      ;*
303      ;* Target:ATMEGA324A
304      ;* Number of words: 10
305      ;* Number of cycles:14
306      ;*
307      ;* High registers modified: r21
308      ;*
309      ;* Returns: N/A
310      ;*
311      ;*****
312      check_unneeded_buttons:
313 0000fc 305b      cpi r21, $0B
314 0000fd f039      breq equal
315
316 0000fe 305d      cpi r21, $0D
317 0000ff f029      breq equal
318
319 000100 305e      cpi r21, $0E
320 000101 f019      breq equal
321
322 000102 305f      cpi r21, $0F
323 000103 f009      breq equal
324
325 000104 c000      rjmp notequal
326
327      equal:
328
329      notequal:
330 000105 9508      ret
331
332      ;*****
333      ;*
334      ;*Title: convert_hex
335      ;* Description: simple program used to convert hex
```

```

336      ;*
337      ;* Target:ATMEGA324A
338      ;* Number of words: 3
339      ;* Number of cycles: 6
340      ;*
341      ;* High registers modified: r17, r21
342      ;* Parameters:r16
343      ;*
344      ;* Returns: r19
345      ;*
346      ;*****
347      convert_hex:
348      000106 e310      ldi r17, $30
349      000107 0f51      add r21, r17
350      000108 9508      ret
351
352      ;*****
353      ;*
354      ;*Title: prompt_burst_count
355      ;* Description: Creates n<space>=<space> on the LCD
356      ;* also used in order to set the X pointer pointing to the start of the
357      ;*numbers to be displayed on the LCD
358      ;*
359      ;* Target:ATMEGA324A
360      ;* Number of words: 11
361      ;* Number of cycles: 18
362      ;*
363      ;*
364      ;* High registers modified: r16, XH, XL
365      ;*
366      ;* Parameters:r16
367      ;*
368      ;* Returns: N/A
369      ;*
370      ;*****

```

```

371                                prompt_burst_count:
372 000109 e0b1                    ldi XH, high(dsp_buff_1)
373 00010a e0a6                    ldi XL, low(dsp_buff_1)
374
375 00010b e60e                    ldi r16, $6E ;displays n
376 00010c 930d                    st X+, r16
377
378 00010d e200                    ldi r16, $20;displays <space>
379 00010e 930d                    st X+, r16
380
381 00010f e30d                    ldi r16, $3D ;displays =
382 000110 930d                    st X+, r16
383
384 000111 e200                    ldi r16, $20 ;displays <space>
385 000112 930d                    st X+, r16
386 000113 df8e                    rcall update_lcd_dog
387                                ;at the end, should display n<space>=<space>
388                                ;on the LCD
389 000114 9508                    ret
390
391
392
393                                ;*****
394                                ;*
395                                ;* "BCD2bin16" - BCD to 16-Bit Binary Conversion
396                                ;*
397                                ;* This subroutine converts a 5-digit packed BCD number represented by
398                                ;* 3 bytes (fBCD2:fBCD1:fBCD0) to a 16-bit number (tbinH:tbinL).
399                                ;* MSD of the 5-digit number must be placed in the lowermost nibble of fBCD2.
400                                ;*
401                                ;* Let "abcde" denote the 5-digit number. The conversion is done by
402                                ;* computing the formula: 10(10(10a+b)+c)+d)+e.
403                                ;* The subroutine "mul10a"/"mul10b" does the multiply-and-add operation
404                                ;* which is repeated four times during the computation.
405                                ;*

```



```

406      ;* Number of words :30
407      ;* Number of cycles :108
408      ;* Low registers used :4 (copyL,copyH,mp10L/tbinL,mp10H/tbinH)
409      ;* High registers used :4 (fBCD0,fBCD1,fBCD2,adder)
410      ;*
411      ;*****
412
413      ;***** "mul10a"/"mul10b" Subroutine Register Variables
414
415      .def    copyL    =r12      ;temporary register
416      .def    copyH    =r13      ;temporary register
417      .def    mp10L    =r14      ;Low byte of number to be multiplied by 10
418      .def    mp10H    =r15      ;High byte of number to be multiplied by 10
419      .def    adder    =r19      ;value to add after multiplication
420
421      ;***** Code
422
423      mul10a:      ;***** multiplies "mp10H:mp10L" with 10 and adds "adder" high nibble
424      000115 9532      swap    adder
425      mul10b:      ;***** multiplies "mp10H:mp10L" with 10 and adds "adder" low nibble
426      000116 2cce      mov    copyL,mp10L ;make copy
427      000117 2cdf      mov    copyH,mp10H
428      000118 0cee      lsl    mp10L      ;multiply original by 2
429      000119 1cff      rol    mp10H
430      00011a 0ccc      lsl    copyL      ;multiply copy by 2
431      00011b 1cdd      rol    copyH
432      00011c 0ccc      lsl    copyL      ;multiply copy by 2 (4)
433      00011d 1cdd      rol    copyH
434      00011e 0ccc      lsl    copyL      ;multiply copy by 2 (8)
435      00011f 1cdd      rol    copyH
436      000120 0cec      add    mp10L,copyL ;add copy to original
437      000121 1cfd      adc    mp10H,copyH
438      000122 703f      andi    adder,0x0f ;mask away upper nibble of adder
439      000123 0ee3      add    mp10L,adder ;add lower nibble of adder
440      000124 f408      brcc    m10_1      ;if carry not cleared

```

```

441 000125 94f3          inc mp10H          ; inc high byte
442 000126 9508          m10_1: ret
443
444                      ;***** Main Routine Register Variables
445
446                      .def    tbinL    =r14          ;low byte of binary result (same as mp10L)
447                      .def    tbinH    =r15          ;High byte of binary result (same as mp10H)
448                      .def    fBCD0    =r16          ;BCD value digits 1 and 0
449                      .def    fBCD1    =r17          ;BCD value digits 2 and 3
450                      .def    fBCD2    =r18          ;BCD value digit 5
451
452                      ;***** Code
453
454                      BCD2bin16:
455 000127 702f          andi    fBCD2,0x0f    ;mask away upper nibble of fBCD2
456 000128 24ff          clr    mp10H
457 000129 2ee2          mov    mp10L,fBCD2    ;mp10H:mp10L = a
458 00012a 2f31          mov    adder,fBCD1
459 00012b dfe9          rcall   mul10a          ;mp10H:mp10L = 10a+b
460 00012c 2f31          mov    adder,fBCD1
461 00012d dfe8          rcall   mul10b          ;mp10H:mp10L = 10(10a+b)+c
462 00012e 2f30          mov    adder,fBCD0
463 00012f dfe5          rcall   mul10a          ;mp10H:mp10L = 10(10(10a+b)+c)+d
464 000130 2f30          mov    adder,fBCD0
465 000131 dfe4          rcall   mul10b          ;mp10H:mp10L = 10(10(10(10a+b)+c)+d)+e
466 000132 9508          ret
467
468                      ;*****
469                      ;NAME:      clr_dsp_buffs
470                      ;FUNCTION:  Initializes dsp_buffers 1, 2, and 3 with blanks (0x20)
471                      ;ASSUMES:   Three CONTIGUOUS 16-byte dram based buffers named
472                      ;          dsp_buff_1, dsp_buff_2, dsp_buff_3.
473                      ;RETURNS:   nothing.
474                      ;MODIFIES:  r25,r26, Z-ptr
475                      ;CALLS:     none

```

```

476 ;CALLED BY: main application and diagnostics
477 ;*****
478 clr_dsp_buffs:
479     000133 e390        ldi R25, 48            ; load total length of both buffer.
480     000134 e2a0        ldi R26, ' '          ; load blank/space into R26.
481     000135 e0f1        ldi ZH, high (dsp_buff_1) ; Load ZH and ZL as a pointer to 1st
482     000136 e0e6        ldi ZL, low (dsp_buff_1) ; byte of buffer for line 1.
483
484 ;set DDRAM address to 1st position of first line.
485 store_bytes:
486     000137 93a1        st Z+, R26            ; store ' ' into 1st/next buffer byte and
487                                     ; auto inc ptr to next location.
488     000138 959a        dec R25                ;
489     000139 f7e9        brne store_bytes      ; cont until r25=0, all bytes written.
490     00013a 9508        ret
491
492 ;*****
493
494 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
495 ;DELAY FUNCTIONS
496 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
497
498
499 ;*****
500 ;*
501 ;*Title: var_delay
502 ;* Description: Creates a delay for the ATMEGA324(has to run @ 1Mhz CLK)
503 ;*
504 ;* Target:ATMEGA324A
505 ;* Number of words:6
506 ;* Number of cycles: 25248 cycles (when n = 32 and m = 255)
507 ;* n = inner loop variable and m = outer loop variable
508 ;* 3(nm+m+1)= total number of cycles
509 ;* When n=0, then number of cycles : 3(nm+m+n+3)
510 ;*

```

```

511                                     ;* High registers modified: r16, r17
512                                     ;* Parameters:r16
513                                     ;*
514                                     ;* Returns: N/A
515                                     ;*
516                                     ;*Delay provided should be around n*0.1ms
517                                     ;*As stated before, n is the number inputted into r16
518                                     ;*****
519                                     var_delay: ;delay for ATmega324 @ 1MHz = r16 * 0.1 ms
520                                     outer_loop:
521 00013b e210                           ldi r17, 32
522                                     inner_loop:
523 00013c 951a                           dec r17
524 00013d f7f1                           brne inner_loop
525 00013e 950a                           dec r16
526 00013f f7d9                           brne outer_loop
527 000140 9508                           ret
528
529
530                                     ;*
531                                     ;*Title: var_delay_2
532                                     ;* Description: Creates a delay for the ATMEGA324(has to run @ 1Mhz CLK)
533                                     ;* Same purpose as var_delay, except that r17 is defined as 31
534                                     ;* Target:ATMEGA324A
535                                     ;* Number of words:6
536                                     ;* Number of cycles:
537                                     ;* n = inner loop variable and m = outer loop variable
538                                     ;* 3(nm+m+1)= total number of cycles
539                                     ;* When n=0, then number of cycles : 3(nm+m+n+3)
540                                     ;*
541                                     ;* High registers modified: r16, r17
542                                     ;* Parameters:r16
543                                     ;*
544                                     ;* Returns: N/A
545                                     ;*

```

```

546                                     ;*Delay provided should be around n*0.1ms
547                                     ;*As stated before, n is the number inputted into r16
548                                     ;*****
549                                     var_delay_2:
550                                     outer:
551 000141 e11f                           ldi r17, 31
552                                     inner:
553 000142 951a                           dec r17
554 000143 f7f1                           brne inner
555 000144 950a                           dec r16
556 000145 f7d9                           brne outer
557 000146 9508                           ret
558
559                                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
560                                     ;FSM
561                                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
562                                     ;*****
563                                     ;*
564                                     ;* "fsm" - Simplified Table Driven Finite State Machine
565                                     ;*
566                                     ;* Description:
567                                     ;* This table driven FSM can handle 255 or fewer input symbols.
568                                     ;*
569                                     ;* Author:      Aaron
570                                     ;* Version:     1.0
571                                     ;* Last updated: 12/1/2019
572                                     ;* Target:      ATmega324a
573                                     ;* Total number of cycles depends on the task being called, which is why it
574                                     ;* is excluded here
575                                     ;* Total number of words: 54 words
576                                     ;* Low regs modified:  r16, r18, r20, r21, r31, and r31
577                                     ;* High registers used:
578                                     ;*
579                                     ;* Parameters:      present state in r25:r24 prior to call
580                                     ;*                  input symbol in r16 prior to call

```

