```
 1
 2  AVRASM ver. 2.2.7  C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\
 3  ppg_III_pos_edge_ints\main.asm Thu Nov 21 20:38:22 2019
 4
 5  C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\ppg_III_pos_edge_ints\main.as m(18):
 6  Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.3.300\avrasm\
 7  inc\m324adef.inc'
 8  C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\ppg_III_pos_edge_ints\main.as m(377):
 9   warning: Register r14 already defined by the .DEF directive
10  C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\ppg_III_pos_edge_ints\main.as m(378):
11   warning: Register r15 already defined by the .DEF directive
12  C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\ppg_III_pos_edge_ints\main.as m(506):
13   Including file 'C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\
14   ppg_III_pos_edge_ints\lcd_dog_asm_driver_m324a.inc'
15  C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\ppg_III_pos_edge_ints\main.as m(18):
16  Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.3.300\avrasm\
17  inc\m324adef.inc'
18  C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\ppg_III_pos_edge_ints\main.as m(506):
19   Including file 'C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\
20   ppg_III_pos_edge_ints\lcd_dog_asm_driver_m324a.inc'
21
22
23                                  ;*
24                                  ;* Title: ppg_III_pos_edge_ints
25                                  ;* Author: Seyi Olajuyi & Bassel El Amine
26                                  ;* Version: 1.0
27                                  ;* Last updated: 2019/11/21
28                                  ;* Target: ATmega 324
29                                  ;*
30                                  ;* DESCRIPTION
31                                  ;*
32                                  ;*
33                                  ;*
34                                  ;*
35                                  ;* VERSION HISTORY
```

```
36                                  ;* 1.0 Original version
37                                  ;************************************************************* *********
38                                  .list
39
40                                  .dseg  ;The variable below are in SRAM
41  000100                          burst_count_setting_bcd:       .byte 3; setting unpacked BCD
42  ;THIS HAS THREE BTYE allocated to the variable name
43  000103                          burst_count:                   .byte 1; pulses left to
44   generated in burst
45  000104                          keyvalue:                      .byte 1; stores the
46  keyvalue into a variable
47  000105                          polling_for_keypad:                .byte 1; used to store
48  the values in the external interrupt flag register
49  000106                          polling_for_button:                .byte 1; used to store
50  the values in the external interrupt flag register
51
52
53                                  ;burst_count_setting_bcd is right most digit and
54                                  ; (burst_count_setting_bcd + 2) is the left most digit
55
56                                  .cseg
57                                  reset:
58                                  .org RESET              ;reset interrupt vector
59  000000 c004                         rjmp start          ;program starts here at reset
60                                  .org INT0addr           ;INT0 interrupt vector
61  000002 c0d6                         rjmp keypress_ISR
62                                  .org INT1addr
63  000004 c0e7                        rjmp pb_press_ISR
64
65
66                                  ;************************************************************* ****
67                                  ;************* M A I N   A P P L I C A T I O N   C O D E  ********* ****
68                                  ;************************************************************* ****
69
70
```

```
 71                                 start:
 72  000005 ef0f                        ldi r16, LOW(RAMEND)    ;initialize SP to point to top of stack
 73  000006 bf0d                        out SPL, r16
 74  000007 e008                        ldi r16, HIGH(RAMEND)
 75  000008 bf0e                        out SPH, r16
 76
 77  000009 e00f                        ldi r16, (1 << ISC00) | (1 << ISC01) | (1 << ISC10) | (1 << ISC 11)
 78  00000a 9300 0069                   sts EICRA, r16
 79  00000c e003                       ldi r16, $03        ; Enable interrupt request at INTO & INT1
 80  00000d bb0d                        out EIMSK, r16
 81
 82  00000e ef0f                        ldi r16, $ff      ; load r16 with all 1s.
