```
 1
 2  AVRASM ver. 2.2.7  C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0
      \ppg_IV_fsm_extra\
 3  ppg_IV_fsm_extra\main.asm Thu Dec 05 18:05:47 2019
 4
 5  C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_IV_fsm_extra
      \ppg_IV_fsm_extra\main.asm(24):
 6  Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP
      \
 7  1.3.300\avrasm\inc\m324adef.inc'
 8  C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_IV_fsm_extra
      \ppg_IV_fsm_extra\main.asm(780):
 9   warning: Register r14 already defined by the .DEF directive
10  C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_IV_fsm_extra
      \ppg_IV_fsm_extra\main.asm(781):
11   warning: Register r15 already defined by the .DEF directive
12  C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_IV_fsm_extra
      \ppg_IV_fsm_extra\main.asm(949):
13   Including file 'C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0
       \ppg_IV_fsm_extra\
14   ppg_IV_fsm_extra\lcd_dog_asm_driver_m324a.inc'
15  C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_IV_fsm_extra
      \ppg_IV_fsm_extra\main.asm(24):
16   Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel
       \ATmega_DFP\1.3.300\
17   avrasm\inc\m324adef.inc'
18  C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_IV_fsm_extra
      \ppg_IV_fsm_extra\main.asm(949):
19   Including file 'C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0
       \ppg_IV_fsm_extra\
20   ppg_IV_fsm_extra\lcd_dog_asm_driver_m324a.inc'
21
22
23                          ;*****************************************
                    ******************************
24                          ;*
25                          ;* Title: ppg_IV_fsm
26                          ;* Author:          Seyi Olajuyi & Bassel El
                    Amine
27                          ;* Version:         2.0
28                          ;* Last updated:    12/05/19
29                          ;* Target:          ATmega16
30                          ;*
31                          ;* DESCRIPTION
32                          ;* This is a simplified version of the table
                    driven FSM. It handles only 255
33                          ;* or less input symbols.
34                          ;*
35                          ;* A sample table is included for a simple
```

```
                        FSM. This table can be modified to
 36                                 ;* handle any FSM by equating the input
                        symbols to byte values starting at
 37                                 ;* $00 and entering the appropriate next
                        state and task subroutine names.
 38                                 ;*
 39                                 ;*
 40                                 ;* VERSION HISTORY
 41                                 ;* 1.0 Original version
 42                                 ;* 2.0 Subroutines moved to end of file
 43                                 ;*****************************************
                        ******************************
 44                                 .list
 45
 46                                 .dseg   ;The variable below are in SRAM
 47 000100                          burst_count_setting_bcd:        .byte 3;
     setting unpacked BCD ;THIS HAS
 48  THREE BTYE allocated to the variable name
 49 000103                          burst_count:                    .byte 1;
     pulses left to generated in
 50  burst
 51 000104                          pulse_width_bcd:                .byte 3
 52 000107                          pulse_width:                    .byte 1
 53 000108                          pulse_delay_bcd:                .byte 3
 54 00010b                          pulse_delay:                    .byte 1
 55 00010c                          line:                           .byte 3
 56 00010f                          keyvalue:                       .byte 1;
     stores the keyvalue into
 57  a variable
 58 000110                          make_pulse:                         .byte 1;
 59 000111                          is_burst_zero:                  .byte 1; Used
     to check if burst
 60  count is equal to zero. 1 means burst count is equal to zero
 61 000112                          input:                          .byte 1; input
 62
 63
 64                                 ;burst_count_setting_bcd is right most digit
                        and
 65                                 ; (burst_count_setting_bcd + 2) is the left
                        most digit
 66
 67                                 .cseg
 68                                 reset:
 69                                 .org RESET              ;reset interrupt
                        vector
 70 000000 c004                         rjmp start          ;program starts here
     at reset
 71                                 .org INT0addr           ;INT0 interrupt
                        vector
```

```
 72  000002 c210                         rjmp keypress_ISR
 73                                    .org INT1addr
 74  000004 c23e                         rjmp pb_press_ISR
 75
 76                                    start:
 77  000005 ef0f                         ldi r16, LOW(RAMEND)    ;initialize SP to
       point to
 78   top of stack
 79  000006 bf0d                         out SPL, r16
 80  000007 e008                         ldi r16, HIGH(RAMEND)
 81  000008 bf0e                         out SPH, r16
 82
 83  000009 e00f                         ldi r16, (1 << ISC00) | (1 << ISC01) | (1
       << ISC10) | (1 << ISC11)
 84  00000a 9300 0069                    sts EICRA, r16
 85  00000c e003                        ldi r16, $03        ; Enable interrupt
      request at INTO & INT1
 86  00000d bb0d                         out EIMSK, r16
 87
 88  00000e ef0f                         ldi r16, $ff      ; load r16 with all 1s.
 89  00000f b904                         out DDRB, r16     ; set portB = output
 90
 91  000010 e003                        ldi r16, $03        ; Set pin 0 & pin 1 to
       output, everyother
 92   pin is an input
 93  000011 b90a                        out DDRD, r16
 94
 95  000012 9a0e                        sbi DDRA, 6         ;Set Pin 6 on PORTA
      (Buzzer)
 96
 97  000013 9a0f                        sbi DDRA, 7         ; Set pin 7 on PORTA
      to output (OUTPUT)
 98
 99  000014 9a2c                         sbi portB, 4      ; set /SS of DOG LCD =
      1 (Deselected)
100
101  000015 d27e                         rcall init_lcd_dog        ; init display,
       using SPI serial
102   interface
103  000016 d09d                        rcall clr_dsp_buffs       ; clear all
      three buffer lines
104  000017 d29a                        rcall update_lcd_dog      ; update the
      display
105
106  000018 e0d1                        ldi YH, high (dsp_buff_1) ; Load YH and YL
       as a pointer to 1st
107  000019 e1c3                         ldi YL, low (dsp_buff_1)  ; byte of
      dsp_buff_1 (Note - assuming
108                                                                ; (dsp_buff_1
```

```
                        for now).
109
110                                     ;put FSM in initial state
111  00001a e487                         ldi pstatel, LOW(begin)
112  00001b e090                         ldi pstateh, HIGH(begin)
113
114  00001c 9478                        sei                 ;set global interrupt
       enable
115
116                                   variable_reset:
117                                     ; RESET THE VARIABLES WITH ZERO
118  00001d e010                        ldi r17, $00
119  00001e 9310 0102                   sts burst_count_setting_bcd + 2, r17
120  000020 9310 0101                   sts burst_count_setting_bcd + 1, r17
121  000022 9310 0100                   sts burst_count_setting_bcd + 0, r17
122
123  000024 9310 0110                   sts make_pulse, r17
124  000026 9310 0111                   sts is_burst_zero, r17
125
126  000028 9310 0103                   sts burst_count, r17
127
128  00002a 9310 010f                   sts keyvalue, r17
129
130  00002c e00a                        ldi r16, 10
131  00002d 9300 0107                   sts pulse_width, r16
132  00002f 9300 010b                   sts pulse_delay, r16
133
134  000031 e203                        ldi r16, '#'
135  000032 e210                        ldi r17, ' '
136  000033 9300 010c                   sts line, r16
137  000035 9310 010d                   sts line + 1, r17
138  000037 9310 010e                   sts line + 2, r17
139
140
141                                   test:
142  000039 9100 0110                    lds r16, make_pulse
143  00003b ff00                        sbrs r16, 0             ; Skip the rjmp
       instruction if the
144  make_pulse flag is set
145  00003c cffc                         rjmp test
146
147  00003d 9100 0111                   lds r16, is_burst_zero
148  00003f 3001                        cpi r16, 1
149  000040 f419                        brne gen_1_pulse
150
151  000041 940e 01d7                   call generate_a_pulse
152  000043 cff5                        rjmp test
153
154                                   gen_1_pulse:
```

```
000044 940e 01ca                    call pulse_generator
000046 cff2                         rjmp test



                        ;***************************************************************************
                        ;*
                        ;* "fsm" - Simplified Table Driven Finite State Machine
                        ;*
                        ;* Description:
                        ;* This table driven FSM can handle 255 or fewer input symbols.
                        ;*
                        ;* Author:              Ken Short
                        ;* Version:             2.0
                        ;* Last updated:        11/09/15
                        ;* Target:              ATmega16
                        ;* Number of words:
                        ;* Number of cycles:
                        ;* Low regs modified:   r16, r18, r20, r21, r31, and r31
                        ;* High registers used:
                        ;*
                        ;* Parameters:          present state in r25:r24 prior to call
                        ;*                      input symbol in r16 prior to call
                        ;*
                        ;* Notes:
                        ;*
                        ;***************************************************************************

                        .def pstatel = r24 ;low byte of present state address
                        .def pstateh = r25 ;high byte of present state address

                        ;input symbols for example finite state machine
                        .equ number = $00   ;input symbols equated to numerical values ;
                        .equ enter = $01
                        .equ clear = $02
                        .equ pushb = $03
                        .equ up_arrow = $04
                        .equ down_arrow = $05
```

```
                                                ;additional symbols would go here