```
581      ;*
582      ;* Notes: Calls upon the stub of tasks given in "taskn"
583      ;*
584      ;*****
585
586      ;input symbols for example finite state machine
587      .equ i0 = $00      ;input symbols equated to numerical values ;
588      .equ i1 = $01
589      .equ i2 = $02
590      .equ i3 = $03
591      .equ i4 = $04
592      .equ i5 = $05
593      .equ i6 = $06
594      .equ i7 = $07
595      .equ i8 = $08
596      .equ i9 = $09
597      .equ UP = $0F
598      .equ DOWN = $0E
599      .equ key2nd = $0D
600      .equ CLEAR = $0A
601      .equ HELP = $0B
602      .equ ENTERED = $0C
603      .equ pbpress = $10
604
605      .equ eol = $FF      ;end of list (subtable) do not change
606
607      ;state table
608      ;each row consists of input symbol, next state address, task
609      ;subroutine address
610
611      state_table:
612
613      clear_screen:
614      000147 000a
615      000148 014d
```

```
616 000149 0186          .dw CLEAR,      input_char,      task1
617 00014a 00ff
618 00014b 0147
619 00014c 017c          .dw eol,      clear_screen,      task0
620
621          input_char:
622 00014d 000c
623 00014e 0153
624 00014f 01ab          .dw ENTERED,      check_trigger_pressed,      task3
625 000150 00ff
626 000151 014d
627 000152 018f          .dw eol,      input_char,      task2
628          check_trigger_pressed:
629 000153 0010
630 000154 0000
631 000155 01c7          .dw pbpress, 0 , task5
632 000156 00ff
633 000157 0153
634 000158 01c2          .dw eol,      check_trigger_pressed,      task4
635          normalburst_check_buttons:
636 000159 000a
637 00015a 014d
638 00015b 0186          .dw CLEAR,      input_char,      task1
639 00015c 0010
640 00015d 0159
641 00015e 01d8          .dw pbpress, normalburst_check_buttons ,task6
642 00015f 00ff
643 000160 0159
644 000161 01e0          .dw eol, normalburst_check_buttons, task7
645          continuous_check_clr:
646 000162 000a
647 000163 014d
648 000164 0186          .dw CLEAR,      input_char,      task1
649 000165 00ff
650 000166 0162
```

```
651 000167 01e0          .dw eol,          continuous_check_clr,   task7
652
653
654          fsm:
655          ;load Z with a byte pointer to the subtable corresponding to the
656          ;present state
657 000168 2fe8          mov ZL, pstatel ;load Z pointer with pstate address * 2
658 000169 0fee          add ZL, ZL ;since Z will be used as a byte pointer with the lpm instr.
659 00016a 2ff9          mov ZH, pstateh
660 00016b 1fff          adc ZH, ZH
661
662          ;search subtable rows for input symbol match
663          search:
664 00016c 9124          lpm r18, Z ;get symbol from state table
665 00016d 1720          cp r18, r16 ;compare table entry with input symbol
666 00016e f021          breq match
667
668          ;check input symbol against eol
669          check_eol:
670 00016f 3f2f          cpi r18, eol ;compare low byte of table entry with eol
671 000170 f011          breq match
672
673          nomatch:
674 000171 9636          adiw ZL, $06 ;adjust Z to point to next row of state table
675 000172 cff9          rjmp search ;continue searching
676
677          ;a match on input value to row input value has been found
678          ;the next word in this row is the next state address
679          ;the word following that is the task subroutine's address
680          match:
681          ;make preesent state equal to next state value in row
682          ;this accomplishes the stat transition
683 000173 9632          adiw ZL, $02 ;point to low byte of state address
684 000174 9185          lpm pstatel, Z+ ;copy next state addr. from table to preesent stat
685 000175 9195          lpm pstateh, Z+
```



```

686
687             ;execute the subroutine that accomplishes the task associated
688             ;with the transition
689 000176 9135    lpm r19, Z+ ;get subroutine address from state table
690 000177 9144    lpm r20, Z ;and put it in Z pointer
691 000178 2fe3    mov ZL, r19
692 000179 2ff4    mov ZH, r20
693 00017a 9509    icall ;Z pointer is now used as a word pointer
694 00017b 9508    ret
695
696
697
698             ;*****
699             ;*
700             ;* "taskn" - Stub subroutines for testing
701             ;*
702             ;* Description: These are the tasks called by the FSM state table
703             ;* Each task has a specific way of being defined, depending on where in the
704             ;* state diagram it is being used
705             ;*
706             ;* Author: Aaron Varghese
707             ;* Version: 1.0
708             ;* Last updated: 12/1/2019
709             ;* Target:ATMEGA324A
710             ;*
711             ;*
712             ;*
713             ;* Notes: The registers that is used for each task is different, and the
714             ;*number of words and cycles for each task is different, which is why the
715             ;*information is excluded
716             ;*
717             ;*****
718
719
720             ;subroutine stubs for tasks to be implemented

```

```
721
722             ;clears the screen, and sound buzzer for inputs that aren't clear
723             ;add stuff to sound buzzer
724             task0:
725             ;sounds the buzzer for all other inputs pressed besides CLR
726 00017c ef0f    ldi r16, 255
727 00017d 9a16    sbi PORTA, 6
728 00017e dfbc    rcall var_delay
729 00017f 9816    cbi PORTA, 6
730             ;clears the screen
731 000180 dee0    rcall init_spi_lcd;initialize lcd screen
732 000181 dfb1    rcall clr_dsp_buffs;displays a blank screen
733 000182 df1f    rcall update_lcd_dog;updates the screen
734
735
736             ;put FSM in initial state
737 000183 e487    ldi pstatel, LOW(clear_screen)
738 000184 e091    ldi pstateh, HIGH(clear_screen)
739
740 000185 9508    ret
741
742             ;displays the prompt, when clear is pressed
743             task1:
744 000186 e002    ldi r16, 2
745 000187 9300 0104 sts normal_flag, r16
746
747 000189 e002    ldi r16, 2
748 00018a 9300 0105 sts continuous_flag, r16
749
750             ;prompt for the burst count
751 00018c df7c    rcall prompt_burst_count
752 00018d df14    rcall update_lcd_dog
753 00018e 9508    ret
754
755             ;input numbers into the screen, and sound buzzer for
```

```

756                                     ;characters that aren't digits
757                                     task2:
758                                     ;getting the value of the burst count
759                                     ;-----
760                                     ;meant to run in an infinite loop getting values
761                                     ;until enter is pressed
762                                     ;r23-->MSB
763                                     ;r21-->LSB
764                                     inner_loop1:
765 00018f 2f20                         mov r18, r16
766
767 000190 302c                         cpi r18, ENTERED
768 000191 f099                         breq end
769
770 000192 302f                         cpi r18,UP
771 000193 f091                         breq endwithbuzzer
772
773 000194 302e                         cpi r18,DOWN
774 000195 f081                         breq endwithbuzzer
775
776 000196 302d                         cpi r18,key2nd
777 000197 f071                         breq endwithbuzzer
778
779 000198 302b                         cpi r18,HELP
780 000199 f061                         breq endwithbuzzer
781
782 00019a 302a                         cpi r18,CLEAR
783 00019b f051                         breq endwithbuzzer
784
785 00019c 3120                         cpi r18, pbpress
786 00019d f041                         breq endwithbuzzer
787
788 00019e 2f76                         mov r23, r22
789 00019f 2f65                         mov r22, r21
790

```

```

791 0001a0 2f52      mov r21, r18
792
793 0001a1 df64      rcall convert_hex
794
795
796 0001a2 df66      rcall prompt_burst_count
797 0001a3 df53      rcall update
798 0001a4 defd      rcall update_lcd_dog
799                ;when the enter key is pressed
800                end:
801 0001a5 9508      ret
802
803                ;turn on buzzer for all other inputs that
804                ;can't be used
805                endwithbuzzer:
806 0001a6 ef0f      ldi r16, 255
807 0001a7 9a16      sbi PORTA, 6
808 0001a8 df92      rcall var_delay
809 0001a9 9816      cbi PORTA, 6
810 0001aa 9508      ret
811
812
813                ;saves the setting (both binary and bcd setting)
814                task3:
815                ;added in case enter is pressed without inputting
816                ;numbers (000 is default)
817 0001ab e0d1      ldi YH, high(burst_count_bcd_setting)
818 0001ac e0c0      ldi YL, low(burst_count_bcd_setting)
819 0001ad df5b      rcall prompt_burst_count
820 0001ae df48      rcall update
821 0001af def2      rcall update_lcd_dog
822                ;if enter is pressed
823                ;storing burst_count_bcd_setting
824 0001b0 e0f1      ldi ZH, high(burst_count_bcd_setting)
825 0001b1 e0e0      ldi ZL, low(burst_count_bcd_setting)

```

```

826                                     ;convert back into unpacked hex
827 0001b2 5370                         subi r23, $30
828 0001b3 5360                         subi r22, $30
829 0001b4 5350                         subi r21, $30
830                                     ;storing it into the setting
831 0001b5 9371                         st Z+, r23
832 0001b6 9361                         st Z+, r22
833 0001b7 9351                         st Z+, r21
834
835                                     ;prepping for the BCD2BIN function -->43210
836 0001b8 9562                         swap r22
837                                     ;will give BCD digits 1 and 0
838 0001b9 2f05                         mov r16, r21
839 0001ba 0f06                         add r16, r22
840                                     ;will give BCD digits 3 and 2
841 0001bb 2f17                         mov r17, r23
842                                     ;set BCD digit 4 and 5 as 0
843 0001bc e020                         ldi r18, $00
844
845 0001bd df69                         rcall BCD2bin16
846
847 0001be e0d1                         ldi YH, high(burst_count_binary_setting)
848 0001bf e0c3                         ldi YL, low(burst_count_binary_setting)
849                                     ;has the value of binary setting stored in Y
850 0001c0 82e8                         st Y, r14
851 0001c1 9508                         ret
852
853                                     ;sounds the buzzer for inputs other than pb press
854                                     task4:
855 0001c2 ef0f                         ldi r16, 255
856 0001c3 9a16                         sbi PORTA, 6
857 0001c4 df76                         rcall var_delay
858 0001c5 9816                         cbi PORTA, 6
859 0001c6 9508                         ret
860

```

```

861                                     ;check burst count number
862                                     task5:
863 0001c7 90e0 0103                     lds r14, burst_count_binary_setting
864 0001c9 2d6e                         mov r22, r14
865 0001ca 3060                         cpi r22, 0
866 0001cb f031                         breq continuous_flag_set
867
868                                     normal_flag_set:
869 0001cc e001                         ldi r16,1
870 0001cd 9300 0104                     sts normal_flag, r16
871 0001cf e589                         ldi pstatel, low(normalburst_check_buttons)
872 0001d0 e091                         ldi pstateh, high(normalburst_check_buttons)
873 0001d1 9508                         ret
874
875                                     continuous_flag_set:
876 0001d2 e001                         ldi r16,1
877 0001d3 9300 0105                     sts continuous_flag, r16
878 0001d5 e682                         ldi pstatel, low(continuous_check_clr)
879 0001d6 e091                         ldi pstateh, high(continuous_check_clr)
880 0001d7 9508                         ret
881
882                                     ;reinitialize burst setting
883                                     task6:
884 0001d8 e0d1                         ldi YH, high(burst_count_binary_setting)
885 0001d9 e0c3                         ldi YL, low(burst_count_binary_setting)
886                                     ;has the value of binary setting stored in Y
887 0001da 80e8                         ld r14, Y
888 0001db 2d6e                         mov r22, r14
889
890                                     ;3 means that you redo the burst setting
891 0001dc e003                         ldi r16, 3
892 0001dd 9300 0104                     sts normal_flag, r16
893 0001df 9508                         ret
894
895                                     ;meant to sound the buzzer for all other

```