 83  00000f b904                        out DDRB, r16     ; set portB = output
 84
 85  000010 e003                       ldi r16, $03        ; Set pin 0 & pin 1 to output,
 86  everyother pin is an input
 87  000011 b90a                       out DDRD, r16
 88
 89  000012 9a0f                       sbi DDRA, 7         ; Set pin 7 on PORTA to output
 90
 91  000013 9a2c                        sbi portB, 4      ; set /SS of DOG LCD = 1 (Deselected)
 92
 93  000014 d126                        rcall init_lcd_dog        ; init display, using SPI serial inte rface
 94  000015 d059                       rcall clr_dsp_buffs       ; clear all three buffer lines
 95  000016 d142                       rcall update_lcd_dog      ; update the display
 96
 97  000017 e0d1                       ldi YH, high (dsp_buff_1) ; Load YH and YL as a pointer to 1st
 98  000018 e0c7                        ldi YL, low (dsp_buff_1)  ; byte of dsp_buff_1 (Note - assuming
 99                                                               ; (dsp_buff_1 for now).
100
101                                    ; RESET THE VARIABLES WITH ZERO
102  000019 e010                       ldi r17, $00
103  00001a 9310 0102                  sts burst_count_setting_bcd + 2, r17
104  00001c 9310 0101                  sts burst_count_setting_bcd + 1, r17
105  00001e 9310 0100                  sts burst_count_setting_bcd + 0, r17
```

```
106
107  000020 9310 0105                    sts polling_for_keypad, r17
108  000022 9310 0106                    sts polling_for_button, r17
109
110  000024 9310 0103                    sts burst_count, r17
111
112  000026 9310 0104                    sts keyvalue, r17
113
114  000028 9478                         sei                 ;set global interrupt enable
115
116                                  ;*****************************************************************
117                                  ;**********************CODE BEGINS********************************
118                                  ;*****************************************************************
119
120                                  ;This runs after the peripherals are initalized
121                                  state_1:
122
123                                     ; Reset the polling for the keypad press, this is
124                                     important because state_2 jumps to this label
125  000029 e010                        ldi r17, $00
126  00002a 9310 0105                   sts polling_for_keypad, r17
127
128
129  00002c d04a                        rcall display_the_value
130                                  ;This Convert the registers to PACKED BCD
131                                  convert_to_Packed_BCD:
132  00002d 9100 0100                   lds r16, burst_count_setting_bcd
133  ; Retrieve the value store in the FIRST byte of burst_count_setting_bcd
134  and store it in r16
135  00002f 9110 0101                   lds r17, burst_count_setting_bcd + 1
136  ; Retrieve the value store in the SECOND byte of burst_count_setting_bcd
137  and store it in r17
138  000031 9120 0102                   lds r18, burst_count_setting_bcd + 2
139  ; Retrieve the value store in the THIRD byte of burst_count_setting_bcd
140   and store it in r18
```

```
141
142 000033 9512                       swap r17
143 ; Swap the nibble in r17
144 000034 2b01                       or r16, r17
145 ; Or r16 & r17, Combine the two contents of two registers into one register (r16)
146 000035 702f                       andi r18, $0F
147 ; AND r18 & $0F, clear the high nibble of r18
148 000036 2f12                       mov r17, r18
149 ; Move the content of r18 into r17
150 000037 e020                       ldi r18, $00                          ; Load r18 with zero, this will be
      useful
151  when we are trying to convert
152                                                                         ; Packed BCD into a 16-bit
153                                ;This converts the Packed BCD into the 16-bit binary
154                                convert_BCD_to_Binary:
155 000038 940e 00cd                  call BCD2bin16
156
157 00003a 2d3e                       mov r19, r14                          ; Moves the low byte of the 16-bit
158  binary value into r17
159 00003b 9330 0103                  sts burst_count, r19                  ; Store the value of r17 into
160 burst_count_bin
161
162 00003d 9100 0106                  lds r16, polling_for_button
163 00003f 3001                       cpi r16, 1
164 000040 f009                       breq state_2
165
166 000041 cfe7                       rjmp state_1
167
168                                state_2:
169 000042 e000                       ldi r16, 0                        ; Reset the flag that polls the push
      button
170 000043 9300 0106                  sts polling_for_button, r16
171
172                                   ; Reinitialize the Burst count
173 000045 e00a                       ldi r16, 10                       ; Load ten into r16, This is to create
```

```
174   the 1 ms delay
175  000046 9130 0103                 lds r19, burst_count               ; This loads r19 with the orginal
176   binary value
177