                        .equ eol = $FF  ;end of list (subtable) do not change

                        ;state table for example finite state machine
                        ;each row consists of input symbol, next state address, task
                        ;subroutine address

                        state_table:

000047 0000
000048 0059
000049 00bc             begin:  .dw number,     line1,          update_all
00004a 0004
00004b 0059
00004c 00bc                     .dw up_arrow,   line1,          update_all
00004d 0005
00004e 0059
00004f 00bc                     .dw down_arrow, line1,          update_all
000050 0001
000051 0059
000052 00bc                     .dw enter,      line1,          update_all
000053 0002
000054 0059
000055 00bc                     .dw clear,      line1,          update_all
000056 00ff
000057 0059
000058 00bc                     .dw eol,        line1,          update_all

000059 0000
00005a 0059
00005b 00c3             line1:  .dw number,     line1,          update_line_1
00005c 0004
00005d 007d
00005e 015b                     .dw up_arrow,   line3,          switch_up_and_display
00005f 0005
000060 006b
000061 0156                     .dw down_arrow, line2,
```

```
       switch_down_and_display
231  000062 0001
232  000063 008f
233  000064 0193                          .dw enter,          idle,              ↪
       convert
234  000065 0002
235  000066 0059
236  000067 018d                          .dw clear,          line1,       buzz
237  000068 00ff
238  000069 0059
239  00006a 018d                          .dw eol,            line1,       buzz
240
241  00006b 0000
242  00006c 006b
243  00006d 00c8                  line2:  .dw number,         line2,             ↪
       update_line_2
244  00006e 0004
245  00006f 0059
246  000070 015b                          .dw up_arrow,       line1,             ↪
       switch_up_and_display
247  000071 0005
248  000072 007d
249  000073 0156                          .dw down_arrow,     line3,             ↪
       switch_down_and_display
250  000074 0002
251  000075 006b
252  000076 018d                          .dw clear,          line2,       buzz
253  000077 0001
254  000078 008f
255  000079 0193                          .dw enter,          idle,              ↪
       convert
256  00007a 00ff
257  00007b 006b
258  00007c 018d                          .dw eol,            line2,       buzz
259
260  00007d 0000
261  00007e 007d
262  00007f 00cd                  line3:  .dw number,         line3,             ↪
       update_line_3
263  000080 0004
264  000081 006b
265  000082 015b                          .dw up_arrow,       line2,             ↪
       switch_up_and_display
266  000083 0005
267  000084 0059
268  000085 0156                          .dw down_arrow,     line1,             ↪
       switch_down_and_display
269  000086 0002
270  000087 007d
```

```
000088 018d                             .dw clear,          line3,      buzz
000089 0001
00008a 008f
00008b 0193                             .dw enter,          idle,           ↵
  convert
00008c 00ff
00008d 007d
00008e 018d                             .dw eol,            line3,      buzz
00008f 0003
000090 0095
000091 01dc                     idle:   .dw pushb,          burst,          ↵
  update_flags
000092 00ff
000093 008f
000094 018d                             .dw eol,            idle,       buzz
000095 0003
000096 0095
000097 01dc                     burst:  .dw pushb,          burst,          ↵
  update_flags
000098 0002
000099 0059
00009a 01eb                             .dw clear,          line1,          ↵
  clear_flags
00009b 00ff
00009c 0095
00009d 018d                             .dw eol,            burst,      buzz ↵


                                fsm:
                                ;load Z with a byte pointer to the subtable ↵
                    corresponding to the
                                ;present state
00009e 9100 0112                   lds r16, input
0000a0 2fe8                         mov ZL, pstatel ;load Z pointer with    ↵
  pstate address * 2
0000a1 0fee                         add ZL, ZL ;since Z will be used as a   ↵
  byte pointer with the lpm instr.
0000a2 2ff9                         mov ZH, pstateh
0000a3 1fff                         adc ZH, ZH

                                ;search subtable rows for input symbol match
                                search:
0000a4 9124                         lpm r18, Z ;get symbol from state table
0000a5 1720                         cp r18, r16 ;compare table entry with   ↵
  input symbol
0000a6 f021                         breq match
```

```
311
312                                 ;check input symbol against eol
313                                 check_eol:
314 0000a7 3f2f                         cpi r18, eol ;compare low byte of table entry with eol
315 0000a8 f011                         breq match
316
317                                 nomatch:
318 0000a9 9636                         adiw ZL, $06 ;adjust Z to point to next row of state table
319 0000aa cff9                         rjmp search ;continue searching
320
321                                 ;a match on input value to row input value has been found
322                                 ;the next word in this row is the next state address
323                                 ;the word following that is the task subroutine's address
324                                 match:
325                                    ;make preseent state equal to next state value in row
326                                    ;this accomplishes the stat transition
327 0000ab 9632                        adiw ZL, $02 ;point to low byte of state address
328 0000ac 9185                        lpm pstatel, Z+; ;copy next state addr. from table to preseent stat
329 0000ad 9195                        lpm pstateh, Z+
330
331                                    ;execute the subroutine that accomplihes the task associated
332                                    ;with the transition
333 0000ae 9145                        lpm r20, Z+ ;get subroutine address from state table
334 0000af 9154                        lpm r21, Z ;and put it in Z pointer
335 0000b0 2fe4                        mov ZL, r20
336 0000b1 2ff5                        mov ZH, r21
337 0000b2 9509                        icall ;Z pointer is now used as a word pointer
338 0000b3 9508                        ret
339
340
341                                 ;************************
342                                 ;NAME:      clr_dsp_buffs
343                                 ;FUNCTION:  Initializes dsp_buffers 1, 2, and 3 with blanks (0x20)
344                                 ;ASSUMES:   Three CONTIGUOUS 16-byte dram based buffers named
345                                 ;           dsp_buff_1, dsp_buff_2, dsp_buff_3.
```

```
346                                 ;RETURNS:   nothing.
347                                 ;MODIFIES:  r25,r26, Z-ptr
348                                 ;CALLS:     none
349                                 ;CALLED BY: main application and diagnostics
350                                 ;*****************************************
                    ************************
351                                 clr_dsp_buffs:
352 0000b4 e390                         ldi R25, 48           ; load total
      length of both buffer.
353 0000b5 e2a0                         ldi R26, ' '              ; load blank/
      space into R26.
354 0000b6 e0f1                         ldi ZH, high (dsp_buff_1) ; Load ZH and
      ZL as a pointer to 1st
355 0000b7 e1e3                         ldi ZL, low (dsp_buff_1)  ; byte of
      buffer for line 1.
356
357                                     ;set DDRAM address to 1st position of
                    first line.
358                                 store_bytes:
359 0000b8 93a1                         st  Z+, R26       ; store ' ' into 1st/
      next buffer byte and
360                                                       ; auto inc ptr to next
                     location.
361 0000b9 959a                         dec  R25          ;
362 0000ba f7e9                         brne store_bytes  ; cont until r25=0,
      all bytes written.
363 0000bb 9508                         ret
364
365
366                                 ;*****************************************
                    **
367                                 ;SUBROUTINE FOR UPDATE ALL
368                                 ;*****************************************
                    **
369                                 update_all:
370 0000bc 940e 00d2                    call display_line_1
371 0000be 940e 00f1                    call display_line_2
372 0000c0 940e 0110                    call display_line_3
373 0000c2 9508                         ret
374
375
376                                 ;*****************************************
                    **
377                                 ;SUBROUTINE FOR UPDATING (LINE 1)
378                                 ;*****************************************
                    **
379                                 update_line_1:
380 0000c3 940e 012f                    call store_value_line_1
381 0000c5 940e 00d2                    call display_line_1
```

```
0000c7 9508                         ret


                                    ;***************************************** ↵
                    **
                                    ;SUBROUTINE FOR UPDATING (LINE 2)
                                    ;***************************************** ↵
                    **
                                    update_line_2:
0000c8 940e 013c                    call store_value_line_2
0000ca 940e 00f1                    call display_line_2
0000cc 9508                         ret


                                    ;***************************************** ↵
                    **
                                    ;SUBROUTINE FOR UPDATING (LINE 3)
                                    ;***************************************** ↵
                    **
                                    update_line_3:
0000cd 940e 0149                    call store_value_line_3
0000cf 940e 0110                    call display_line_3
0000d1 9508                         ret


                                    ;***************************************** ↵
                    **
                                    ;SUBROUTINE FOR DISPLAYING THE INPUT TO LCD   ↵
                    (LINE 1)
                                    ;***************************************** ↵
                    **
                                    display_line_1:
0000d2 e0d1                         ldi YH, high (dsp_buff_1) ; Load YH and YL ↵
     as a pointer to 1st
0000d3 e1c3                          ldi YL, low (dsp_buff_1)  ; byte of       ↵
    dsp_buff_1 (Note - assuming
                                                                  ; (dsp_buff_1   ↵
                    for now).

0000d4 e60e                         ldi r16, 'n'
0000d5 9309                         st Y+, r16
0000d6 e200                         ldi r16, ' '
0000d7 9309                         st Y+, r16
