```

AVRASM ver. 2.2.7  C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\ppg_III_po s_edge_ints
  \main.asm Thu Nov 21 20:49:40 2019

C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\ppg_III_pos_edge_ints\main.as m(18):
  Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.3.300\avrasm\inc
  \m324adef.inc'
C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\ppg_III_pos_edge_ints\main.as m(360):
  warning: Register r14 already defined by the .DEF directive
C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\ppg_III_pos_edge_ints\main.as m(361):
  warning: Register r15 already defined by the .DEF directive
C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\ppg_III_pos_edge_ints\main.as m(489):
  Including file 'C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\ppg_III_pos_e dge_ints
  \lcd_dog_asm_driver_m324a.inc'
C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\ppg_III_pos_edge_ints\main.as m(18):
  Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.3.300\avrasm\inc
  \m324adef.inc'
C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\ppg_III_pos_edge_ints\main.as m(489):
  Including file 'C:\Users\Seyi Olajuyi\Documents\Atmel Studio\7.0\ppg_III_pos_edge_ints\ppg_III_pos_e dge_ints
  \lcd_dog_asm_driver_m324a.inc'


                                 ;*
                                 ;* Title: ppg_III_pos_edge_ints
                                 ;* Author: Seyi Olajuyi & Bassel El Amine
                                 ;* Version: 1.0
                                 ;* Last updated: 2019/11/21
                                 ;* Target: ATmega 324
                                 ;*
                                 ;* DESCRIPTION
                                 ;*
                                 ;*
                                 ;*
                                 ;*
                                 ;* VERSION HISTORY
                                 ;* 1.0 Original version
                                 ;****************************************************************** *********
                                 .list
                                 
                                 .dseg  ;The variable below are in SRAM
000100                           burst_count_setting_bcd:       .byte 3; setting unpacked BCD ;THIS HAS THREE BTYE allocated to the variable name
000103                           burst_count:                   .byte 1; pulses left to generated in burst
000104                           keyvalue:                      .byte 1; stores the keyvalue into a variable
000105                           polling_for_keypad:                .byte 1; used to store the values in the external interrupt flag register
000106                           polling_for_button:                .byte 1; used to store the values in the external interrupt flag register
                                 
                                 
                                 ;burst_count_setting_bcd is right most digit and
                                 ; (burst_count_setting_bcd + 2) is the left most digit
                                 
                                 .cseg
                                 reset:
                                 .org RESET              ;reset interrupt vector
000000 c004                          rjmp start          ;program starts here at reset
                                 .org INT0addr           ;INT0 interrupt vector
000002 c0cc                          rjmp keypress_ISR
                                 .org INT1addr
000004 c0dd                         rjmp pb_press_ISR
                                 
                                 
                                 ;****************************************************************** ****
                                 ;************* M A I N   A P P L I C A T I O N   C O D E  ********* ****
                                 ;****************************************************************** ****
                                 
                                 
                                 start:
000005 ef0f                          ldi r16, LOW(RAMEND)    ;initialize SP to point to top of stack
```

```
000006 bf0d                           out SPL, r16
000007 e008                           ldi r16, HIGH(RAMEND)
000008 bf0e                           out SPH, r16

000009 e00f                           ldi r16, (1 << ISC00) | (1 << ISC01) | (1 << ISC10) | (1 << ISC 11)
00000a 9300 0069                      sts EICRA, r16
00000c e003                          ldi r16, $03        ; Enable interrupt request at INTO & INT1
00000d bb0d                           out EIMSK, r16

00000e ef0f                           ldi r16, $ff      ; load r16 with all 1s.
00000f b904                           out DDRB, r16     ; set portB = output

000010 e003                          ldi r16, $03        ; Set pin 0 & pin 1 to output, everyother pin is an input
000011 b90a                          out DDRD, r16

000012 9a0f                          sbi DDRA, 7         ; Set pin 7 on PORTA to output

000013 9a2c                           sbi portB, 4      ; set /SS of DOG LCD = 1 (Deselected)

000014 d11c                           rcall init_lcd_dog        ; init display, using SPI serial inte rface
000015 d059                          rcall clr_dsp_buffs       ; clear all three buffer lines
000016 d138                          rcall update_lcd_dog      ; update the display

000017 e0d1                          ldi YH, high (dsp_buff_1) ; Load YH and YL as a pointer to 1st
000018 e0c7                           ldi YL, low (dsp_buff_1)  ; byte of dsp_buff_1 (Note - assuming
                                                                ; (dsp_buff_1 for now).

                                     ; RESET THE VARIABLES WITH ZERO
000019 e010                          ldi r17, $00
00001a 9310 0102                     sts burst_count_setting_bcd + 2, r17
00001c 9310 0101                     sts burst_count_setting_bcd + 1, r17
00001e 9310 0100                     sts burst_count_setting_bcd + 0, r17

000020 9310 0105                     sts polling_for_keypad, r17
```

```
000022 9310 0106                    sts polling_for_button, r17

000024 9310 0103                    sts burst_count, r17