178                                  check_zero:
179  000048 3030                      cpi r19, $00                       ; Branch to generate a pulse if
180   burst count = 0
181  000049 f0b1                      breq generate_a_pulse
182
183                                  ;This generate a pulse that is supposed to be 1 ms wide
184                                  pulse_generator:
185  00004a 9a17                      sbi PORTA, 7                       ; set bit for pulse
186  00004b d069                      rcall var_delay
187  00004c e00a                      ldi r16, 10                        ; pulse width delay
188  00004d 9817                      cbi PORTA, 7                       ; clear bit for pulse
189  00004e d066                      rcall var_delay
190  00004f e00a                      ldi r16, 10                        ; time between pulses delay
191  000050 953a                      dec r19                            ; decrement the binary value
192  000051 f7c1                      brne pulse_generator
193
194                                  ;This part is reached if the burst count is equal to zero
195                                  check_flag_2:
196
197  000052 9140 0106                 lds r20, polling_for_button
198  000054 9150 0105                 lds r21, polling_for_keypad
199
200  000056 3041                      cpi r20, 1                         ; Check if the pushbutton is pressed
201  000057 f351                      breq state_2
202
203  000058 3051                      cpi r21, 1                         ; Check if the pushbutton is pressed
204  000059 f009                      breq service_keypad_input
205  00005a cff7                      rjmp check_flag_2
206
207                                  service_keypad_input:
208  00005b 9120 0104                 lds r18, keyvalue
```

```
209  00005d 302a                        cpi r18, $0A                        ; checks if the key value is equal
210   to CLEAR
211  00005e f251                        breq state_1                        ; goes to the beginning if the key
212  value is equal to CLEAR
213  00005f cff2                        rjmp check_flag_2                   ; goes back to generate another set
214  of pulses
215
216                                   ; This is branched if burst count is equal to zero
217                                   generate_a_pulse:
218  000060 9a17                        sbi PORTA, 7                        ; set bit for pulse
219  000061 d053                        rcall var_delay
220  000062 e00a                        ldi r16, 10                         ; pulse width delay
221  000063 9817                        cbi PORTA, 7                        ; clear bit for pulse
222  000064 d050                        rcall var_delay
223  000065 e00a                        ldi r16, 10
224                                                                         ; time between pulses delay
225  000066 9140 0105                   lds r20, polling_for_keypad
226  000068 ff40                        sbrs r20, 0                         ; Skips the rjmp instruction
227  if the value in polling_for_keypad = 1
228  000069 cff6                        rjmp generate_a_pulse
229
230  00006a 9120 0104                   lds r18, keyvalue
231  00006c 302a                        cpi r18, $0A                        ; Check if key value is
232  equal to clear
233  00006d f001                        breq prompt1
234
235                                     prompt1:
236  00006e cfba                        rjmp state_1
237
238                                   ;---------------------------- SUBROUTINES ------------------------ ---
239
240                                   ;************************
241                                   ;NAME:      clr_dsp_buffs
242                                   ;FUNCTION:  Initializes dsp_buffers 1, 2, and 3 with blanks (0x20)
243                                   ;ASSUMES:   Three CONTIGUOUS 16-byte dram based buffers named
```

```
244                                 ;           dsp_buff_1, dsp_buff_2, dsp_buff_3.
245                                 ;RETURNS:   nothing.
246                                 ;MODIFIES:  r25,r26, Z-ptr
247                                 ;CALLS:     none
248                                 ;CALLED BY: main application and diagnostics
249                                 ;*************************************************************** **
250                                 clr_dsp_buffs:
251  00006f e390                         ldi R25, 48           ; load total length of both buffer.
252  000070 e2a0                         ldi R26, ' '              ; load blank/space into R26.
253  000071 e0f1                         ldi ZH, high (dsp_buff_1) ; Load ZH and ZL as a pointer to 1st
254  000072 e0e7                         ldi ZL, low (dsp_buff_1)  ; byte of buffer for line 1.
255
256                                     ;set DDRAM address to 1st position of first line.
257                                 store_bytes:
258  000073 93a1                         st  Z+, R26       ; store ' ' into 1st/next buffer byte and
259                                                        ; auto inc ptr to next location.