0000d8 e30d                         ldi r16, '='
0000d9 9309                         st Y+, r16
0000da e200                         ldi r16, ' '
0000db 9309                         st Y+, r16

0000dc e310                         ldi r17, $30                            ;  ↵
```

```
      Load $30 into r16
421                                   ; store the ascii representation of the
                      digit in the buffer
422 0000dd 9100 0102                  lds r16, (burst_count_setting_bcd + 2)

423 ; Store the leftmost keyvalue into r16
424
425 0000df 2b01                       or r16, r17
426 ; Adds $30 to the keyvalue, which turn the keyvalue into ASCII
427 0000e0 9309                       st Y+, r16

428 ; Put the value into the display buffer
429
430 0000e1 9100 0101                  lds r16, (burst_count_setting_bcd + 1)
              ;
431 0000e3 2b01                       or r16, r17
432 ; Adds $30 to the keyvalue, which turn the keyvalue into ASCII
433 0000e4 9309                       st Y+, r16
434
435 0000e5 9100 0100                  lds r16, (burst_count_setting_bcd + 0)

436 ; Store the rightmost keyvalue into r16
437 0000e7 2b01                       or r16, r17
438 ; Adds $30 to the keyvalue, which turn the keyvalue into ASCII
439 0000e8 9309                       st Y+, r16

440 ; Put the value into the display buffer
441
442 0000e9 e200                       ldi r16, ' '
443 0000ea 9309                       st Y+, r16
444
445 0000eb 9100 010c                  lds r16, line + 0
446 0000ed 9309                       st Y+, r16
447
448 0000ee 940e 02b2                  call update_lcd_dog

449 ; update the display
450 0000f0 9508                       ret
451
452
453                                  ;*****************************************
                      **
454                                  ;SUBROUTINE FOR DISPLAYING THE INPUT TO LCD
                      (LINE 2)
455                                  ;*****************************************
                      **
456                                  display_line_2:
457 0000f1 e0d1                      ldi YH, high (dsp_buff_2) ; Load YH and YL
       as a pointer to 1st
```

```
458 0000f2 e2c3                          ldi YL, low (dsp_buff_2)  ; byte of
      dsp_buff_1 (Note - assuming
459                                                                ; (dsp_buff_1
                     for now).
460
461 0000f3 e704                          ldi r16, 't'
462 0000f4 9309                          st Y+, r16
463 0000f5 e200                          ldi r16, ' '
464 0000f6 9309                          st Y+, r16
465 0000f7 e30d                          ldi r16, '='
466 0000f8 9309                          st Y+, r16
467 0000f9 e200                          ldi r16, ' '
468 0000fa 9309                          st Y+, r16
469
470 0000fb e310                          ldi r17, $30
471 ; Load $30 into r16
472                                      ; store the ascii representation of the
473                                      digit in the buffer
474 0000fc 9100 0106                     lds r16, (pulse_width_bcd + 2)
475 ; Store the leftmost keyvalue into r16
476
477 0000fe 2b01                          or r16, r17
478 ; Adds $30 to the keyvalue, which turn the keyvalue into ASCII
479 0000ff 9309                          st Y+, r16

480 ; Put the value into the display buffer
481
482 000100 9100 0105                     lds r16, (pulse_width_bcd + 1)
483 ;
484 000102 2b01                          or r16, r17
485 ; Adds $30 to the keyvalue, which turn the keyvalue into ASCII
486 000103 9309                          st Y+, r16
487
488 000104 9100 0104                     lds r16, (pulse_width_bcd + 0)
489 ; Store the rightmost keyvalue into r16
490 000106 2b01                          or r16, r17
491 ; Adds $30 to the keyvalue, which turn the keyvalue into ASCII
492 000107 9309                          st Y+, r16

493
494 000108 e200                          ldi r16, ' '
495 000109 9309                          st Y+, r16
496
497 00010a 9100 010d                     lds r16, line + 1
498 00010c 9309                          st Y+, r16
499
500 00010d 940e 02b2                     call update_lcd_dog
              ; update the display
501 00010f 9508                          ret
```

```
502
503
504                             ;****************************************** ↵
                    **
505                             ;SUBROUTINE FOR DISPLAYING THE INPUT TO LCD  ↵
                    (LINE 3)
506                             ;****************************************** ↵
                    **
507                             display_line_3:
508 000110 e0d1                    ldi YH, high (dsp_buff_3) ; Load YH and YL ↵
     as a pointer to 1st
509 000111 e3c3                     ldi YL, low (dsp_buff_3)  ; byte of       ↵
    dsp_buff_1 (Note - assuming
510                                                           ; (dsp_buff_1   ↵
                    for now).
511
512 000112 e604                    ldi r16, 'd'
513 000113 9309                    st Y+, r16
514 000114 e200                    ldi r16, ' '
515 000115 9309                    st Y+, r16
516 000116 e30d                    ldi r16, '='
517 000117 9309                    st Y+, r16
518 000118 e200                    ldi r16, ' '
519 000119 9309                    st Y+, r16
520
521 00011a e310                    ldi r17, $30                           ;
522                                ; store the ascii representation of the    ↵
                    digit in the buffer
523 00011b 9100 010a               lds r16, (pulse_delay_bcd + 2)         ; ↵
    Store the leftmost keyvalue into r16
524
525 00011d 2b01                    or r16, r17
526 00011e 9309                    st Y+, r16                               ↵

527
528 00011f 9100 0109               lds r16, (pulse_delay_bcd + 1)         ;
529 000121 2b01                    or r16, r17
530 000122 9309                    st Y+, r16
531
532 000123 9100 0108               lds r16, (pulse_delay_bcd + 0)
533 000125 2b01                    or r16, r17
534 000126 9309                    st Y+, r16                               ↵

535
536 000127 e200                    ldi r16, ' '
537 000128 9309                    st Y+, r16
538
539 000129 9100 010e               lds r16, line + 2
540 00012b 9309                    st Y+, r16
```

```
541
542  00012c 940e 02b2                   call update_lcd_dog                     ↵
                ; update the display
543  00012e 9508                        ret
544
545
546                                     ;****************************************** ↵
                        ***
547                                     ;SUBROUTINE FOR STORING THE VALUE (LINE 1)
548                                     ;****************************************** ↵
                        **
549                                     store_value_line_1:
550                                        ;r18 is the value read by the input
551  00012f 9120 010f                   lds r18, keyvalue
552  000131 9100 0101                   lds r16, burst_count_setting_bcd + 1
553  000133 9300 0102                   sts burst_count_setting_bcd + 2, r16
554
555  000135 9100 0100                   lds r16, burst_count_setting_bcd + 0
556  000137 9300 0101                   sts burst_count_setting_bcd + 1, r16
557
558  000139 9320 0100                   sts burst_count_setting_bcd + 0, r18
559  00013b 9508                        ret
560
561
562                                     ;****************************************** ↵
                        ***
563                                     ;SUBROUTINE FOR STORING THE VALUE (LINE 2)
564                                     ;****************************************** ↵
                        **
565                                     store_value_line_2:
566                                        ;r18 is the value read by the input
567  00013c 9120 010f                   lds r18, keyvalue
568  00013e 9100 0105                   lds r16, pulse_width_bcd + 1
569  000140 9300 0106                   sts pulse_width_bcd + 2, r16
570
571  000142 9100 0104                   lds r16, pulse_width_bcd + 0
572  000144 9300 0105                   sts pulse_width_bcd + 1, r16
573
574  000146 9320 0104                   sts pulse_width_bcd + 0, r18
575  000148 9508                        ret
576
577
578                                     ;****************************************** ↵
                        ***
579                                     ;SUBROUTINE FOR STORING THE VALUE (LINE 3)
580                                     ;****************************************** ↵
                        **
581                                     store_value_line_3:
582                                        ;r18 is the value read by the input
```

```
000149 9120 010f                    lds r18, keyvalue
00014b 9100 0109                    lds r16, pulse_delay_bcd + 1    ; Load r16 with the middle digit
00014d 9300 010a                    sts pulse_delay_bcd + 2, r16    ; Put the middle digit into the leftmost digit

00014f 9100 0108                    lds r16, pulse_delay_bcd + 0    ; Load r16 with the Rightmost digit
000151 9300 0109                    sts pulse_delay_bcd + 1, r16    ; Put the rightmost digit into the middle digit

000153 9320 0108                    sts pulse_delay_bcd + 0, r18    ; Store the new number into the rightmost digit
000155 9508                         ret


                                    ;************************************
                                    ;SUBROUTINE FOR WHEN THE # IS MOVED DOWN
                                    ;************************************
                                    switch_down_and_display:
000156 940e 0160                       call switch_lines_down
000158 940e 00bc                       call update_all
00015a 9508                            ret


                                    ;************************************
                                    ;SUBROUTINE FOR WHEN THE # IS MOVED UP
                                    ;************************************
                                    switch_up_and_display:
00015b 940e 016d                       call switch_lines_up
00015d 940e 00bc                       call update_all
00015f 9508                            ret

                                    ;************************************
                                    ;SUBROUTINE FOR SWITCHING LINES DOWN
                                    ;************************************
                                    switch_lines_down:
000160 9100 010c                       lds r16, line
000162 9110 010d                       lds r17, line + 1
000164 9120 010e                       lds r18, line + 2

000166 9300 010d                       sts line + 1, r16
000168 9310 010e                       sts line + 2, r17
00016a 9320 010c                       sts line, r18

00016c 9508                            ret


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