000026 9310 0104                    sts keyvalue, r17

000028 9478                         sei                 ;set global interrupt enable

                                 ;****************************************************************
                                 ;***********************CODE BEGINS******************************
                                 ;****************************************************************

                                 ;This runs after the peripherals are initalized
                                 state_1:

                                    ; Reset the polling for the keypad press, this is important because state_2 jumps to this label
000029 e010                         ldi r17, $00
00002a 9310 0105                    sts polling_for_keypad, r17


00002c d04a                         rcall display_the_value
                                 ;This Convert the registers to PACKED BCD
                                 convert_to_Packed_BCD:
00002d 9100 0100                    lds r16, burst_count_setting_bcd        ; Retrieve the value store in the FIRST byte of burst_count_setting_bcd   and store it in r16
00002f 9110 0101                    lds r17, burst_count_setting_bcd + 1    ; Retrieve the value store in the SECOND byte of burst_count_setting_bcd and store it in r17
000031 9120 0102                    lds r18, burst_count_setting_bcd + 2    ; Retrieve the value store in the THIRD byte of burst_count_setting_bcd and store it in r18

000033 9512                         swap r17                                ; Swap the nibble in r17

000034 2b01                         or r16, r17                             ; Or r16 & r17, Combine the two contents of two registers into one register (r16)
```

```
000035 702f                 andi r18, $0F                       ; AND r18 & $0F, clear the high nibble of r18
000036 2f12                 mov r17, r18                        ; Move the content of r18 into r17
000037 e020                 ldi r18, $00                        ; Load r18 with zero, this will be useful when we are trying to convert
                                                                ; Packed BCD into a 16-bit      R16 0x0a    byte{registers}@R16bianry value
                          ;This converts the Packed BCD into the 16-bit binary
                          convert_BCD_to_Binary:
000038 940e 00c3            call BCD2bin16

00003a 2d3e                 mov r19, r14                        ; Moves the low byte of the 16-bit binary value into r17
00003b 9330 0103            sts burst_count, r19                ; Store the value of r17 into burst_count_bin

00003d 9100 0106            lds r16, polling_for_button
00003f 3001                 cpi r16, 1
000040 f009                 breq state_2

000041 cfe7                 rjmp state_1

                          state_2:
000042 e000                 ldi r16, 0                        ; Reset the flag that polls the push button
000043 9300 0106            sts polling_for_button, r16

                            ; Reinitialize the Burst count
000045 e00a                 ldi r16, 10                       ; Load ten into r16, This is to create the 1 ms delay
000046 9130 0103            lds r19, burst_count              ; This loads r19 with the orginal binary value

                          check_zero:
000048 3030                 cpi r19, $00                      ; Branch to generate a pulse if burst
```

```
  count = 0
000049 f0b1                         breq generate_a_pulse

                                 ;This generate a pulse that is supposed to be 1 ms wide
                                 pulse_generator:
00004a 9a17                         sbi PORTA, 7                         ; set bit for pulse
00004b d05f                         rcall var_delay
00004c e00a                         ldi r16, 10                          ; pulse width delay
00004d 9817                         cbi PORTA, 7                         ; clear bit for pulse
00004e d05c                         rcall var_delay
00004f e00a                         ldi r16, 10                          ; time between pulses delay
000050 953a                         dec r19                              ; decrement the binary value
000051 f7c1                         brne pulse_generator

                                 ;This part is reached if the burst count is equal to zero
                                 check_flag_2:

000052 9140 0106                    lds r20, polling_for_button
000054 9150 0105                    lds r21, polling_for_keypad

000056 3041                         cpi r20, 1                           ; Check if the pushbutton is pressed
000057 f351                         breq state_2

000058 3051                         cpi r21, 1                           ; Check if the pushbutton is pressed
000059 f009                         breq service_keypad_input
00005a cff7                         rjmp check_flag_2

                                 service_keypad_input:
00005b 9120 0104                    lds r18, keyvalue
00005d 302a                         cpi r18, $0A                         ; checks if the key value is equal to CLEAR
00005e f251                         breq state_1                         ; goes to the beginning if the key value is equal to CLEAR
00005f cff2                         rjmp check_flag_2                    ; goes back to generate another set of pulses
```

```

                                 ; This is branched if burst count is equal to zero
                                 generate_a_pulse:
000060 9a17                         sbi PORTA, 7                         ; set bit for pulse
000061 d049                         rcall var_delay
000062 e00a                         ldi r16, 10                          ; pulse width delay
000063 9817                         cbi PORTA, 7                         ; clear bit for pulse
000064 d046                         rcall var_delay
000065 e00a                         ldi r16, 10
                                                                         ; time between pulses delay
000066 9140 0105                    lds r20, polling_for_keypad
000068 ff40                         sbrs r20, 0                          ; Skips the rjmp instruction if the