```

896             ;inputs besides pbpress and CLR
897             task7:
898 0001e0 ef0f      ldi r16, 255
899 0001e1 9a16      sbi PORTA, 6
900 0001e2 df58      rcall var_delay
901 0001e3 9816      cbi PORTA, 6
902
903
904 RESOURCE USE INFORMATION
905 -----
906
907 Notice:
908 The register and instruction counts are symbol table hit counts,
909 and hence implicitly used resources are not counted, eg, the
910 'lpm' instruction without operands implicitly uses r0 and z,
911 none of which are counted.
912
913 x,y,z are separate entities in the symbol table and are
914 counted separately from r26..r31 here.
915
916 .dseg memory usage only counts static data declared with .byte
917
918 "ATmega324A" register use summary:
919 x : 7 y : 2 z : 13 r0 : 0 r1 : 0 r2 : 0 r3 : 0 r4 : 0
920 r5 : 0 r6 : 0 r7 : 0 r8 : 0 r9 : 0 r10: 0 r11: 0 r12: 5
921 r13: 5 r14: 10 r15: 5 r16: 126 r17: 10 r18: 19 r19: 9 r20: 10
922 r21: 12 r22: 14 r23: 8 r24: 10 r25: 8 r26: 3 r27: 1 r28: 3
923 r29: 3 r30: 13 r31: 11
924 Registers used: 23 out of 35 (65.7%)
925
926 "ATmega324A" instruction use summary:
927 .lds : 0 .sts : 0 adc : 3 add : 6 adiw : 2 and : 0
928 andi : 3 asr : 0 bclr : 0 bld : 0 brbc : 0 brbs : 0
929 brcc : 1 brcs : 0 break : 0 breq : 19 brge : 0 brhc : 0
930 brhs : 0 brid : 0 brie : 0 brlo : 0 brlt : 0 brmi : 0

```

```

931 brne : 14 brpl : 0 brsh : 0 brtc : 0 brts : 0 brvc : 0
932 brvs : 0 bset : 0 bst : 0 call : 0 cbi : 9 cbr : 0
933 clc : 0 clh : 0 cli : 0 cln : 0 clr : 1 cls : 0
934 clt : 0 clv : 0 clz : 0 com : 0 cp : 1 cpc : 0
935 cpi : 18 cpse : 0 dec : 12 eor : 0 fmul : 0 fmuls : 0
936 fmulsu: 0 icall : 1 ijmp : 0 in : 12 inc : 1 jmp : 2
937 ld : 4 ldd : 0 ldi : 87 lds : 6 lpm : 9 lsl : 4
938 lsr : 0 mov : 21 movw : 0 mul : 0 muls : 0 mulsu : 0
939 neg : 0 nop : 4 or : 0 ori : 0 out : 13 pop : 10
940 push : 10 rcall : 67 ret : 29 reti : 2 rjmp : 8 rol : 4
941 ror : 0 sbc : 0 sbci : 0 sbi : 13 sbic : 1 sbis : 1
942 sbiw : 0 sbr : 0 sbrc : 0 sbrs : 2 sec : 0 seh : 0
943 sei : 1 sen : 0 ser : 0 ses : 0 set : 0 sev : 0
944 sez : 0 sleep : 0 spm : 0 st : 12 std : 0 sts : 10
945 sub : 0 subi : 3 swap : 3 tst : 0 wdr : 0

```

946 Instructions used: 40 out of 113 (35.4%)

947

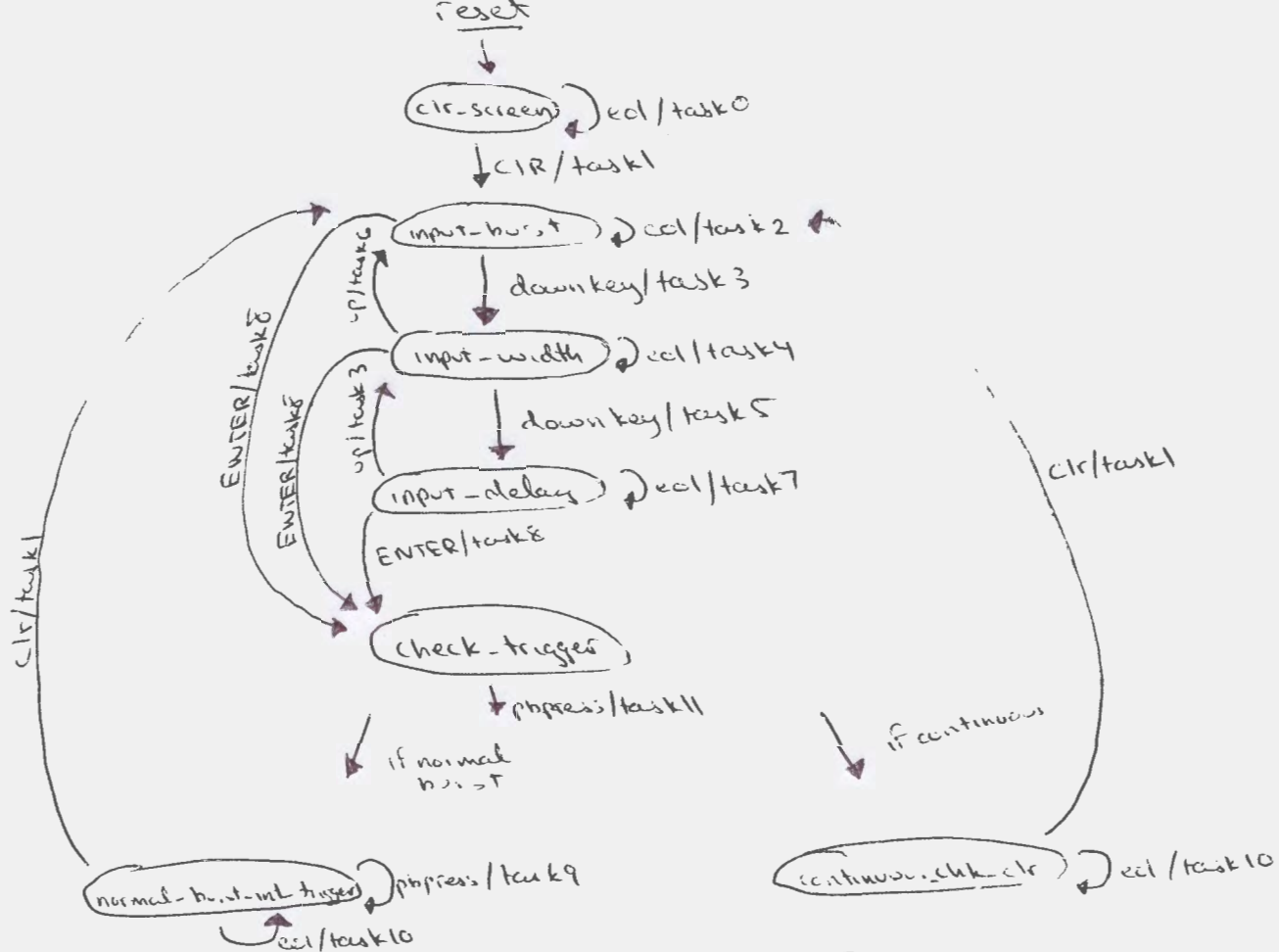
948 "ATmega324A" memory use summary [bytes]:

| 949 Segment | Begin | End | Code | Data | Used | Size | Use% |
|-------------|----------|----------|------|------|------|-------|------|
| 950 ----- | | | | | | | |
| 951 [.cseg] | 0x000000 | 0x0003ca | 888 | 82 | 970 | 32768 | 3.0% |
| 952 [.dseg] | 0x000100 | 0x000136 | 0 | 54 | 54 | 2048 | 2.6% |
| 953 [.eseg] | 0x000000 | 0x000000 | 0 | 0 | 0 | 1024 | 0.0% |

954

955 Assembly complete, 0 errors, 2 warnings

956



task0 --> sounds ~~all~~ buzzer for inputs press (not CLR), clr screen

task1 --> displays all prompts, makes burst line active

task2 --> inputs digits into display buffer (burst) sounds buzzer for other digits besides ENTER

task3 --> width line active (clr active line for all other lines)

task4 --> inputs digits into width line, sounds buzzer for digits (not ENTER)

task5 --> delay line active

task6 --> burst line active

task7 --> input digits into delay line, sounds buzzer for digits (not ENTER)

task8 --> saves all settings

task9 --> reinitialize burst settings

task10 --> sounds buzzer.

task11 --> check if burst condition is 0 or normal.

```

...11\ppg_IV_fsm_extra\ppg_IV_fsm_extra\ppg_IV_fsm_extra\Debug\ppg_IV_fsm_extra.lss 1
1
2 AVRASM ver. 2.2.7 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11 2
   \ppg_IV_fsm_extra\ppg_IV_fsm_extra\ppg_IV_fsm_extra\main.asm Thu Dec 05 18:49:39 2019
3
4 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm_extra 2
   \ppg_IV_fsm_extra\ppg_IV_fsm_extra\main.asm(13): Including file 'C:/Program Files (x86)\Atmel\Studio \7.0 2
   \Packs\atmel\ATmega_DFP\1.2.150\avrasm\inc\m324adef.inc'
5 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm_extra 2
   \ppg_IV_fsm_extra\ppg_IV_fsm_extra\main.asm(167): Including file 'C:\Users\Aaron\Desktop\College\Aar on\Junior 2
   year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm_extra\ppg_IV_fsm_extra\ppg_IV_fsm_extra 2
   \lcd_dog_asm_driver_m324a.inc'
6 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm_extra 2
   \ppg_IV_fsm_extra\ppg_IV_fsm_extra\main.asm(168): Including file 'C:\Users\Aaron\Desktop\College\Aar on\Junior 2
   year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm_extra\ppg_IV_fsm_extra\ppg_IV_fsm_extra\subroutines .inc'
7 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm_extra 2
   \ppg_IV_fsm_extra\ppg_IV_fsm_extra\main.asm(502): warning: Register r14 already defined by the .DEF directive
8 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm_extra 2
   \ppg_IV_fsm_extra\ppg_IV_fsm_extra\main.asm(503): warning: Register r15 already defined by the .DEF directive
9 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm_extra 2
   \ppg_IV_fsm_extra\ppg_IV_fsm_extra\main.asm(13): Including file 'C:/Program Files (x86)\Atmel\Studio \7.0 2
   \Packs\atmel\ATmega_DFP\1.2.150\avrasm\inc\m324adef.inc'
10 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm_extra 2
   \ppg_IV_fsm_extra\ppg_IV_fsm_extra\main.asm(167): Including file 'C:\Users\Aaron\Desktop\College\Aar on\Junior 2
   year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm_extra\ppg_IV_fsm_extra\ppg_IV_fsm_extra 2
   \lcd_dog_asm_driver_m324a.inc'
11 C:\Users\Aaron\Desktop\College\Aaron\Junior year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm_extra 2
   \ppg_IV_fsm_extra\ppg_IV_fsm_extra\main.asm(168): Including file 'C:\Users\Aaron\Desktop\College\Aar on\Junior 2
   year\Fall 2019\ESE 280\Labs\Prelab11\ppg_IV_fsm_extra\ppg_IV_fsm_extra\ppg_IV_fsm_extra\subroutines .inc'
12
13
14 ; ppg_IV_fsm_extra.asm
15 ;
16 ; Created: 12/1/2019 11:33:18 PM
17 ; Author : Aaron
18 ;Target:ATMEGA324A

```