```
627                                 ;SUBROUTINE FOR SWITCHING LINES UP
628                                 ;************************************
629                                 switch_lines_up:
630  00016d 9100 010c                  lds r16, line
631  00016f 9110 010d                  lds r17, line + 1
632  000171 9120 010e                  lds r18, line + 2
633
634  000173 9300 010e                  sts line + 2, r16
635  000175 9320 010d                  sts line + 1, r18
636  000177 9310 010c                  sts line, r17
637
638  000179 9508                       ret
639
640
641                                 ;*********************************
642                                 ;SUBROUTINE FOR RETRIEVING INPUT(PART 2)
643                                 ;*********************************
644                                 get_key_value:
645  00017a b129                       in r18, PIND            ; Store the Input
       into r18
646  00017b 7f20                       andi r18, $F0           ; Clear the low
       nibble of r18
647  00017c 9522                       swap r18                ; Swap the nibble
648  00017d 940e 0180                  call keycode2keyvalue   ; Convert the
       input into HEXVALUES (NOT ASCII)
649  00017f 9508                       ret
650
651
652                                 ;*******************************
653                                 ;SUBROUTINE FOR LOOKUP TABLE
654                                 ;*******************************
655                                 keycode2keyvalue:
656                                 lookup:
657  000180 e0f4                       ldi ZH, high (keytable * 2)     ;set Z to
       point to start of table
658  000181 eaee                       ldi ZL, low (keytable * 2)
659  000182 e000                       ldi r16, $00                    ;add
       offset to Z pointer
660  000183 0fe2                       add ZL, r18
       ;originally r18
661  000184 0ff0                       add ZH, r16
662  000185 9124                       lpm r18, Z
663  000186 9508                       ret
664
665
666                                 ;*************************
667                                 ;SUBROUTINE FOR DELAY
668                                 ;*************************
669                                 var_delay: ;delay for ATmega324 @ 1MHz = r16
```