  value in polling_for_keypad = 1
000069 cff6                         rjmp generate_a_pulse

00006a 9120 0104                    lds r18, keyvalue
00006c 302a                         cpi r18, $0A                         ; Check if key value is equal to clear
00006d f001                         breq prompt1

                                    prompt1:
00006e cfba                         rjmp state_1

                                 ;---------------------------- SUBROUTINES ------------------------- ---

                                 ;************************
                                 ;NAME:      clr_dsp_buffs
                                 ;FUNCTION:  Initializes dsp_buffers 1, 2, and 3 with blanks (0x20)
                                 ;ASSUMES:   Three CONTIGUOUS 16-byte dram based buffers named
                                 ;           dsp_buff_1, dsp_buff_2, dsp_buff_3.
                                 ;RETURNS:   nothing.
                                 ;MODIFIES:  r25,r26, Z-ptr
                                 ;CALLS:     none
                                 ;CALLED BY: main application and diagnostics
                                 ;****************************************************************** **
                                 clr_dsp_buffs:
```

```
00006f e390                           ldi R25, 48           ; load total length of both buffer.
000070 e2a0                           ldi R26, ' '              ; load blank/space into R26.
000071 e0f1                           ldi ZH, high (dsp_buff_1) ; Load ZH and ZL as a pointer to 1st
000072 e0e7                           ldi ZL, low (dsp_buff_1)  ; byte of buffer for line 1.

                                     ;set DDRAM address to 1st position of first line.
                                 store_bytes:
000073 93a1                           st  Z+, R26       ; store ' ' into 1st/next buffer byte and
                                                        ; auto inc ptr to next location.
000074 959a                           dec  R25          ;
000075 f7e9                           brne store_bytes  ; cont until r25=0, all bytes written.
000076 9508                           ret


                                 ;***********************************************
                                 ;SUBROUTINE FOR DISPLAYING THE INPUT TO LCD
                                 ;***********************************************
                                 display_the_value:
000077 e0d1                         ldi YH, high (dsp_buff_1) ; Load YH and YL as a pointer to 1st
000078 e0c7                          ldi YL, low (dsp_buff_1)  ; byte of dsp_buff_1 (Note - assuming
                                                               ; (dsp_buff_1 for now).

000079 e60e                         ldi r16, 'n'
00007a 9309                         st Y+, r16
00007b e200                         ldi r16, ' '
00007c 9309                         st Y+, r16
00007d e30d                         ldi r16, '='
00007e 9309                         st Y+, r16
00007f e200                         ldi r16, ' '
000080 9309                         st Y+, r16

000081 e310                         ldi r17, $30                            ; Load $30 into r16
                                    ; store the ascii representation of the digit in the buffer
000082 9100 0102                    lds r16, (burst_count_setting_bcd + 2)          ; Store the leftmost keyvalue into r16
```

```

000084 2b01         or r16, r17                 ; Adds $30 to the keyvalue, which turn the keyvalue into ASCII
000085 9309         st Y+, r16                      ; Put the value into the display buffer

000086 9100 0101    lds r16, (burst_count_setting_bcd + 1)      ;
000088 2b01         or r16, r17                 ; Adds $30 to the keyvalue, which turn the keyvalue into ASCII
000089 9309         st Y+, r16

00008a 9100 0100    lds r16, (burst_count_setting_bcd + 0)      ; Store the rightmost keyvalue into r16
00008c 2b01         or r16, r17                 ; Adds $30 to the keyvalue, which turn the keyvalue into ASCII
00008d 9309         st Y+, r16                      ; Put the value into the display buffer

00008e 940e 014f    call update_lcd_dog                 ; update the display
000090 9508         ret

                  ;***********************************************
                  ;SUBROUTINE FOR STORING THE VALUE INTO THE Variable
                  ;**********************************************
                  store_value:
                    ;r18 is the value read by the input

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 2)
                                 ;*********************************
                                 get_key_value:
00009c b129                         in r18, PIND                ; Store the Input into r18
00009d 7f20                         andi r18, $F0               ; Clear the low nibble of r18
00009e 9522                         swap r18                    ; Swap the nibble
00009f 940e 00a4                    call keycode2keyvalue       ; Convert the input into HEXVALUES (NOT ASCI I)
0000a1 9847                         cbi PORTC, 7                ; Clear the FLip Flop that is connected to the encoder
0000a2 9a47                         sbi PORTC, 7                ;
0000a3 9508                         ret

                                 ;********************************
                                 ;SUBROUTINE FOR LOOKUP TABLE
                                 ;******************************
                                 keycode2keyvalue:
                                 lookup:
0000a4 e0f1                         ldi ZH, high (keytable * 2)         ;set Z to point to start of table
0000a5 eee8                         ldi ZL, low (keytable * 2)
0000a6 e000                         ldi r16, $00                        ;add offset to Z pointer
0000a7 0fe2                         add ZL, r18                         ;originally r18
0000a8 0ff0                         add ZH, r16
0000a9 9124                         lpm r18, Z
0000aa 9508                         ret

                                 ;**************************
                                 ;************************