```

19      ; Created: 11/3/2019 11:16:51 PM
20      ;Description: Using the ATMEGA324A, we will be creating a system
21      ;that would allow the user to implement functionality from lab9
22      ;using interrupts and the fsm table
23
24      .list
25      `
26
27      .def pstatel = r24 ;low byte of present state address
28      .def pstateh = r25 ;high byte of present state address
29
30      .dseg
31 000100      burst_count_bcd_setting:      .byte 3 ;setting in bcd
32 000103      burst_count_binary_setting:      .byte 1 ;setting in binary
33
34 000104      pulse_width_bcd_setting:      .byte 3 ;setting in bcd
35 000107      pulse_width_binary_setting:      .byte 1 ;setting in binary
36
37 000108      delay_time_bcd_setting:      .byte 3 ;setting in bcd
38 00010b      delay_time_binary_setting:      .byte 1 ;setting in binary
39
40 00010c      continuous_flag: .byte 1 ;flag for continuous burst
41 00010d      normal_flag: .byte 1; flag for the normal burst
42
43      .cseg
44      start:
45      .org 0x0000
46 000000 c005      rjmp init
47      .org int0addr
48 000002 940c 00d1      jmp ISR0
49      .org int1addr
50 000004 940c 00df      jmp ISR1
51
52
53      init:

```

```

54                                     ;initialize the stack pointer
55 000006 ef0f                        ldi r16, LOW(RAMEND)
56 000007 bf0d                        out SPL,r16
57 000008 e008                        ldi r16, HIGH(RAMEND)
58 000009 bf0e                        out SPH, r16
59
60                                     ;setting up PORTB
61 00000a ef0f                        ldi r16, $FF      ; set portB = output.
62 00000b b904                        out DDRB, r16      ;
63 00000c 9a2c                        sbi PORTB, 4       ; set /SS of DOG LCD = 1 (Deselected)
64
65                                     ;setting up PORTD
66 00000d e000                        ldi r16, $00;PortD is an input port
67 00000e b90a                        out DDRD, r16
68
69 00000f e003                        ldi r16, $03;initialize pull-up resistors for PD0-1
70 000010 b90b                        out PORTD, r16
71
72                                     ;setting up PORTA
73 000011 ec00                        ldi r16, $C0
74 000012 b901                        out DDRA, r16
75
76 000013 e30f                        ldi r16, $3F;initialize pull-up resistors for PD0-1
77 000014 b90b                        out PORTD, r16
78
79
80
81                                     ;initializing the interrupts
82                                     ;configures positive edge triggering
83 000015 e00f                        ldi r16, (1<<ISC00)|(1<<ISC01)|(1<<ISC10)|(1<<ISC11)
84 000016 9300 0069                  sts EICRA, r16
85                                     ;configures each of the interrupts
86 000018 e003                        ldi r16, (1<<INT0)|(1<<INT1)
87 000019 bb0d                        out EIMSK, r16
88

```

```

89 00001a d050      rcall init_spi_lcd;initialize lcd screen
90
91 00001b d12b      rcall clr_dsp_buffs;displays a blank screen
92
93 00001c d08f      rcall update_lcd_dog;updates the screen
94
95                ;put FSM in initial state
96 00001d e98a      ldi pstatel,low(clr_screen)
97 00001e e091      ldi pstateh, high(clr_screen)
98 00001f 9478      sei
99
100               main_loop:
101               ;polling to see if any of the flags are set
102 000020 9100 010d  lds r16, normal_flag
103 000022 3001      cpi r16, 1
104 000023 f029      breq normal_burst_setting
105 000024 9100 010c  lds r16, continuous_flag
106 000026 3001      cpi r16, 1
107 000027 f0d9      breq zero_burst_setting
108
109 000028 cff7      rjmp main_loop
110
111               ;if the burst is not set at 0
112               ;gives a 1 of 998 cycles positive and
113               ;gives a 0 of around 99-1000 cycles
114               normal_burst_setting:
115 000029 e000      ldi r16, 0
116 00002a 9300 010d  sts normal_flag, r16
117               ;time delay
118 00002c e0d1      ldi YH, high(delay_time_binary_setting)
119 00002d e0cb      ldi YL, low(delay_time_binary_setting)
120 00002e 80f8      ld r15, Y
121               ;pulse width
122 00002f e0d1      ldi YH, high(pulse_width_binary_setting)
123 000030 e0c7      ldi YL, low(pulse_width_binary_setting)

```

```
124 000031 80d8      ld r13, Y
125                output_normal_burst_setting:
126 000032 2d0d      mov r16, r13
127 000033 9a17      sbi PORTA, 7
128 000034 d126      rcall var_delay
129 000035 9817      cbi PORTA,7
130 000036 2d0f      mov r16, r15
131 000037 d123      rcall var_delay
132 000038 956a      dec r22
133 000039 f7c1      brne output_normal_burst_setting
134
135                polling:
136                ;check and see if setting is reinitialized
137 00003a 9100 010d  lds r16, normal_flag
138 00003c 3003      cpi r16, 3
139 00003d f359      breq normal_burst_setting
140
141                ;check and see if CLR is pressed
142 00003e 9100 010d  lds r16, normal_flag
143 000040 3002      cpi r16, 2
144 000041 f2f1      breq main_loop
145
146 000042 f7b9      brne polling
147
148                ;if the burst happens to be set at 0
149                zero_burst_setting:
150 000043 e000      ldi r16, 0
151 000044 9300 010c  sts continuous_flag, r16
152                ;time delay
153 000046 e0d1      ldi YH, high(delay_time_binary_setting)
154 000047 e0cb      ldi YL, low(delay_time_binary_setting)
155 000048 80f8      ld r15, Y
156                ;pulse width
157 000049 e0d1      ldi YH, high(pulse_width_binary_setting)
158 00004a e0c7      ldi YL, low(pulse_width_binary_setting)
```

```

159 00004b 80d8      ld r13, Y
160                ;gives a 1 of 1000 cycles and a 0 of 1007 cycles
161                output_zero_burst_setting:
162 00004c 2d0d      mov r16, r13
163 00004d 9a17      sbi PORTA, 7
164 00004e d10c      rcall var_delay
165 00004f 0000      nop
166 000050 0000      nop
167 000051 9817      cbi PORTA,7
168 000052 2d0f      mov r16, r15
169 000053 d107      rcall var_delay
170 000054 9100 010c  lds r16, continuous_flag
171 000056 3002      cpi r16, 2
172 000057 f241      breq main_loop
173
174 000058 cff3      rjmp output_zero_burst_setting
175
176                .list
177
178                ;*****
179                ;*
180                ;*Title: ISR0
181                ;* Description: checks what button on the keypad is presseed
182                ;*
183                ;* Target:ATMEGA324A
184                ;* Number of words: 17 words
185                ;* Number of cycles: 25 cycles
186                ;*
187                ;* High registers modified:
188                ;*
189                ;* Returns: N/A
190                ;*
191                ;*also calls upon the code_to_value subroutine to decode the button press
192                ;* returns the keycode in r18
193                ;*****

```

```

194          ISR0:
195 0000d1 930f      push r16
196 0000d2 b70f      in r16, SREG
197 0000d3 930f      push r16
198 0000d4 b109      in r16, PIND
199 0000d5 7f00      andi r16, $F0
200 0000d6 9502      swap r16
201 0000d7 2f20      mov r18, r16
202 0000d8 d02b      rcall code_to_value
203 0000d9 2f02      mov r16, r18
204 0000da d0f8      rcall fsm
205 0000db 910f      pop r16
206 0000dc bf0f      out SREG, r16
207 0000dd 910f      pop r16
208 0000de 9518      reti
209
210          ;*****
211          ;*
212          ;*Title: ISR1
213          ;* Description: Checks to see if the pushbutton is activated
214          ;*
215          ;* Target:ATMEGA324A
216          ;* Number of words: 22
217          ;* Number of cycles: 24769
218          ;*
219          ;* Low registers modified: N/A
220          ;* High registers modified: N/A
221
222          ;* Returns: carry flag set or cleared
223          ;*
224          ;*****
225          ISR1:
226 0000df 930f      push r16
227 0000e0 b70f      in r16, SREG
228 0000e1 930f      push r16

```



```

229
230             makedebounce:
231 0000e2 9b4b      sbis PIND, 3
232 0000e3 cffe      rjmp makedebounce
233 0000e4 e604      ldi r16, 100
234 0000e5 d075      rcall var_delay
235             breakdebounce:
236 0000e6 994b      sbic PIND, 3
237 0000e7 cffe      rjmp breakdebounce
238 0000e8 e604      ldi r16, 100
239 0000e9 d071      rcall var_delay
240
241 0000ea e002      ldi r16, (1<<INT1)
242 0000eb bb0c      out EIFR, r16
243              ;using 16 as the pbpress
244 0000ec e100      ldi r16, $10
245 0000ed d0e5      rcall fsm
246
247 0000ee 910f      pop r16
248 0000ef bf0f      out SREG, r16
249 0000f0 910f      pop r16
250 0000f1 9518      reti
251              ;*****
252              ;*
253              ;*Title: decode_button
254              ;* Description: checks what button on the keypad is presseed
255              ;*
256              ;* Target:ATMEGA324A
257              ;* Number of words: 12 words
258              ;* Number of cycles: 31 cycles
259              ;*
260              ;* High registers modified: r16, r18, ZH, ZL
261              ;*
262              ;*also calls upon the code_to_value subroutine to decode the button press
263              ;*returns the decoded keycode from the table into r18

```

```

264 ;*****
265 decode_button:
266 ;check to see if a button is pressed
267 0000f2 9488 clc ;indicates that button is not pressed
268 0000f3 9b36 sbis PINC, 6
269 0000f4 9508 ret
270 0000f5 b109 in r16, PIND
271 0000f6 7f00 andi r16, $F0
272 0000f7 9502 swap r16
273 0000f8 2f20 mov r18, r16
274 0000f9 d00a rcall code_to_value
275 0000fa 9408 sec ;indicates that a key is pressed
276 ;clears the D-FF and sets it again for polling
277 0000fb 9847 cbi PORTC, 7
278 0000fc 9a47 sbi PORTC, 7
279 0000fd 9508 ret
280
281 ;*****
282 ;*
283 ;*Title: select_line
284 ;* Description: meant to show which line is currently active and
285 ;*writing inputs to the LCD
286 ;*
287 ;* Target:ATMEGA324A
288 ;* Number of words: 3 words
289 ;* Number of cycles: 7 cycles
290 ;*
291 ;* High registers modified: r18, YH, YL
292 ;*
293 ;* Returns: N/A
294 ;*
295 ;*****
296 select_line:
297 0000fe e223 ldi r18, '#'
298 0000ff 872b std Y+11, r18

```

```

299 000100 9508      ret
300
301                ;*****
302                ;*
303                ;*Title: deselect_line
304                ;* Description: meant to show which line is currently inactive
305                ;* Target:ATMEGA324A
306                ;* Number of words: 3 words
307                ;* Number of cycles: 7 cycles
308                ;*
309                ;* High registers modified: r18, YH, YL
310                ;*
311                ;* Returns: N/A
312                ;*
313                ;*****
314                deselect_line:
315 000101 e220      ldi r18, ' '
316 000102 872b      std Y+11, r18
317 000103 9508      ret
318
319
320                ;*****
321                ;*
322                ;*Title: code_to_value
323                ;* Description:function that decodes the button
324                ;*
325                ;* Target:ATMEGA324A
326                ;* Number of words: 15 words
327                ;* Number of cycles: 12 cycles
328                ;*
329                ;* High registers modified: r16, r18, ZH, ZL
330                ;* Parameters:r16
331                ;*
332                ;* Returns: N/A
333                ;*

```