260  000074 959a                         dec  R25          ;
261  000075 f7e9                         brne store_bytes  ; cont until r25=0, all bytes written.
262  000076 9508                         ret
263
264
265                                 ;*******************************************
266                                 ;SUBROUTINE FOR DISPLAYING THE INPUT TO LCD
267                                 ;*******************************************
268                                 display_the_value:
269  000077 e0d1                        ldi YH, high (dsp_buff_1) ; Load YH and YL as a pointer to 1st
270  000078 e0c7                         ldi YL, low (dsp_buff_1)  ; byte of dsp_buff_1 (Note - assuming
271                                                               ; (dsp_buff_1 for now).
272
273  000079 e60e                        ldi r16, 'n'
274  00007a 9309                        st Y+, r16
275  00007b e200                        ldi r16, ' '
276  00007c 9309                        st Y+, r16
277  00007d e30d                        ldi r16, '='
278  00007e 9309                        st Y+, r16
```

```
279  00007f e200                        ldi r16, ' '
280  000080 9309                        st Y+, r16
281
282  000081 e310                        ldi r17, $30                          ; Load $30 into r16
283                                     ; store the ascii representation of the digit in the buffer
284  000082 9100 0102                   lds r16, (burst_count_setting_bcd + 2)          ; Store the
285  leftmost keyvalue into r16
286
287  000084 2b01                        or r16, r17                           ; Adds $30 to the keyvalue,
288  which turn the keyvalue into ASCII
289  000085 9309                        st Y+, r16                                     ; Put the value
290   into the display buffer
291
292  000086 9100 0101                   lds r16, (burst_count_setting_bcd + 1)          ;
293  000088 2b01                        or r16, r17                           ; Adds $30 to the keyvalue,
294   which turn the keyvalue into ASCII
295  000089 9309                        st Y+, r16
296
297  00008a 9100 0100                   lds r16, (burst_count_setting_bcd + 0)          ; Store the
298   rightmost keyvalue into r16
299  00008c 2b01                        or r16, r17                           ; Adds $30 to the keyvalue,
300   which turn the keyvalue into ASCII
301  00008d 9309                        st Y+, r16                                     ; Put the value
302   into the display buffer
303
304  00008e 940e 0159                   call update_lcd_dog                            ; update the display
305  000090 9508                        ret
306
307                                  ;*********************************************
308                                  ;SUBROUTINE FOR STORING THE VALUE INTO THE Variable
309                                  ;*********************************************
310                                  store_value:
311                                     ;r18 is the value read by the input
312
313  000091 9100 0101                   lds r16, burst_count_setting_bcd + 1    ; Load r16
```

```
with the middle digit
000093 9300 0102                    sts burst_count_setting_bcd + 2, r16    ; Put the
middle digit into the leftmost digit

000095 9100 0100                    lds r16, burst_count_setting_bcd + 0    ; Load r16
with the Rightmost digit
000097 9300 0101                    sts burst_count_setting_bcd + 1, r16    ; Put the
rightmost digit into the middle digit

000099 9320 0100                    sts burst_count_setting_bcd + 0, r18    ; Store the
new number into the rightmost digit
00009b 9508                         ret


                                ;*********************************
                                ;SUBROUTINE FOR RETRIEVING INPUT(PART 1)
                                ;*********************************
                                get_key_value:
00009c 9b36                         sbis PINC, 6            ; Check if any value on the
keypad is press
00009d cffe                         rjmp get_key_value      ; Loop back if no keypad is
pressed
00009e b129                         in r18, PIND            ; Store the Input into r18
00009f 7f20                         andi r18, $F0           ; Clear the low nibble of r18
0000a0 9522                         swap r18                ; Swap the nibble
0000a1 940e 00ae                    call keycode2keyvalue   ; Convert the input into
 HEXVALUES (NOT ASCII)
0000a3 9847                         cbi PORTC, 7            ; Clear the FLip Flop
that is connected to the encoder
0000a4 9a47                         sbi PORTC, 7            ;
0000a5 9508                         ret


                                ;*********************************
```

```
349                                 ;SUBROUTINE FOR RETRIEVING INPUT(PART 2)
350                                 ;*********************************
351                                 get_key_value_3:
352  0000a6 b129                       in r18, PIND            ; Store the Input
353   into r18
354  0000a7 7f20                       andi r18, $F0           ; Clear the low nibble