                                 ;SUBROUTINE FOR DELAY
                                 ;************************
                                 var_delay: ;delay for ATmega324 @ 1MHz = r16 * 0.1 ms
                                 outer_loop:; r16 should equal to 10
```

```
0000ab e210                         ldi r17, 32
                                 inner_loop:
0000ac 951a                         dec r17
0000ad f7f1                         brne inner_loop
0000ae 950a                         dec r16
0000af f7d9                         brne outer_loop
0000b0 9508                         ret


                                 ;****************************************************************** *********
                                 ;*
                                 ;* "BCD2bin16" - BCD to 16-Bit Binary Conversion
                                 ;*
                                 ;* This subroutine converts a 5-digit packed BCD number represented  by
                                 ;* 3 bytes (fBCD2:fBCD1:fBCD0) to a 16-bit number (tbinH:tbinL).
                                 ;* MSD of the 5-digit number must be placed in the lowermost nibble  of fBCD2.
                                 ;*
                                 ;* Let "abcde" denote the 5-digit number. The conversion is done by
                                 ;* computing the formula: 10(10(10(10a+b)+c)+d)+e.
                                 ;* The subroutine "mul10a"/"mul10b" does the multiply-and-add opera tion
                                 ;* which is repeated four times during the computation.
                                 ;*
                                 ;* Number of words  :30
                                 ;* Number of cycles     :108
                                 ;* Low registers used   :4 (copyL,copyH,mp10L/tbinL,mp10H/tbinH)
                                 ;* High registers used  :4 (fBCD0,fBCD1,fBCD2,adder)
                                 ;*
                                 ;****************************************************************** *********

                                 ;***** "mul10a"/"mul10b" Subroutine Register Variables

                                 .def    copyL   =r12        ;temporary register
                                 .def    copyH   =r13        ;temporary register
                                 .def    mp10L   =r14        ;Low byte of number to be multiplied by 10
                                 .def    mp10H   =r15        ;High byte of number to be multiplied by 10
                                 .def    adder    =r19        ;value to add after multiplication

                                 ;***** Code

                                 mul10a:    ;***** multiplies "mp10H:mp10L" with 10 and adds "adder" high nibble

0000b1 9532                         swap    adder
                                 mul10b:    ;***** multiplies "mp10H:mp10L" with 10 and adds "adder" low nibble
0000b2 2cce                         mov copyL,mp10L ;make copy
0000b3 2cdf                         mov copyH,mp10H
0000b4 0cee                         lsl mp10L       ;multiply original by 2
0000b5 1cff                         rol mp10H
0000b6 0ccc                         lsl copyL       ;multiply copy by 2
0000b7 1cdd                         rol copyH
0000b8 0ccc                         lsl copyL       ;multiply copy by 2 (4)
0000b9 1cdd                         rol copyH
0000ba 0ccc                         lsl copyL       ;multiply copy by 2 (8)
0000bb 1cdd                         rol copyH
0000bc 0cec                         add mp10L,copyL ;add copy to original
0000bd 1cfd                         adc mp10H,copyH
0000be 703f                         andi    adder,0x0f   ;mask away upper nibble of adder
0000bf 0ee3                         add mp10L,adder ;add lower nibble of adder
0000c0 f408                         brcc    m10_1       ;if carry not cleared
0000c1 94f3                         inc mp10H       ;    inc high byte
0000c2 9508                      m10_1: ret

                                 ;***** Main Routine Register Variables

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

                                 ;***** Code
```

```

                                    BCD2bin16:
0000c3 702f                            andi    fBCD2,0x0f  ;mask away upper nibble of fBCD2
0000c4 24ff                            clr mp10H
0000c5 2ee2                            mov mp10L,fBCD2 ;mp10H:mp10L = a
0000c6 2f31                            mov adder,fBCD1
0000c7 dfe9                            rcall   mul10a      ;mp10H:mp10L = 10a+b
0000c8 2f31                            mov adder,fBCD1
0000c9 dfe8                            rcall   mul10b      ;mp10H:mp10L = 10(10a+b)+c
0000ca 2f30                            mov adder,fBCD0
0000cb dfe5                            rcall   mul10a      ;mp10H:mp10L = 10(10(10a+b)+c)+d
0000cc 2f30                            mov adder,fBCD0
0000cd dfe4                            rcall   mul10b      ;mp10H:mp10L = 10(10(10(10a+b)+c)+d)+e
0000ce 9508                            ret


                                    ;***************************************************************** *********
                                    ;*
                                    ;* "keypress_ISR" - Check Interrupts at INT0
                                    ;*
                                    ;* Description: Get the keyvalue if the key is pressed, the keyvalu e is stored if the key is a number
                                    ;*
                                    ;* Author:                 Seyi Olajuyi & Bassel El Amine
                                    ;* Version:
                                    ;* Last updated:           11/21/19
                                    ;* Target:                 ATmega324A
                                    ;* Number of words:
                                    ;* Number of cycles:       N/A
                                    ;* Low registers modified:  none
                                    ;* High registers modified: none
                                    ;*
                                    ;* Parameters:
                                    ;* Notes:
                                    ;*
```