```

334 ;*****
335 code_to_value://table lookup
336 conversion:
337 000104 e0f2 ldi ZH, high(keyconvert*2)
338 000105 e1e6 ldi ZL, low(keyconvert*2)
339 000106 e000 ldi r16, $00
340 000107 0fe2 add ZL, r18
341 000108 1ff0 adc ZH, r16
342 000109 9124 lpm r18, Z
343 00010a 9508 ret
344 ;table of values used for all of the pushbuttons
345 00010b 0201
346 00010c 0f03
347 00010d 0504
348 00010e 0e06 keyconvert: .db $01, $02, $03,$0F, $04, $05, $06, $0E
349 00010f 0807
350 000110 0d09
351 000111 000a
352 000112 0c0b .db $07, $08, $09, $0D, $0A, $00, $0B, $0C
353
354
355
356
357 ;*****
358 ;*
359 ;*Title: keep_values
360 ;* Description: meant to take in the values in a line after switchi ng in
361 ;* between lines
362 ;*used to prevent lines from being written with other line's values in them
363 ;* Target:ATMEGA324A
364 ;* Number of words: 4 words
365 ;* Number of cycles: 10 cycles
366 ;*
367 ;* High registers modified: r21,r22,r23
368 ;*
```

```

369      ;* Returns: Previous values of the a current line
370      ;*
371      ;* meant to be used after initializing the prompt for a specific li ne
372      ;* using either prompt_burst_count or prompt_pulse_width_count
373      ;*****
374      keep_values:
375 000113 9179      ld r23, Y+
376 000114 9169      ld r22, Y+
377 000115 9159      ld r21, Y+
378 000116 9508      ret
379
380
381
382      ;*****
383      ;*
384      ;*Title: check_unneeded_buttons
385      ;* Description: Checks if unneeded buttons were pressed on the keyp ad
386      ;* Unneeded buttons are now the 2nd and HELP buttons
387      ;*
388      ;* Target:ATMEGA324A
389      ;* Number of words: 10 words
390      ;* Number of cycles:14 cycles
391      ;*
392      ;* High registers modified: r18
393      ;*
394      ;* Returns: N/A
395      ;*
396      ;*****
397      check_unneeded_buttons:
398 000117 302b      cpi r18, $0B
399 000118 f019      breq equal
400
401 000119 302d      cpi r18, $0D
402 00011a f009      breq equal
403

```

```

404 00011b c000      rjmp notequal
405
406      equal:
407
408      notequal:
409 00011c 9508      ret
410
411      ;*****
412      ;*
413      ;*Title: convert_hex
414      ;* Description: simple program used to convert hex
415      ;*
416      ;* Target:ATMEGA324A
417      ;* Number of words: 3 words
418      ;* Number of cycles: 6 cycles
419      ;*
420      ;* High registers modified: r17, r21
421      ;* Parameters:r16
422      ;*
423      ;* Result is in: r21
424      ;*
425      ;*****
426      convert_hex:
427 00011d e310      ldi r17, $30
428 00011e 0f51      add r21, r17
429 00011f 9508      ret
430
431
432
433
434      ;*****
435      ;*
436      ;*Title: check_internal_trigger
437      ;* Description:checking if Q at PA3 is set
438      ;*

```

```

439      ;* Target:ATMEGA324A
440      ;* Number of words: 9 words
441      ;* Number of cycles: 24769 cycles
442      ;*
443      ;* High registers modified: r16
444      ;* Returns: carry flag set or cleared
445      ;*
446      ;*****
447      check_internal_trigger:
448      000120 9488      clc ;carry clear means that the pushbutton is not pressed
449      000121 9b03      sbis PINA,3
450      000122 9508      ret;pushbutton is not pressed
451
452      000123 9408      sec; pushbutton has been pressed
453      000124 9812      cbi PORTA, 2
454      000125 ef0a      ldi r16, 250
455      000126 d034      rcall var_delay
456      000127 9a12      sbi PORTA, 2
457      000128 9508      ret
458
459
460      ;*****
461      ;*
462      ;* "BCD2bin16" - BCD to 16-Bit Binary Conversion
463      ;*
464      ;* This subroutine converts a 5-digit packed BCD number represented by
465      ;* 3 bytes (fBCD2:fBCD1:fBCD0) to a 16-bit number (tbinH:tbinL).
466      ;* MSD of the 5-digit number must be placed in the lowermost nibble of fBCD2.
467      ;*
468      ;* Let "abcde" denote the 5-digit number. The conversion is done by
469      ;* computing the formula: 10(10(10(10a+b)+c)+d)+e.
470      ;* The subroutine "mul10a"/"mul10b" does the multiply-and-add operation
471      ;* which is repeated four times during the computation.
472      ;*
473      ;* Number of words :30

```

```

474      ;* Number of cycles      :108
475      ;* Low registers used   :4 (copyL,copyH,mp10L/tbinL,mp10H/tbinH)
476      ;* High registers used  :4 (fBCD0,fBCD1,fBCD2,adder)
477      ;*
478      ;*****
479
480      ;***** "mul10a"/"mul10b" Subroutine Register Variables
481
482      .def    copyL    =r12      ;temporary register
483      .def    copyH    =r13      ;temporary register
484      .def    mp10L    =r14      ;Low byte of number to be multiplied by 10
485      .def    mp10H    =r15      ;High byte of number to be multiplied by 10
486      .def    adder    =r19      ;value to add after multiplication
487
488      ;***** Code
489
490      mul10a:      ;***** multiplies "mp10H:mp10L" with 10 and adds "adder" high
491      ;nibble
492      000129 9532      swap      adder
493      mul10b:      ;***** multiplies "mp10H:mp10L" with 10 and adds "adder" low
494      ;nibble
495      00012a 2cce      mov copyL,mp10L ;make copy
496      00012b 2cdf      mov copyH,mp10H
497      00012c 0cee      lsl mp10L      ;multiply original by 2
498      00012d 1cff      rol mp10H
499      00012e 0ccc      lsl copyL      ;multiply copy by 2
500      00012f 1cdd      rol copyH
501      000130 0ccc      lsl copyL      ;multiply copy by 2 (4)
502      000131 1cdd      rol copyH
503      000132 0ccc      lsl copyL      ;multiply copy by 2 (8)
504      000133 1cdd      rol copyH
505      000134 0cec      add mp10L,copyL ;add copy to original
506      000135 1cfd      adc mp10H,copyH
507      000136 703f      andi  adder,0x0f ;mask away upper nibble of adder
508      000137 0ee3      add mp10L,adder ;add lower nibble of adder

```



```

509 000138 f408          brcc    m10_1      ;if carry not cleared
510 000139 94f3          inc     mp10H      ; inc high byte
511 00013a 9508          m10_1: ret
512
513                      ;***** Main Routine Register Variables
514
515                      .def     tbinL    =r14      ;Low byte of binary result (same as mp10L)
516                      .def     tbinH    =r15      ;High byte of binary result (same as mp10H)
517                      .def     fBCD0    =r16      ;BCD value digits 1 and 0
518                      .def     fBCD1    =r17      ;BCD value digits 2 and 3
519                      .def     fBCD2    =r18      ;BCD value digit 5
520
521                      ;***** Code
522
523                      BCD2bin16:
524 00013b 702f          andi     fBCD2,0x0f    ;mask away upper nibble of fBCD2
525 00013c 24ff          clr     mp10H
526 00013d 2ee2          mov     mp10L,fBCD2 ;mp10H:mp10L = a
527 00013e 2f31          mov     adder,fBCD1
528 00013f dfe9          rcall   mul10a      ;mp10H:mp10L = 10a+b
529 000140 2f31          mov     adder,fBCD1
530 000141 dfe8          rcall   mul10b      ;mp10H:mp10L = 10(10a+b)+c
531 000142 2f30          mov     adder,fBCD0
532 000143 dfe5          rcall   mul10a      ;mp10H:mp10L = 10(10(10a+b)+c)+d
533 000144 2f30          mov     adder,fBCD0
534 000145 dfe4          rcall   mul10b      ;mp10H:mp10L = 10(10(10(10a+b)+c)+d)+e
535 000146 9508          ret
536
537                      ;*****
538                      ;NAME:      clr_dsp_buffers
539                      ;FUNCTION:  Initializes dsp_buffers 1, 2, and 3 with blanks (0x20)
540                      ;ASSUMES:   Three CONTIGUOUS 16-byte dram based buffers named
541                      ;          dsp_buff_1, dsp_buff_2, dsp_buff_3.
542                      ;RETURNS:   nothing.
543                      ;MODIFIES:  r25,r26, Z-ptr

```

```

544 ;CALLS: none
545 ;CALLED BY: main application and diagnostics
546 ;*****
547 clr_dsp_buffs:
548 000147 e390 ldi R25, 48 ; load total length of both buffer.
549 000148 e2a0 ldi R26, ' ' ; load blank/space into R26.
550 000149 e0f1 ldi ZH, high (dsp_buff_1) ; Load ZH and ZL as a pointer to 1st
551 00014a e0ee ldi ZL, low (dsp_buff_1) ; byte of buffer for line 1.
552
553 ;set DDRAM address to 1st position of first line.
554 store_bytes:
555 00014b 93a1 st Z+, R26 ; store ' ' into 1st/next buffer byte and
556 ; auto inc ptr to next location.
557 00014c 959a dec R25 ;
558 00014d f7e9 brne store_bytes ; cont until r25=0, all bytes written.
559 00014e 9508 ret
560 ;*****
561
562
563 ;*****
564 ;NAME: prepare_for_BCD2bin
565 ;FUNCTION: prepares for the use of the BCD2bin function
566 ;RETURNS:
567 ;MODIFIES: Zh, ZL, r16, r17, r18
568 ;cycles: 18 cycles
569 ;words: 11 words
570 ;(MSB)(MIDDLE DIGIT)(LSB)
571 ; r16-----> (MIDDLE DIGIT)(LSB)
572 ; r17-----> (0)(MSB)
573 ; r18-----> (0)(0)
574 ;*****
575 prepare_for_BCD2bin:
576 ;convert back into unpacked hex
577 00014f 5370 subi r23, $30
578 000150 5360 subi r22, $30

```

```

579 000151 5350      subi r21, $30
580                ;storing it into the setting
581 000152 9371      st Z+, r23
582 000153 9361      st Z+, r22
583 000154 9351      st Z+, r21
584
585                ;prepping for the BCD2BIN function -->43210
586 000155 9562      swap r22
587                ;will give BCD digits 1 and 0
588 000156 2f05      mov r16, r21
589 000157 0f06      add r16, r22
590                ;will give BCD digits 3 and 2
591 000158 2f17      mov r17, r23
592                ;set BCD digit 4 and 5 as 0
593 000159 e020      ldi r18, $00
594 00015a 9508      ret
595
596                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
597                ;DELAY FUNCTIONS
598                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
599
600
601                ;*****
602                ;*
603                ;*Title: var_delay
604                ;* Description: Creates a delay for the ATMEGA324(has to run @ 1Mhz CLK)
605                ;*
606                ;* Target:ATMEGA324A
607                ;* Number of words:6
608                ;* Number of cycles: 25248 cycles (when n = 32 and m = 255)
609                ;* n = inner loop variable and m = outer loop variable
610                ;* 3(nm+m+1)= total number of cycles
611                ;*
612                ;* High registers modified: r16, r17
613                ;* Parameters:r16

```