355  of r18
356  0000a8 9522                       swap r18                ; Swap the nibble
357  0000a9 940e 00ae                  call keycode2keyvalue   ; Convert the input into
358  HEXVALUES (NOT ASCII)
359  0000ab 9847                       cbi PORTC, 7            ; Clear the FLip Flop that
360  is connected to the encoder
361  0000ac 9a47                       sbi PORTC, 7            ;
362  0000ad 9508                       ret
363
364                                 ;*********************************
365                                 ;SUBROUTINE FOR LOOKUP TABLE
366                                 ;*********************************
367                                 keycode2keyvalue:
368                                 lookup:
369  0000ae e0f1                       ldi ZH, high (keytable * 2)     ;set Z to point to
370  start of table
371  0000af efec                       ldi ZL, low (keytable * 2)
372  0000b0 e000                       ldi r16, $00                    ;add offset to Z
373  pointer
374  0000b1 0fe2                       add ZL, r18                     ;originally r18
375  0000b2 0ff0                       add ZH, r16
376  0000b3 9124                       lpm r18, Z
377  0000b4 9508                       ret
378
379                                 ;*************************
380                                 ;SUBROUTINE FOR DELAY
381                                 ;*************************
382                                 var_delay: ;delay for ATmega324 @ 1MHz = r16 * 0.1 ms
383                                 outer_loop:; r16 should equal to 10
```

```
384  0000b5 e210                        ldi r17, 32
385                                   inner_loop:
386  0000b6 951a                        dec r17
387  0000b7 f7f1                        brne inner_loop
388  0000b8 950a                        dec r16
389  0000b9 f7d9                        brne outer_loop
390  0000ba 9508                        ret
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(10(10a+b)+c)+d)+e.
403                                   ;* The subroutine "mul10a"/"mul10b" does the multiply-and-add opera tion
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" hi gh nibble ↵

424 0000bb 9532                        swap    adder
425                                 mul10b:     ;***** multiplies "mp10H:mp10L" with 10 and adds "adder" lo w nibble
426 0000bc 2cce                        mov copyL,mp10L ;make copy
427 0000bd 2cdf                        mov copyH,mp10H
428 0000be 0cee                        lsl mp10L       ;multiply original by 2
429 0000bf 1cff                        rol mp10H
430 0000c0 0ccc                        lsl copyL       ;multiply copy by 2
431 0000c1 1cdd                        rol copyH
432 0000c2 0ccc                        lsl copyL       ;multiply copy by 2 (4)
433 0000c3 1cdd                        rol copyH
434 0000c4 0ccc                        lsl copyL       ;multiply copy by 2 (8)
435 0000c5 1cdd                        rol copyH
436 0000c6 0cec                        add mp10L,copyL ;add copy to original
437 0000c7 1cfd                        adc mp10H,copyH
438 0000c8 703f                        andi    adder,0x0f  ;mask away upper nibble of adder
439 0000c9 0ee3                        add mp10L,adder ;add lower nibble of adder
440 0000ca f408                        brcc    m10_1       ;if carry not cleared
441 0000cb 94f3                        inc mp10H       ;   inc high byte
442 0000cc 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  0000cd 702f                       andi    fBCD2,0x0f  ;mask away upper nibble of fBCD2
456  0000ce 24ff                       clr mp10H
457  0000cf 2ee2                       mov mp10L,fBCD2 ;mp10H:mp10L = a
458  0000d0 2f31                       mov adder,fBCD1
459  0000d1 dfe9                       rcall   mul10a      ;mp10H:mp10L = 10a+b
460  0000d2 2f31                       mov adder,fBCD1
461  0000d3 dfe8                       rcall   mul10b      ;mp10H:mp10L = 10(10a+b)+c
462  0000d4 2f30                       mov adder,fBCD0
463  0000d5 dfe5                       rcall   mul10a      ;mp10H:mp10L = 10(10(10a+b)+c)+d
464  0000d6 2f30                       mov adder,fBCD0
465  0000d7 dfe4                       rcall   mul10b      ;mp10H:mp10L = 10(10(10(10a+b)+c)+d)+e
466  0000d8 9508                       ret
467
468
469                                 ;*************************************************************** *********
470                                 ;*
471                                 ;* "keypress_ISR" - Check Interrupts at INT0
472                                 ;*
473                                 ;* Description: Get the keyvalue if the key is pressed,
474                                  the keyvalue is stored if the key is a number
475                                 ;*
476                                 ;* Author:                 Seyi Olajuyi & Bassel El Amine
477                                 ;* Version:
478                                 ;* Last updated:           11/21/19
479                                 ;* Target:                 ATmega324A