```
                        * 0.1 ms
670                                     outer_loop:
671  000187 e210                           ldi r17, 32
672                                     inner_loop:
673  000188 951a                           dec r17
674  000189 f7f1                           brne inner_loop
675  00018a 950a                           dec r16
676  00018b f7d9                           brne outer_loop
677  00018c 9508                           ret
678
679
680                                     ;*********************************
681                                     ;SUBROUTINE FOR BUZZER
682                                     ;*********************************
683                                     buzz:
684  00018d 9a16                           sbi PORTA, 6
685  00018e ef0f                           ldi r16 , 255   ; For delay
686  00018f 940e 0187                      call var_delay
687  000191 9816                           cbi PORTA, 6
688  000192 9508                           ret
689
690
691                                     ;*****************************************
                        *********
692                                     ;SUBROUTINE convert all three unpacked_bcd to
                         binary
693                                     ;*****************************************
                        ********
694                                     convert:
695  000193 940e 01a6                      call convert_line1_to_Packed_BCD
696  000195 940e 0207                      call BCD2bin16
697  000197 92e0 0103                      sts burst_count, r14              ;
       Store the value of r17 into burst_count_bin
698
699
700  000199 940e 01b2                      call convert_line2_to_Packed_BCD
701  00019b 940e 0207                      call BCD2bin16
702  00019d 92e0 0107                      sts pulse_width, r14
703
704
705  00019f 940e 01be                      call convert_line3_to_Packed_BCD
706  0001a1 940e 0207                      call BCD2bin16
707  0001a3 92e0 010b                      sts pulse_delay, r14
708
709  0001a5 9508                           ret
710
711
712                                     ;*****************************************
                        *********
```