```

614      ;*
615      ;* Returns: N/A
616      ;*
617      ;*Delay provided should be around n*0.1ms
618      ;*As stated before, n is the number inputted into r16
619      ;*****
620      var_delay: ;delay for ATmega324 @ 1MHz = r16 * 0.1 ms
621      outer_loop:
622 00015b e210      ldi r17, 32
623      inner_loop:
624 00015c 951a      dec r17
625 00015d f7f1      brne inner_loop
626 00015e 950a      dec r16
627 00015f f7d9      brne outer_loop
628 000160 9508      ret
629
630
631      ;*
632      ;*Title: var_delay_2
633      ;* Description: Creates a delay for the ATMEGA324(has to run @ 1Mhz CLK)
634      ;* Same purpose as var_delay, except that r17 is defined as 31
635      ;* Target:ATMEGA324A
636      ;* Number of words:6
637      ;* Number of cycles:
638      ;* n = inner loop variable and m = outer loop variable
639      ;* 3(nm+m+1)= total number of cycles
640      ;*
641      ;* High registers modified: r16, r17
642      ;* Parameters:r16
643      ;*
644      ;* Returns: N/A
645      ;*
646      ;*Delay provided should be around n*0.1ms
647      ;*As stated before, n is the number inputted into r16
648      ;*****

```

```

649                                     var_delay_2:
650                                     outer:
651 000161 e11f                           ldi r17, 31
652                                     inner:
653 000162 951a                           dec r17
654 000163 f7f1                           brne inner
655 000164 950a                           dec r16
656 000165 f7d9                           brne outer
657 000166 9508                           ret
658
659                                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
660                                     ;PROMPTS AND UPDATES FOR THE PROMPTS
661                                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
662
663                                     ;*****
664                                     ;*
665                                     ;*Title: prompt_burst_count
666                                     ;* Description: Creates n<space>=<space> on the LCD
667                                     ;* also used in order to set the X pointer pointing to the start of the
668                                     ;*numbers to be displayed on the LCD
669                                     ;*
670                                     ;* Target:ATMEGA324A
671                                     ;* Number of words: 11 words
672                                     ;* Number of cycles: 6313 cycles
673                                     ;*
674                                     ;*
675                                     ;* High registers modified: r16, YH, YL
676                                     ;*
677                                     ;* Parameters:r16
678                                     ;*
679                                     ;* Returns: N/A
680                                     ;*
681                                     ;*****
682                                     prompt_burst_count:
683 000167 e0d1                           ldi YH, high(dsp_buff_1)

```

```

684 000168 e0ce      ldi YL, low(dsp_buff_1)
685
686 000169 e60e      ldi r16, $6E ;displays n
687 00016a 9309      st Y+, r16
688
689 00016b e200      ldi r16, $20;displays <space>
690 00016c 9309      st Y+, r16
691
692 00016d e30d      ldi r16, $3D ;displays =
693 00016e 9309      st Y+, r16
694
695 00016f e200      ldi r16, $20 ;displays <space>
696 000170 9309      st Y+, r16
697 000171 df3a      rcall update_lcd_dog
698                ;at the end, should display n<space>=<space>
699                ;on the LCD
700 000172 9508      ret
701
702                ;*****
703                ;*
704                ;*Title: update_burst_count
705                ;* Description: meant to update the LCD buffer when doing the burst prompt
706                ;*in real time (for the burst count line (line 1))
707                ;*
708                ;* Target:ATMEGA324A
709                ;* Number of words: 5 words
710                ;* Number of cycles: 6326 cycles
711                ;*
712                ;* High registers modified: YH, YL
713                ;* Parameters:r16
714                ;*
715                ;* Returns: N/A
716                ;*
717                ;*calls prompt burst count to reinitialize that line for the n =
718                ;*****

```

```

719
720             update_burst_count:
721 000173 9379      st Y+, r23
722 000174 9369      st Y+, r22
723 000175 9359      st Y+, r21
724 000176 dff0      rcall prompt_burst_count
725 000177 9508      ret
726
727
728             ;*****
729             ;*
730             ;*Title: prompt_pulse_width_count
731             ;* Description: Creates t<space>=<space> on the LCD
732             ;* also used in order to set the X pointer pointing to the start of the
733             ;*numbers to be displayed on the LCD
734             ;*
735             ;* Target:ATMEGA324A
736             ;* Number of words: 11 words
737             ;* Number of cycles: 6313 cycles
738             ;*
739             ;*
740             ;* High registers modified: r16, YH, YL
741             ;*
742             ;* Parameters:r16
743             ;*
744             ;* Returns: N/A
745             ;*
746             ;*****
747             prompt_pulse_width_count:
748 000178 e0d1      ldi YH, high(dsp_buff_2)
749 000179 e1ce      ldi YL, low(dsp_buff_2)
750
751 00017a e704      ldi r16, 't' ;displays t
752 00017b 9309      st Y+, r16
753

```

```

754 00017c e200      ldi r16, $20;displays <space>
755 00017d 9309      st Y+, r16
756
757 00017e e30d      ldi r16, $3D ;displays =
758 00017f 9309      st Y+, r16
759
760 000180 e200      ldi r16, $20 ;displays <space>
761 000181 9309      st Y+, r16
762 000182 df29      rcall update_lcd_dog
763                  ;at the end, should display t<space>=<space>
764                  ;on the LCD
765 000183 9508      ret
766
767
768                  ;*****
769                  ;*
770                  ;*Title: update_pulse_width
771                  ;* Description: meant to update the LCD buffer when doing the pulse width
772                  ;*in real time (for the pulse width line (line 2))
773                  ;*
774                  ;* Target:ATMEGA324A
775                  ;* Number of words: 5 words
776                  ;* Number of cycles: 6326 cycles
777                  ;*
778                  ;* High registers modified: YH, YL
779                  ;*
780                  ;* Returns: N/A
781                  ;*
782                  ;*calls prompt pulse width count to reinitialize that line for the t =
783                  ;*****
784
785 update_pulse_width:
786 000184 9379      st Y+, r23
787 000185 9369      st Y+, r22
788 000186 9359      st Y+, r21

```



```

789 000187 dff0      rcall prompt_pulse_width_count
790 000188 9508      ret
791
792                ;*****
793                ;*
794                ;*Title: prompt_delay_count
795                ;* Description: Creates d<space>=<space> on the LCD
796                ;* also used in order to set the X pointer pointing to the start of the
797                ;*numbers to be displayed on the LCD
798                ;*
799                ;* Target:ATMEGA324A
800                ;* Number of words: 11 words
801                ;* Number of cycles: 6313 cycles
802                ;*
803                ;*
804                ;* High registers modified: r16, YH, YL
805                ;*
806                ;* Parameters:r16
807                ;*
808                ;* Returns: N/A
809                ;*
810                ;*****
811                prompt_delay_count:
812 000189 e0d1      ldi YH, high(dsp_buff_3)
813 00018a e2ce      ldi YL, low(dsp_buff_3)
814
815 00018b e604      ldi r16, 'd' ;displays n
816 00018c 9309      st Y+, r16
817
818 00018d e200      ldi r16, $20;displays <space>
819 00018e 9309      st Y+, r16
820
821 00018f e30d      ldi r16, $3D ;displays =
822 000190 9309      st Y+, r16
823

```

```

824 000191 e200      ldi r16, $20 ;displays <space>
825 000192 9309      st Y+, r16
826 000193 df18      rcall update_lcd_dog
827                ;at the end, should display n<space>=<space>
828                ;on the LCD
829 000194 9508      ret
830
831                ;*****
832                ;*
833                ;*Title: update_delay
834                ;* Description: meant to update the LCD buffer when doing the pulse width
835                ;*in real time
836                ;*
837                ;* Target:ATMEGA324A
838                ;* Number of words: 5 words
839                ;* Number of cycles: 6326 cycles
840                ;*
841                ;* High registers modified: r23,r22, r21, YH, YL
842                ;*
843                ;* Returns: N/A
844                ;*
845                ;*calls prompt burst count to reinitialize that line for the t =
846                ;*****
847
848                update_delay:
849 000195 9379      st Y+, r23
850 000196 9369      st Y+, r22
851 000197 9359      st Y+, r21
852 000198 dff0      rcall prompt_delay_count
853 000199 9508      ret
854
855                ;*****
856                ;*
857                ;* "fsm" - Simplified Table Driven Finite State Machine
858                ;*

```

```

859      ;* Description:
860      ;* This table driven FSM can handle 255 or fewer input symbols.
861      ;*
862      ;* Author:          Ken Short
863      ;* Version:        2.0
864      ;* Last updated:   11/09/15
865      ;* Target:         ATmega16
866      ;* Number of words:
867      ;* Number of cycles:
868      ;* Low regs modified:  r16, r18, r20, r21, r31, and r31
869      ;* High registers used:
870      ;*
871      ;* Parameters:      present state in r25:r24 prior to call
872      ;*                  input symbol in r16 prior to call
873      ;*
874      ;* Notes:
875      ;*
876      ;*****
877
878      ;input symbols for example finite state machine
879      .equ i0 = $00      ;input symbols equated to numerical values ;
880      .equ i1 = $01
881      .equ i2 = $02
882      .equ i3 = $03
883      .equ i4 = $04
884      .equ i5 = $05
885      .equ i6 = $06
886      .equ i7 = $07
887      .equ i8 = $08
888      .equ i9 = $09
889      .equ UP = $0F
890      .equ DOWN = $0E
891      .equ key2nd = $0D
892      .equ CLEAR = $0A
893      .equ HELP = $0B

```

```
894 .equ ENTERED = $0C
895 .equ pbpress = $10
896 .equ eol = $FF ;end of list (subtable) do not change
897
898 ;state table for example finite state machine
899 ;each row consists of input symbol, next state address, task
900 ;subroutine address
901
902 state_table:
903
904 clr_screen:
905 00019a 000a
906 00019b 01a0
907 00019c 01ef .dw CLEAR, input_burst, task1
908 00019d 00ff
909 00019e 019a
910 00019f 01e7 .dw eol, clr_screen, task0
911
912 input_burst:
913 0001a0 000c
914 0001a1 01be
915 0001a2 026f .dw ENTERED, check_trigger, task8
916 0001a3 000e
917 0001a4 01a9
918 0001a5 021b .dw DOWN, input_width, task3
919 0001a6 00ff
920 0001a7 01a0
921 0001a8 0200 .dw eol, input_burst, task2
922
923 input_width:
924 0001a9 000c
925 0001aa 01be
926 0001ab 026f .dw ENTERED, check_trigger, task8
927 0001ac 000f
928 0001ad 01a0
```

```
929 0001ae 0249          .dw UP,          input_burst,          task6
930 0001af 000e
931 0001b0 01b5
932 0001b1 023f          .dw DOWN,          input_delay,          task5
933 0001b2 00ff
934 0001b3 01a9
935 0001b4 0225          .dw eol,    input_width,          task4
936          input_delay:
937 0001b5 000c
938 0001b6 01be
939 0001b7 026f          .dw ENTERED,          check_trigger,          task8
940 0001b8 000f
941 0001b9 01a9
942 0001ba 021b          .dw UP,          input_width,          task3
943 0001bb 00ff
944 0001bc 01b5
945 0001bd 0253          .dw eol,    input_delay,          task7
946
947          check_trigger:
948 0001be 0010
949 0001bf 0000
950 0001c0 02ac          .dw pbpress,          0,          task11
951 0001c1 00ff
952 0001c2 01be
953 0001c3 02a7          .dw eol,    check_trigger,          task10
954
955          normal_burst_int_trigger:
956 0001c4 0010
957 0001c5 01c4
958 0001c6 02a0          .dw pbpress,          normal_burst_int_trigger,          task9
959 0001c7 000a
960 0001c8 01a0
961 0001c9 01ef          .dw CLEAR,          input_burst,          task1
962 0001ca 00ff
963 0001cb 01c4
```