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

                                     ;INT0 interrupt service routine
                                 keypress_ISR:
0000cf 932f                         push r18
0000d0 930f                          push r16           ;save r16
0000d1 b70f                         in r16, SREG       ;save SREG
0000d2 930f                          push r16

0000d3 e001                         ldi r16 ,1                      ; Set polling_for_keypad
0000d4 9300 0105                    sts polling_for_keypad, r16     ; Use to find out if the keypad was pressed

0000d6 dfc5                         rcall get_key_value
0000d7 9320 0104                    sts keyvalue, r18
0000d9 302a                         cpi r18, $0A                            ; if key value is not a number, end the subroutine.
0000da f028                         brlo skip_line_1

                                 restore_values_1:
0000db 910f                          pop r16            ;restore SREG
0000dc bf0f                          out SREG,r16
0000dd 910f                         pop r16            ;restore r16
0000de 912f                         pop r18            ;restore r18

0000df 9518                          reti               ;return from interrupt

                                 skip_line_1:
0000e0 dfb0                         rcall store_value
0000e1 cfed                         rjmp keypress_ISR



                                 ;****************************************************************** *********
                                 ;*
                                 ;* "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:
0000e2 930f                         push r16            ;save r16
0000e3 b70f                         in r16, SREG        ;save SREG
0000e4 930f                          push r16