```
                                    ;SUBROUTINE convert the line 1 to PACKED BCD
                                    ;*****************************************
                        ********
                                    convert_line1_to_Packed_BCD:
0001a6 9100 0100                       lds r16, burst_count_setting_bcd
0001a8 9110 0101                       lds r17, burst_count_setting_bcd + 1
0001aa 9120 0102                       lds r18, burst_count_setting_bcd + 2

0001ac 9512                            swap r17

0001ad 2b01                            or r16, r17
0001ae 702f                            andi r18, $0F
0001af 2f12                            mov r17, r18
0001b0 e020                            ldi r18, $00

0001b1 9508                            ret


                                    ;*****************************************
                        *********
                                    ;SUBROUTINE convert the line 2 to PACKED BCD
                                    ;*****************************************
                        ********
                                    convert_line2_to_Packed_BCD:
0001b2 9100 0104                       lds r16, pulse_width_bcd
0001b4 9110 0105                       lds r17, pulse_width_bcd + 1
0001b6 9120 0106                       lds r18, pulse_width_bcd + 2

0001b8 9512                            swap r17

0001b9 2b01                            or r16, r17
0001ba 702f                            andi r18, $0F
0001bb 2f12                            mov r17, r18
0001bc e020                            ldi r18, $00

0001bd 9508                            ret


                                    ;*****************************************
                        *********
                                    ;SUBROUTINE convert the line 3 to PACKED BCD
                                    ;*****************************************
                        ********
                                    convert_line3_to_Packed_BCD:
0001be 9100 0108                       lds r16, pulse_delay_bcd
0001c0 9110 0109                       lds r17, pulse_delay_bcd + 1
0001c2 9120 010a                       lds r18, pulse_delay_bcd + 2

0001c4 9512                            swap r17
```

```
755  0001c5 2b01                    or r16, r17
756  0001c6 702f                    andi r18, $0F
757  0001c7 2f12                    mov r17, r18
758  0001c8 e020                    ldi r18, $00
759
760  0001c9 9508                    ret
761
762
763                                 ;***************************************** ↵
                     **
764                                 ;SUBROUTINE FOR PULSE GENERATOR
765                                 ;***************************************** ↵
                     **
766                                 pulse_generator:
767  0001ca 9100 0107               lds r16, pulse_width
768  0001cc 9a17                    sbi PORTA, 7
769  0001cd dfb9                    rcall var_delay
770  0001ce 9817                    cbi PORTA, 7
771  0001cf 9100 010b               lds r16, pulse_delay
772  0001d1 dfb5                    rcall var_delay
773  0001d2 953a                    dec r19
774  0001d3 f7b1                    brne pulse_generator
775
776  0001d4 940e 01eb               call clear_flags
777
778  0001d6 9508                    ret
779
780
781                                 ;***************************************** ↵
                     **
782                                 ;SUBROUTINE FOR GENERATING A PULSES
783                                 ;***************************************** ↵
                     **
784                                 generate_a_pulse:
785  0001d7 e00a                    ldi r16, 10                    ;      ↵
       pulse width
786  0001d8 9a17                    sbi PORTA, 7                   ; set  ↵
       bit for pulse
787  0001d9 dfad                    rcall var_delay
788  0001da 9817                    cbi PORTA, 7                   ;      ↵
       clear bit for pulse
789  0001db 9508                    ret
790
791
792                                 ;***************************************** ↵
                     **
793                                 ;SUBROUTINE FOR ASSIGNING FLAGS
794                                 ;***************************************** ↵
```

```
                        **
795                                 update_flags:
796  0001dc e001                       ldi r16, 1                  ; Set the
       make_pulse flag
797  0001dd 9300 0110                  sts make_pulse, r16
798
799  0001df 9100 0103                  lds r16, burst_count
800  0001e1 3000                       cpi r16, $00
801  0001e2 f021                       breq burst_is_zero
802
803  0001e3 e000                       ldi r16, 0
804  0001e4 9300 0111                  sts is_burst_zero, r16
805
806                                 please_go_here:
807  0001e6 9508                       ret
808
809                                 burst_is_zero:
810  0001e7 e001                       ldi r16, 1
811  0001e8 9300 0111                  sts is_burst_zero, r16
812  0001ea cffb                       rjmp please_go_here
813
814
815                                 ;****************************************
                        **
816                                 ;SUBROUTINE FOR CLEARING FLAGS
817                                 ;****************************************
                        **
818                                 clear_flags:
819  0001eb e000                       ldi r16, 0
820  0001ec 9300 0110                  sts make_pulse, r16                 ;
       Reset the make_pluse to zero
821
822  0001ee 9300 0102                  sts burst_count_setting_bcd + 2, r16
823  0001f0 9300 0101                  sts burst_count_setting_bcd + 1, r16
824  0001f2 9300 0100                  sts burst_count_setting_bcd + 0, r16
825
826  0001f4 9508                       ret
827
828
829                                 ;****************************************
                        ******************************
830                                 ;*
831                                 ;* "BCD2bin16" - BCD to 16-Bit Binary
                        Conversion
832                                 ;*
833                                 ;* This subroutine converts a 5-digit packed
                        BCD number represented by
834                                 ;* 3 bytes (fBCD2:fBCD1:fBCD0) to a 16-bit
                        number (tbinH:tbinL).
```

```
835                             ;* MSD of the 5-digit number must be placed in the lowermost nibble of fBCD2.
836                             ;*
837                             ;* Let "abcde" denote the 5-digit number. The conversion is done by
838                             ;* computing the formula: 10(10(10(10a+b)+c)+d)+e.
839                             ;* The subroutine "mul10a"/"mul10b" does the multiply-and-add operation
840                             ;* which is repeated four times during the computation.
841                             ;*
842                             ;* Number of words  :30
843                             ;* Number of cycles    :108
844                             ;* Low registers used  :4 (copyL,copyH,mp10L/tbinL,mp10H/tbinH)
845                             ;* High registers used  :4 (fBCD0,fBCD1,fBCD2,adder)
846                             ;*
847                             ;*****************************************************************************
848
849                             ;***** "mul10a"/"mul10b" Subroutine Register Variables
850
851                             .def   copyL   =r12        ;temporary register
852                             .def   copyH   =r13        ;temporary register
853                             .def   mp10L   =r14        ;Low byte of number to be multiplied by 10
854                             .def   mp10H   =r15        ;High byte of number to be multiplied by 10
855                             .def   adder   =r19        ;value to add after multiplication
856
857                             ;***** Code
858
859                             mul10a:    ;***** multiplies "mp10H:mp10L" with 10 and adds "adder" high nibble
860 0001f5 9532                    swap    adder
861                             mul10b:    ;***** multiplies "mp10H:mp10L" with 10 and adds "adder" low nibble
862 0001f6 2cce                    mov copyL,mp10L ;make copy
863 0001f7 2cdf                    mov copyH,mp10H
864 0001f8 0cee                    lsl mp10L       ;multiply original by 2
865 0001f9 1cff                    rol mp10H
866 0001fa 0ccc                    lsl copyL       ;multiply copy by 2
867 0001fb 1cdd                    rol copyH
```