```

964 0001cc 02a7          .dw eol,    normal_burst_int_trigger,    task10
965
966                    continuous_chk_clr:
967 0001cd 000a
968 0001ce 01a0
969 0001cf 01ef          .dw CLEAR,    input_burst,    task1
970 0001d0 00ff
971 0001d1 01cd
972 0001d2 02a7          .dw eol,    continuous_chk_clr,    task10
973
974
975                    fsm:
976                    ;load Z with a byte pointer to the subtable corresponding to the
977                    ;present state
978 0001d3 2fe8          mov ZL, pstatel ;load Z pointer with pstate address * 2
979 0001d4 0fee          add ZL, ZL ;since Z will be used as a byte pointer with the lpm instr.
980 0001d5 2ff9          mov ZH, pstateh
981 0001d6 1fff          adc ZH, ZH
982
983                    ;search subtable rows for input symbol match
984                    search:
985 0001d7 9124          lpm r18, Z ;get symbol from state table
986 0001d8 1720          cp r18, r16 ;compare table entry with input symbol
987 0001d9 f021          breq match
988
989                    ;check input symbol against eol
990                    check_eol:
991 0001da 3f2f          cpi r18, eol ;compare low byte of table entry with eol
992 0001db f011          breq match
993
994                    nomatch:
995 0001dc 9636          adiw ZL, $06 ;adjust Z to point to next row of state table
996 0001dd cff9          rjmp search ;continue searching
997
998                    ;a match on input value to row input value has been found

```

```

999          ;the next word in this row is the next state address
1000         ;the word following that is the task subroutine's address
1001         match:
1002         ;make preseat state equal to next state value in row
1003         ;this accomplishes the stat transition
1004 0001de 9632      adiw ZL, $02 ;point to low byte of state address
1005 0001df 9185      lpm pstatel, Z+; ;copy next state addr. from table to preseat stat
1006 0001e0 9195      lpm pstateh, Z+
1007
1008         ;execute the subroutine that accomlihes the task associated
1009         ;with the transition
1010 0001e1 9135      lpm r19, Z+ ;get subroutine address from state table
1011 0001e2 9144      lpm r20, Z ;and put it in Z pointer
1012 0001e3 2fe3      mov ZL, r19
1013 0001e4 2ff4      mov ZH, r20
1014 0001e5 9509      icall ;Z pointer is now used as a word pointer
1015 0001e6 9508      ret
1016
1017
1018
1019          ;*****
1020          ;*
1021          ;* "taskn" - Stub subroutines for testing
1022          ;*
1023          ;* Description:
1024          ;* These subroutines are the tasks for the simple table driven FSM example.
1025          ;* When a program is being developed, you should start with each of these
1026          ;* subroutines consisting of just a nop and a return. You can then simulate
1027          ;* the program and verify that the transitions defined by you trans ition
1028          ;* table and original state diagram take place in response to input
1029          ;* sequences.
1030          ;*
1031          ;* Author:          Ken Short
1032          ;* Version:
1033          ;* Last updated:

```

```

1034      ;* Target:
1035      ;* Number of words:
1036      ;* Number of cycles:
1037      ;* Low registers used:
1038      ;* High registers used:
1039      ;*
1040      ;* Parameters:
1041      ;*
1042      ;* Notes:
1043      ;*
1044      ;*****
1045
1046      ;sounds the buzzer for all inputs not CLR, and clears the screen
1047      task0:
1048 0001e7 df5f      rcall clr_dsp_buffs;displays a blank screen
1049      ;rcall update_lcd_dog;updates the screen
1050
1051      ;put FSM in initial state
1052 0001e8 e98a      ldi pstatel,low(clr_screen)
1053 0001e9 e091      ldi pstateh, high(clr_screen)
1054
1055      ;sounds the buzzer
1056 0001ea ef0f      ldi r16, 255
1057 0001eb 9a16      sbi PORTA, 6
1058 0001ec df6e      rcall var_delay
1059 0001ed 9816      cbi PORTA, 6
1060 0001ee 9508      ret
1061
1062      ;displays all prompts, makes burst line active
1063      task1:
1064      ;clears all flags
1065 0001ef e002      ldi r16, 2
1066 0001f0 9300 010d      sts normal_flag, r16
1067
1068 0001f2 e002      ldi r16, 2

```



```

1069 0001f3 9300 010c      sts continuous_flag, r16
1070                          ;initializes all lines to 0
1071
1072                          ;initial values for the burst count
1073 0001f5 e350      ldi r21, $30
1074 0001f6 e360      ldi r22, $30
1075 0001f7 e370      ldi r23, $30
1076
1077                          ;setting r21,r20 and r19 to zero in case of enter
1078                          ;pressed without inputting values
1079 0001f8 df6e      rcall prompt_burst_count
1080 0001f9 df79      rcall update_burst_count
1081 0001fa df03      rcall select_line
1082
1083 0001fb df7c      rcall prompt_pulse_width_count
1084 0001fc df87      rcall update_pulse_width
1085
1086 0001fd df8b      rcall prompt_delay_count
1087 0001fe df96      rcall update_delay
1088
1089 0001ff 9508      ret
1090
1091                          ;inputs digits into burst buffer, sounds buzzer for keys not used
1092 task2:
1093                          ;-----
1094                          ;GETTING BURST COUNT
1095                          ;-----
1096                          ;getting the value of the burst count
1097                          ;meant to run in an infinite loop getting values
1098                          ;until enter is pressed
1099                          ;r23-->MSB
1100                          ;r21-->LSB
1101
1102 input1: ;corresponding to line 1 of the LCD
1103

```

```
1104 000200 2f20      mov r18, r16
1105                ;if CLR is pressed, take another input instead
1106 000201 302a      cpi r18, $0A
1107 000202 f099      breq endwithbuzzer
1108
1109                ;if pushbutton is pressed, take another input
1110 000203 3120      cpi r18, pbpress
1111 000204 f089      breq endwithbuzzer
1112
1113                ;if up arrow is pressed, ignore input
1114 000205 302f      cpi r18, $0F
1115 000206 f079      breq endwithbuzzer
1116
1117 000207 df0f      rcall check_unneeded_buttons
1118 000208 f069      breq endwithbuzzer
1119
1120 000209 2f76      mov r23, r22
1121 00020a 2f65      mov r22, r21
1122
1123 00020b 2f52      mov r21, r18
1124
1125 00020c df10      rcall convert_hex
1126 00020d df59      rcall prompt_burst_count
1127 00020e df64      rcall update_burst_count
1128 00020f deee      rcall select_line
1129 000210 df67      rcall prompt_pulse_width_count
1130 000211 deef      rcall deselect_line
1131 000212 df76      rcall prompt_delay_count
1132 000213 deed      rcall deselect_line
1133 000214 de97      rcall update_lcd_dog
1134 000215 9508      ret
1135
1136                ;sounds the buzzer for useless inputs
1137                endwithbuzzer:
1138 000216 ef0f      ldi r16, 255
```

```

1139 000217 9a16      sbi PORTA, 6
1140 000218 df42      rcall var_delay
1141 000219 9816      cbi PORTA, 6
1142 00021a 9508      ret
1143
1144
1145                      ;width line active (makes other lines inactive)
1146 task3:
1147 00021b df5c      rcall prompt_pulse_width_count
1148 00021c def6      rcall keep_values
1149 00021d df5a      rcall prompt_pulse_width_count
1150 00021e dedf      rcall select_line
1151 00021f df47      rcall prompt_burst_count
1152 000220 dee0      rcall deselect_line
1153 000221 df67      rcall prompt_delay_count
1154 000222 dede      rcall deselect_line
1155 000223 de88      rcall update_lcd_dog
1156 000224 9508      ret
1157
1158                      ;inputs digits into width buffer, sounds buzzer for keys not used
1159 task4:
1160                      ;-----
1161                      ;GETTING PULSE WIDTH
1162                      ;-----
1163
1164 input2:
1165 000225 2f20      mov r18, r16
1166                      ;if CLR is pressed, take another input instead
1167 000226 302a      cpi r18, $0A
1168 000227 f091      breq endwithbuzzer2
1169
1170                      ;if pushbutton is pressed, take another input
1171 000228 3120      cpi r18, pbpress
1172 000229 f081      breq endwithbuzzer2
1173

```

```
1174 00022a deec          rcall check_unneeded_buttons
1175 00022b f071          breq endwithbuzzer2
1176
1177 00022c 2f76          mov r23, r22
1178 00022d 2f65          mov r22, r21
1179
1180 00022e 2f52          mov r21, r18
1181
1182
1183 00022f 2f52          mov r21, r18
1184 000230 deec          rcall convert_hex
1185 000231 df35          rcall prompt_burst_count
1186 000232 dece          rcall deselect_line
1187 000233 df55          rcall prompt_delay_count
1188 000234 decc          rcall deselect_line
1189 000235 df42          rcall prompt_pulse_width_count
1190 000236 df4d          rcall update_pulse_width
1191 000237 dec6          rcall select_line
1192 000238 de73          rcall update_lcd_dog
1193
1194 000239 9508          ret
1195
1196          ;sounds the buzzer for useless inputs
1197          endwithbuzzer2:
1198 00023a ef0f          ldi r16, 255
1199 00023b 9a16          sbi PORTA, 6
1200 00023c df1e          rcall var_delay
1201 00023d 9816          cbi PORTA, 6
1202 00023e 9508          ret
1203
1204          ;delay line active
1205          task5:
1206          inner_loop3:
1207 00023f df49          rcall prompt_delay_count
1208 000240 ded2          rcall keep_values
```

```

1209 000241 df47      rcall prompt_delay_count
1210 000242 debb      rcall select_line
1211 000243 df23      rcall prompt_burst_count
1212 000244 debc      rcall deselect_line
1213 000245 df32      rcall prompt_pulse_width_count
1214 000246 deba      rcall deselect_line
1215 000247 de64      rcall update_lcd_dog
1216 000248 9508      ret
1217
1218                ;burst line active
1219                task6:
1220 000249 df1d      rcall prompt_burst_count
1221 00024a dec8      rcall keep_values
1222 00024b df1b      rcall prompt_burst_count
1223 00024c deb1      rcall select_line
1224 00024d df2a      rcall prompt_pulse_width_count
1225 00024e deb2      rcall deselect_line
1226 00024f df39      rcall prompt_delay_count
1227 000250 deb0      rcall deselect_line
1228 000251 de5a      rcall update_lcd_dog
1229 000252 9508      ret
1230
1231                ;inputs digits into delay buffer, sounds buzzer for keys not used
1232                task7:
1233                ;-----
1234                ;GETTING DELAY COUNT
1235                ;-----
1236
1237                input3;;corresponding to line 3 of the LCD
1238 000253 2f20      mov r18, r16
1239                ;if CLR is pressed, take another input instead
1240 000254 302a      cpi r18, $0A
1241 000255 f0a1      breq endwithbuzzer3
1242
1243                ;if pushbutton is pressed, take another input

```