0000e5 9904                         sbic PINA, 4
0000e6 cffb                         rjmp wait_for_bounce_1
0000e7 e604                         ldi r16, 100
0000e8 dfc2                         rcall var_delay
0000e9 9904                         sbic PINA, 4
0000ea cff7                         rjmp wait_for_bounce_1

0000eb e002                         ldi r16, (1 <<INTF1)
0000ec bb0c                         out EIFR, r16
```

```

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

                                 restore_value_2:
0000f0 910f                          pop r16             ;restore SREG
0000f1 bf0f                          out SREG,r16
0000f2 910f                         pop r16             ;restore r16

0000f3 9518                          reti                ;return from interrupt




0000f4 0201
0000f5 0f03
0000f6 0504
0000f7 0e06
0000f8 0807
0000f9 0d09                      keytable: .db $01, $02, $03, $0F, $04, $05, $06, $0E, $07, $08, $09 , $0D
0000fa 000a
0000fb 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
'lpm' instruction without operands implicitly uses r0 and z,
none of which are counted.
```

```

x,y,z are separate entities in the symbol table and are
counted separately from r26..r31 here.

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

"ATmega324A" register use summary:
x  :   0 y  :   7 z  :   5 r0 :   0 r1 :   0 r2 :   0 r3 :   0 r4 :   0
r5 :   0 r6 :   0 r7 :   0 r8 :   0 r9 :   0 r10:   0 r11:   0 r12:   5
r13:   5 r14:   6 r15:   5 r16: 103 r17:  23 r18:  20 r19:  12 r20:  12
r21:   2 r22:   2 r23:   2 r24:   4 r25:   2 r26:   2 r27:   0 r28:   2
r29:   2 r30:   6 r31:   6
Registers used: 21 out of 35 (60.0%)

"ATmega324A" instruction use summary:
.lds  :   0 .sts  :   0 adc   :   1 add   :   4 adiw  :   0 and   :   0
andi  :   4 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
brcc  :   1 brcs  :   0 break :   0 breq  :   6 brge  :   0 brhc  :   0
brhs  :   0 brid  :   0 brie  :   0 brlo  :   1 brlt  :   0 brmi  :   0
brne  :  10 brpl  :   0 brsh  :   0 brtc  :   0 brts  :   0 brvc  :   0
brvs  :   0 bset  :   0 bst   :   0 call  :   3 cbi   :   6 cbr   :   0
clc   :   0 clh   :   0 cli   :   0 cln   :   0 clr   :   1 cls   :   0
clt   :   0 clv   :   0 clz   :   0 com   :   0 cp    :   0 cpc   :   0
cpi   :   7 cpse  :   0 dec   :  10 eor   :   0 fmul  :   0 fmuls :   0
fmulsu:   0 icall :   0 ijmp  :   0 in    :  12 inc   :   1 jmp   :   0
ld    :   3 ldd   :   0 ldi   :  59 lds   :  15 lpm   :   2 lsl   :   4
lsr   :   0 mov   :   9 movw  :   0 mul   :   0 muls  :   0 mulsu :   0
neg   :   0 nop   :   2 or    :   4 ori   :   0 out   :  11 pop   :  11
push  :  11 rcall :  50 ret   :  16 reti  :   2 rjmp  :  13 rol   :   4
ror   :   0 sbc   :   0 sbci  :   0 sbi   :  11 sbic  :   2 sbis  :   0
sbiw  :   0 sbr   :   0 sbrc  :   0 sbrs  :   3 sec   :   0 seh   :   0
sei   :   1 sen   :   0 ser   :   0 ses   :   0 set   :   0 sev   :   0
sez   :   0 sleep :   0 spm   :   0 st    :   8 std   :   0 sts   :  17
sub   :   0 subi  :   0 swap  :   3 tst   :   0 wdr   :   0
Instructions used: 37 out of 113 (32.7%)
```

```

"ATmega324A" memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
---------------------------------------------------------------
[.cseg] 0x000000 0x0002e8    724     16    740   32768   2.3%
[.dseg] 0x000100 0x000137      0     55     55    2048   2.7%
[.eseg] 0x000000 0x000000      0      0      0    1024   0.0%

Assembly complete, 0 errors, 2 warnings

```