```
868  0001fc 0ccc                         lsl copyL       ;multiply copy by 2 (4)
869  0001fd 1cdd                         rol copyH
870  0001fe 0ccc                         lsl copyL       ;multiply copy by 2 (8)
871  0001ff 1cdd                         rol copyH
872  000200 0cec                         add mp10L,copyL ;add copy to original
873  000201 1cfd                         adc mp10H,copyH
874  000202 703f                         andi    adder,0x0f  ;mask away upper
       nibble of adder
875  000203 0ee3                         add mp10L,adder ;add lower nibble of adder
876  000204 f408                         brcc    m10_1       ;if carry not cleared
877  000205 94f3                         inc mp10H       ;   inc high byte
878  000206 9508                       m10_1:  ret
879
880                                    ;***** Main Routine Register Variables
881
882                                     .def   tbinL   =r14        ;Low byte of
                       binary result (same as mp10L)
883                                     .def   tbinH   =r15        ;High byte of
                       binary result (same as mp10H)
884                                     .def   fBCD0   =r16        ;BCD value digits
                       1 and 0
885                                     .def   fBCD1   =r17        ;BCD value digits
                       2 and 3
886                                     .def   fBCD2   =r18        ;BCD value digit 5
887
888                                    ;***** Code
889
890                                    BCD2bin16:
891  000207 702f                          andi    fBCD2,0x0f  ;mask away upper
       nibble of fBCD2
892  000208 24ff                          clr mp10H
893  000209 2ee2                          mov mp10L,fBCD2 ;mp10H:mp10L = a
894  00020a 2f31                          mov adder,fBCD1
895  00020b dfe9                          rcall   mul10a      ;mp10H:mp10L = 10a+b
896  00020c 2f31                          mov adder,fBCD1
897  00020d dfe8                          rcall   mul10b      ;mp10H:mp10L = 10(10a
       +b)+c
898  00020e 2f30                          mov adder,fBCD0
899  00020f dfe5                          rcall   mul10a      ;mp10H:mp10L = 10(10
       (10a+b)+c)+d
900  000210 2f30                          mov adder,fBCD0
901  000211 dfe4                          rcall   mul10b      ;mp10H:mp10L = 10(10
       (10(10a+b)+c)+d)+e
902  000212 9508                          ret
903
904
905                                    ;****************************************
                       ******************************
906                                    ;*
```

```
907                                 ;* "keypress_ISR" - Check Interrupts at INT0
908                                 ;*
909                                 ;* Description: Get the keyvalue if the key
                    is pressed, the keyvalue
910                                 ;* Author:                 Seyi Olajuyi &
                    Bassel El Amine
911                                 ;* Version:
912                                 ;* Last updated:           11/21/19
913                                 ;* Target:                 ATmega324A
914                                 ;* Number of words:
915                                 ;* Number of cycles:       N/A
916                                 ;* Low registers modified:  none
917                                 ;* High registers modified: none
918                                 ;*
919                                 ;* Parameters:
920                                 ;* Notes:
921                                 ;*
922                                 ;*****************************************
                    ******************************
923
924                                     ;INT0 interrupt service routine
925                                 keypress_ISR:
926 000213 932f                        push r18
927 000214 930f                         push r16            ;save r16
928 000215 b70f                        in r16, SREG        ;save SREG
929 000216 930f                         push r16
930
931 000217 e001                        ldi r16, (1 <<INTF0)
932 000218 bb0c                        out EIFR, r16
933
934 000219 df60                        rcall get_key_value
935 00021a 302a                        cpi r18, $0A
936 00021b f088                        brlo skip_line_1
937
938 00021c f0b1                        breq input_clear
939
940 00021d 302c                        cpi r18, $0C
941 00021e f0c1                        breq input_enter
942
943 00021f 302f                        cpi r18, $0F
944 000220 f0d1                        breq input_up
945
946 000221 302e                        cpi r18, $0E
947 000222 f0e1                        breq input_down
948
949 000223 ef0f                        ldi r16, eol
950 000224 9300 0112                   sts input, r16
951
952                                 restore_values_1:
```

```
000226 940e 009e                    call fsm

000228 910f                          pop r16             ;restore SREG
000229 bf0f                          out SREG,r16
00022a 910f                         pop r16             ;restore r16
00022b 912f                         pop r18             ;restore r18

00022c 9518                          reti                ;return from interrupt

                                skip_line_1:
00022d 9320 010f                    sts keyvalue, r18                    ; if key value is a number

00022f e000                         ldi r16, number                      ; input is assign as a number
000230 9300 0112                    sts input, r16

                                    ;;;;;;;;rcall store_value
000232 cff3                         rjmp restore_values_1

                                input_clear:
000233 e002                         ldi r16, clear
000234 9300 0112                    sts input, r16
000236 cfef                         rjmp restore_values_1

                                input_enter:
000237 e001                         ldi r16, enter
000238 9300 0112                    sts input, r16
00023a cfeb                         rjmp restore_values_1

                                input_up:
00023b e004                         ldi r16, up_arrow
00023c 9300 0112                    sts input, r16
00023e cfe7                         rjmp restore_values_1

                                input_down:
00023f e005                         ldi r16, down_arrow
000240 9300 0112                    sts input, r16
000242 cfe3                         rjmp restore_values_1


                                ;******************************************** ********************************
                                ;*
                                ;* "pb_press_ISR" - Check Interrupts at INT1
                                ;*
                                ;* Description: Checks if the push button is pressed
```