```
1244 000256 3120      cpi r18, pbpress
1245 000257 f091      breq endwithbuzzer3
1246
1247                  ;if down arrow is pressed, read another input again
1248 000258 302e      cpi r18, $0E
1249 000259 f081      breq endwithbuzzer3
1250
1251 00025a debc      rcall check_unneeded_buttons
1252 00025b f071      breq endwithbuzzer3
1253
1254 00025c 2f76      mov r23, r22
1255 00025d 2f65      mov r22, r21
1256
1257 00025e 2f52      mov r21, r18
1258
1259
1260 00025f 2f52      mov r21, r18
1261 000260 debc      rcall convert_hex
1262 000261 df05      rcall prompt_burst_count
1263 000262 de9e      rcall deselect_line
1264 000263 df14      rcall prompt_pulse_width_count
1265 000264 de9c      rcall deselect_line
1266 000265 df23      rcall prompt_delay_count
1267 000266 df2e      rcall update_delay
1268 000267 de96      rcall select_line
1269 000268 de43      rcall update_lcd_dog
1270 000269 9508      ret
1271                  ;sounds the buzzer for useless inputs
1272                  endwithbuzzer3:
1273 00026a ef0f      ldi r16, 255
1274 00026b 9a16      sbi PORTA, 6
1275 00026c deee      rcall var_delay
1276 00026d 9816      cbi PORTA, 6
1277 00026e 9508      ret
1278
```

```
1279 ;saves all settings, if enter is pressed
1280 task8:
1281 ;this is for inputting the prompt burst count into memory
1282 00026f def7 rcall prompt_burst_count
1283 000270 de90 rcall deselect_line
1284
1285
1286 000271 def5 rcall prompt_burst_count
1287 000272 9179 ld r23, Y+
1288 000273 9169 ld r22, Y+
1289 000274 9159 ld r21, Y+
1290
1291 ;storing burst_count_bcd_setting
1292 000275 e0f1 ldi ZH, high(burst_count_bcd_setting)
1293 000276 e0e0 ldi ZL, low(burst_count_bcd_setting)
1294
1295 ;prepares for the BCD2bin
1296 ;stores teh bcd setting into SRAM and
1297 ;gets ready to input to the BCD2bin function
1298 000277 ded7 rcall prepare_for_BCD2bin
1299
1300 000278 dec2 rcall BCD2bin16
1301
1302 000279 e0d1 ldi YH, high(burst_count_binary_setting)
1303 00027a e0c3 ldi YL, low(burst_count_binary_setting)
1304 ;has the value of binary setting stored in Y
1305 00027b 82e8 st Y, r14
1306
1307 ;this is for storing the pulse width
1308 ;this is for inputting the prompt burst count into memory
1309
1310 00027c defb rcall prompt_pulse_width_count
1311 00027d de83 rcall deselect_line
1312
1313 00027e def9 rcall prompt_pulse_width_count
```

```
1314 00027f 9179      ld r23, Y+
1315 000280 9169      ld r22, Y+
1316 000281 9159      ld r21, Y+
1317
1318                ;storing burst_count_bcd_setting
1319 000282 e0f1      ldi ZH, high(pulse_width_bcd_setting)
1320 000283 e0e4      ldi ZL, low(pulse_width_bcd_setting)
1321
1322
1323 000284 deca      rcall prepare_for_BCD2bin
1324
1325 000285 deb5      rcall BCD2bin16
1326
1327 000286 e0d1      ldi YH, high(pulse_width_binary_setting)
1328 000287 e0c7      ldi YL, low(pulse_width_binary_setting)
1329                ;has the value of binary setting stored in Y
1330 000288 82e8      st Y, r14
1331
1332                ;this is for inputting the prompt delay count into memory
1333 000289 deff      rcall prompt_delay_count
1334 00028a de76      rcall deselect_line
1335
1336 00028b defd      rcall prompt_delay_count
1337 00028c 9179      ld r23, Y+
1338 00028d 9169      ld r22, Y+
1339 00028e 9159      ld r21, Y+
1340
1341                ;storing delay_time_bcd_setting
1342 00028f e0f1      ldi ZH, high(delay_time_bcd_setting)
1343 000290 e0e8      ldi ZL, low(delay_time_bcd_setting)
1344
1345                ;prepares for the BCD2bin
1346                ;stores teh bcd setting into SRAM and
1347                ;gets ready to input to the BCD2bin function
1348 000291 debd      rcall prepare_for_BCD2bin
```



```
1349
1350 000292 dea8          rcall BCD2bin16
1351
1352 000293 e0d1          ldi YH, high(delay_time_binary_setting)
1353 000294 e0cb          ldi YL, low(delay_time_binary_setting)
1354                      ;has the value of binary setting stored in Y
1355 000295 82e8          st Y, r14
1356
1357
1358                      ;taking the burst count setting
1359 000296 e0d1          ldi YH, high(burst_count_binary_setting)
1360 000297 e0c3          ldi YL, low(burst_count_binary_setting)
1361 000298 80e8          ld r14, Y
1362 000299 2d6e          mov r22, r14
1363
1364                      ;taking the pulse width count setting
1365 00029a e0d1          ldi YH, high(pulse_width_binary_setting)
1366 00029b e0c7          ldi YL, low(pulse_width_binary_setting)
1367 00029c 80d8          ld r13, Y
1368
1369                      ;taking the delay count setting
1370 00029d e0d1          ldi YH, high(delay_time_binary_setting)
1371 00029e e0cb          ldi YL, low(delay_time_binary_setting)
1372 00029f 9508          ret
1373
1374                      ;reinit burst setting
1375                      task9:
1376 0002a0 e0d1          ldi YH, high(burst_count_binary_setting)
1377 0002a1 e0c3          ldi YL, low(burst_count_binary_setting)
1378 0002a2 8168          ld r22, Y
1379
1380                      ;3 means that you redo the burst setting
1381 0002a3 e003          ldi r16, 3
1382 0002a4 9300 010d     sts normal_flag, r16
1383 0002a6 9508          ret
```

```
1384
1385
1386             ;sounds buzzer for all inputs not CLR or pbpress
1387 task10:
1388 0002a7 ef0f    ldi r16, 255
1389 0002a8 9a16    sbi PORTA, 6
1390 0002a9 deb1    rcall var_delay
1391 0002aa 9816    cbi PORTA, 6
1392 0002ab 9508    ret
1393
1394             ;check to see if the burst count is 0 or a number
1395 task11:
1396 0002ac 90e0 0103 lds r14, burst_count_binary_setting
1397 0002ae 2d6e    mov r22, r14
1398 0002af 3060    cpi r22, 0
1399 0002b0 f049    breq continuous_flag_set
1400
1401 normal_flag_set:
1402 0002b1 e001    ldi r16,1
1403 0002b2 9300 010d sts normal_flag, r16
1404 0002b4 e000    ldi r16, 0
1405 0002b5 9300 010c sts continuous_flag, r16
1406 0002b7 ec84    ldi pstatel, low(normal_burst_int_trigger)
1407 0002b8 e091    ldi pstateh, high(normal_burst_int_trigger)
1408 0002b9 9508    ret
1409
1410 continuous_flag_set:
1411 0002ba e001    ldi r16,1
1412 0002bb 9300 010c sts continuous_flag, r16
1413 0002bd e000    ldi r16,0
1414 0002be 9300 010d sts normal_flag, r16
1415 0002c0 ec8d    ldi pstatel, low(continuous_chk_clr)
1416 0002c1 e091    ldi pstateh, high(continuous_chk_clr)
1417 0002c2 9508    ret
1418
```

1419

1420

1421

1422

1423 RESOURCE USE INFORMATION

1424 -----

1425

1426 Notice:

1427 The register and instruction counts are symbol table hit counts,

1428 and hence implicitly used resources are not counted, eg, the

1429 'lpm' instruction without operands implicitly uses r0 and z,

1430 none of which are counted.

1431

1432 x,y,z are separate entities in the symbol table and are

1433 counted separately from r26..r31 here.

1434

1435 .dseg memory usage only counts static data declared with .byte

1436

1437 "ATmega324A" register use summary:

1438 x : 0 y : 45 z : 13 r0 : 0 r1 : 0 r2 : 0 r3 : 0 r4 : 0

1439 r5 : 0 r6 : 0 r7 : 0 r8 : 0 r9 : 0 r10: 0 r11: 0 r12: 5

1440 r13: 10 r14: 12 r15: 9 r16: 153 r17: 10 r18: 33 r19: 9 r20: 10

1441 r21: 20 r22: 25 r23: 16 r24: 10 r25: 8 r26: 2 r27: 0 r28: 14

1442 r29: 14 r30: 15 r31: 13

1443 Registers used: 21 out of 35 (60.0%)

1444

1445 "ATmega324A" instruction use summary:

1446 .lds : 0 .sts : 0 adc : 3 add : 6 adiw : 2 and : 0

1447 andi : 4 asr : 0 bclr : 0 bld : 0 brbc : 0 brbs : 0

1448 brcc : 1 brcs : 0 break : 0 breq : 21 brge : 0 brhc : 0

1449 brhs : 0 brid : 0 brie : 0 brlo : 0 brlt : 0 brmi : 0

1450 brne : 13 brpl : 0 brsh : 0 brtc : 0 brts : 0 brvc : 0

1451 brvs : 0 bset : 0 bst : 0 call : 0 cbi : 12 cbr : 0

1452 clc : 2 clh : 0 cli : 0 cln : 0 clr : 1 cls : 0

1453 clt : 0 clv : 0 clz : 0 com : 0 cp : 1 cpc : 0

```

1454 cpi   : 17 cpse : 0 dec   : 12 eor   : 0 fmul  : 0 fmuls : 0
1455 fmulsu: 0 icall : 1 ijmp  : 0 in    : 13 inc   : 1 jmp   : 2
1456 ld    : 22 ldd   : 0 ldi   : 121 lds   : 6 lpm   : 9 lsl   : 4
1457 lsr   : 0 mov   : 36 movw  : 0 mul   : 0 muls  : 0 mulsu : 0
1458 neg   : 0 nop   : 4 or    : 0 ori   : 0 out   : 14 pop   : 10
1459 push  : 10 rcall : 144 ret   : 47 reti  : 2 rjmp  : 9 rol   : 4
1460 ror   : 0 sbc   : 0 sbci  : 0 sbi   : 16 sbic  : 1 sbis  : 3
1461 sbiw  : 0 sbr   : 0 sbrc  : 0 sbrs  : 2 sec   : 2 seh   : 0
1462 sei   : 1 sen   : 0 ser   : 0 ses   : 0 set   : 0 sev   : 0
1463 sez   : 0 sleep : 0 spm   : 0 st    : 28 std   : 2 sts   : 10
1464 sub   : 0 subi  : 3 swap  : 4 tst   : 0 wdr   : 0

```

1465 Instructions used: 43 out of 113 (38.1%)

1466

1467 "ATmega324A" memory use summary [bytes]:

| Segment | Begin | End | Code | Data | Used | Size | Use% |
|--------------|----------|----------|------|------|------|-------|------|
| ----- | | | | | | | |
| 1470 [.cseg] | 0x000000 | 0x000586 | 1282 | 130 | 1412 | 32768 | 4.3% |
| 1471 [.dseg] | 0x000100 | 0x00013e | 0 | 62 | 62 | 2048 | 3.0% |
| 1472 [.eseg] | 0x000000 | 0x000000 | 0 | 0 | 0 | 1024 | 0.0% |

1473

1474 Assembly complete, 0 errors, 2 warnings

1475