480                                 ;* Number of words:
481                                 ;* Number of cycles:       N/A
482                                 ;* Low registers modified:  none
483                                 ;* High registers modified: none
484                                 ;*
485                                 ;* Parameters:
486                                 ;* Notes:
487                                 ;*
```

```
488                                 ;*********************************************************************** *********
489
490                                     ;INT0 interrupt service routine
491                                 keypress_ISR:
492  0000d9 932f                       push r18
493  0000da 930f                        push r16            ;save r16
494  0000db b70f                       in r16, SREG        ;save SREG
495  0000dc 930f                        push r16
496
497  0000dd e001                       ldi r16 ,1                      ; Set polling_for_keypad
498  0000de 9300 0105                  sts polling_for_keypad, r16     ; Use to find out if the
499   keypad was pressed
500
501  0000e0 dfbb                       rcall get_key_value
502  0000e1 9320 0104                  sts keyvalue, r18
503  0000e3 302a                       cpi r18, $0A                            ; if key value is not
504  a number, end the subroutine.
505  0000e4 f028                       brlo skip_line_1
506
507                                 restore_values_1:
508  0000e5 910f                        pop r16             ;restore SREG
509  0000e6 bf0f                        out SREG,r16
510  0000e7 910f                       pop r16             ;restore r16
511  0000e8 912f                       pop r18             ;restore r18
512
513  0000e9 9518                        reti                ;return from interrupt
514
515                                 skip_line_1:
516  0000ea dfa6                       rcall store_value
517  0000eb cfed                       rjmp keypress_ISR
518
519
520
521                                 ;*********************************************************************** *********
522                                 ;*
```

```
                                ;* "pb_press_ISR" - Check Interrupts at INT1
                                ;*
                                ;* Description: Checks if the push button is pressed
                                ;*
                                ;* Author:                 Ken Short
                                ;* Version:
                                ;* Last updated:           11/21/19
                                ;* Target:                 ATmega324A
                                ;* Number of words:
                                ;* Number of cycles:       16
                                ;* Low registers modified:  none
                                ;* High registers modified: none
                                ;*
                                ;* Parameters:  Uses PORTB register to hold the count and drive LED s
                                ;* connected to that port.
                                ;*
                                ;* Notes:
                                ;*
                                ;********************************************************************* *********

                                    ;INT1 interrupt service routine
                                pb_press_ISR:
                                wait_for_bounce_1:
0000ec 930f                        push r16            ;save r16
0000ed b70f                        in r16, SREG        ;save SREG
0000ee 930f                         push r16

0000ef 9904                        sbic PINA, 4
0000f0 cffb                        rjmp wait_for_bounce_1
0000f1 e604                        ldi r16, 100
0000f2 dfc2                        rcall var_delay
0000f3 9904                        sbic PINA, 4
0000f4 cff7                        rjmp wait_for_bounce_1

0000f5 e002                        ldi r16, (1 <<INTF1)
```

```
0000f6 bb0c                         out EIFR, r16

0000f7 e001                         ldi r16 ,1                      ; Set polling_for_button
0000f8 9300 0106                    sts polling_for_button, r16     ; Use to find out if
the button was pressed

                                restore_value_2:
0000fa 910f                          pop r16             ;restore SREG
0000fb bf0f                          out SREG,r16
0000fc 910f                         pop r16             ;restore r16

0000fd 9518                          reti                ;return from interrupt



0000fe 0201
0000ff 0f03
000100 0504
000101 0e06
000102 0807
000103 0d09                     keytable: .db $01, $02, $03, $0F, $04, $05, $06, $0E, $07, $08, $09 , $0D
000104 000a
000105 0c0b                             .db $0A, $00, $0B, $0C


                                .list


RESOURCE USE INFORMATION
------------------------

Notice:
The register and instruction counts are symbol table hit counts,
and hence implicitly used resources are not counted, eg, the
```

```
593  'lpm' instruction without operands implicitly uses r0 and z,
594  none of which are counted.