```
 997                                ;*
 998                                ;* Author:                 Ken Short
 999                                ;* Version:
1000                                ;* Last updated:           11/21/19
1001                                ;* Target:                 ATmega324A
1002                                ;* Number of words:
1003                                ;* Number of cycles:       16
1004                                ;* Low registers modified:  none
1005                                ;* High registers modified: none
1006                                ;*
1007                                ;* Parameters:  Uses PORTB register to hold
                                the count and drive LEDs
1008                                ;* connected to that port.
1009                                ;*
1010                                ;* Notes:
1011                                ;*
1012                                ;*****************************************
                                ******************************
1013
1014                                    ;INT1 interrupt service routine
1015                                pb_press_ISR:
1016  000243 930f                       push r16            ;save r16
1017  000244 b70f                       in r16, SREG        ;save SREG
1018  000245 930f                        push r16
1019
1020                                wait_for_bounce_1:
1021  000246 994b                       sbic PIND, 3
1022  000247 cffe                       rjmp wait_for_bounce_1
1023  000248 e604                       ldi r16, 100
1024  000249 df3d                       rcall var_delay
1025  00024a 994b                       sbic PIND, 3
1026  00024b cffa                       rjmp wait_for_bounce_1
1027
1028  00024c e002                       ldi r16, (1 <<INTF1)
1029  00024d bb0c                       out EIFR, r16
1030
1031  00024e e003                       ldi r16 , $03                       ; Set
        polling_for_button
1032  00024f 9300 0112                  sts input, r16      ; Use to find out if
        the button was pressed
1033
1034                                restore_value_2:
1035  000251 940e 009e                  call fsm
1036  000253 910f                        pop r16             ;restore SREG
1037  000254 bf0f                        out SREG,r16
1038  000255 910f                       pop r16             ;restore r16
1039
1040  000256 9518                        reti                ;return from
        interrupt
```

```
1041
1042
1043
1044
1045  000257 0201
1046  000258 0f03
1047  000259 0504
1048  00025a 0e06
1049  00025b 0807
1050  00025c 0d09                     keytable: .db $01, $02, $03, $0F, $04, $05,    ↵
        $06, $0E, $07, $08, $09, $0D
1051  00025d 000a
1052  00025e 0c0b                             .db $0A, $00, $0B, $0C
1053
1054
1055                                  .list
1056
1057
1058
1059  RESOURCE USE INFORMATION
1060  ------------------------
1061
1062  Notice:
1063  The register and instruction counts are symbol table hit counts,
1064  and hence implicitly used resources are not counted, eg, the
1065  'lpm' instruction without operands implicitly uses r0 and z,
1066  none of which are counted.
1067
1068  x,y,z are separate entities in the symbol table and are
1069  counted separately from r26..r31 here.
1070
1071  .dseg memory usage only counts static data declared with .byte
1072
1073  "ATmega324A" register use summary:
1074  x  :   0 y  :  27 z  :  10 r0 :   0 r1 :   0 r2 :   0 r3 :   0 r4 :   0
1075  r5 :   0 r6 :   0 r7 :   0 r8 :   0 r9 :   0 r10:   0 r11:   0 r12:   5
1076  r13:   5 r14:   8 r15:   5 r16: 196 r17:  44 r18:  39 r19:   8 r20:  10
1077  r21:   2 r22:   2 r23:   2 r24:   7 r25:   5 r26:   2 r27:   0 r28:   4
1078  r29:   4 r30:  12 r31:  10
1079  Registers used: 21 out of 35 (60.0%)
1080
1081  "ATmega324A" instruction use summary:
1082  .lds  :   0 .sts  :   0 adc   :   2 add   :   5 adiw  :   2 and   :   0
1083  andi  :   6 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
1084  brcc  :   1 brcs  :   0 break :   0 breq  :   7 brge  :   0 brhc  :   0
1085  brhs  :   0 brid  :   0 brie  :   0 brlo  :   1 brlt  :   0 brmi  :   0
1086  brne  :  11 brpl  :   0 brsh  :   0 brtc  :   0 brts  :   0 brvc  :   0
1087  brvs  :   0 bset  :   0 bst   :   0 call  :  29 cbi   :   6 cbr   :   0
1088  clc   :   0 clh   :   0 cli   :   0 cln   :   0 clr   :   1 cls   :   0
```

```
1089  clt   :   0 clv   :   0 clz   :   0 com   :   0 cp    :   1 cpc   :   0
1090  cpi   :   7 cpse  :   0 dec   :  10 eor   :   0 fmul  :   0 fmuls :   0
1091  fmulsu:   0 icall :   1 ijmp  :   0 in    :  12 inc   :   1 jmp   :   0
1092  ld    :   3 ldd   :   0 ldi   :  88 lds   :  42 lpm   :   9 lsl   :   4
1093  lsr   :   0 mov   :  14 movw  :   0 mul   :   0 muls  :   0 mulsu :   0
1094  neg   :   0 nop   :   2 or    :  12 ori   :   0 out   :  12 pop   :  11
1095  push  :  11 rcall :  47 ret   :  38 reti  :   2 rjmp  :  17 rol   :   4
1096  ror   :   0 sbc   :   0 sbci  :   0 sbi   :  12 sbic  :   2 sbis  :   0
1097  sbiw  :   0 sbr   :   0 sbrc  :   0 sbrs  :   3 sec   :   0 seh   :   0
1098  sei   :   1 sen   :   0 ser   :   0 ses   :   0 set   :   0 sev   :   0
1099  sez   :   0 sleep :   0 spm   :   0 st    :  28 std   :   0 sts   :  46
1100  sub   :   0 subi  :   0 swap  :   5 tst   :   0 wdr   :   0
1101  Instructions used: 40 out of 113 (35.4%)
1102
1103  "ATmega324A" memory use summary [bytes]:
1104  Segment   Begin    End      Code   Data   Used    Size   Use%
1105  ----------------------------------------------------------------
1106  [.cseg] 0x000000 0x0005ae   1260    190   1450   32768   4.4%
1107  [.dseg] 0x000100 0x000143      0     67     67    2048   3.3%
1108  [.eseg] 0x000000 0x000000      0      0      0    1024   0.0%
1109
1110  Assembly complete, 0 errors, 2 warnings
1111
```