595
596  x,y,z are separate entities in the symbol table and are
597  counted separately from r26..r31 here.
598
599  .dseg memory usage only counts static data declared with .byte
600
601  "ATmega324A" register use summary:
602  x  :   0 y  :   7 z  :   5 r0 :   0 r1 :   0 r2 :   0 r3 :   0 r4 :   0
603  r5 :   0 r6 :   0 r7 :   0 r8 :   0 r9 :   0 r10:   0 r11:   0 r12:   5
604  r13:   5 r14:   6 r15:   5 r16: 103 r17:  23 r18:  23 r19:  12 r20:  12
605  r21:   2 r22:   2 r23:   2 r24:   4 r25:   2 r26:   2 r27:   0 r28:   2
606  r29:   2 r30:   6 r31:   6
607  Registers used: 21 out of 35 (60.0%)
608
609  "ATmega324A" instruction use summary:
610  .lds  :   0 .sts  :   0 adc   :   1 add   :   4 adiw  :   0 and   :   0
611  andi  :   5 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
612  brcc  :   1 brcs  :   0 break :   0 breq  :   6 brge  :   0 brhc  :   0
613  brhs  :   0 brid  :   0 brie  :   0 brlo  :   1 brlt  :   0 brmi  :   0
614  brne  :  10 brpl  :   0 brsh  :   0 brtc  :   0 brts  :   0 brvc  :   0
615  brvs  :   0 bset  :   0 bst   :   0 call  :   4 cbi   :   7 cbr   :   0
616  clc   :   0 clh   :   0 cli   :   0 cln   :   0 clr   :   1 cls   :   0
617  clt   :   0 clv   :   0 clz   :   0 com   :   0 cp    :   0 cpc   :   0
618  cpi   :   7 cpse  :   0 dec   :  10 eor   :   0 fmul  :   0 fmuls :   0
619  fmulsu:   0 icall :   0 ijmp  :   0 in    :  13 inc   :   1 jmp   :   0
620  ld    :   3 ldd   :   0 ldi   :  59 lds   :  15 lpm   :   2 lsl   :   4
621  lsr   :   0 mov   :   9 movw  :   0 mul   :   0 muls  :   0 mulsu :   0
622  neg   :   0 nop   :   2 or    :   4 ori   :   0 out   :  11 pop   :  11
623  push  :  11 rcall :  50 ret   :  17 reti  :   2 rjmp  :  14 rol   :   4
624  ror   :   0 sbc   :   0 sbci  :   0 sbi   :  12 sbic  :   2 sbis  :   1
625  sbiw  :   0 sbr   :   0 sbrc  :   0 sbrs  :   3 sec   :   0 seh   :   0
626  sei   :   1 sen   :   0 ser   :   0 ses   :   0 set   :   0 sev   :   0
627  sez   :   0 sleep :   0 spm   :   0 st    :   8 std   :   0 sts   :  17
```

```
628 sub   :   0 subi  :   0 swap  :   4 tst   :   0 wdr   :   0
629 Instructions used: 38 out of 113 (33.6%)
630
631 "ATmega324A" memory use summary [bytes]:
632 Segment   Begin    End      Code   Data   Used    Size   Use%
633 ----------------------------------------------------------------
634 [.cseg] 0x000000 0x0002fc    744     16    760   32768   2.3%
635 [.dseg] 0x000100 0x000137      0     55     55    2048   2.7%
636 [.eseg] 0x000000 0x000000      0      0      0    1024   0.0%
637
638 Assembly complete, 0 errors, 2 warnings
639
```