```

AVRASM ver. 2.2.7  E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_11\temp_meas\temp_meas\main.asm Tue Nov 17 19:25:58 2020

E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_11\temp_meas\temp_meas\main.asm(9): Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.3.300\avrasm\inc\m4809def.inc'
E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_11\temp_meas\temp_meas\main.asm(439): warning: Register r16 already defined by the .DEF directive
E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_11\temp_meas\temp_meas\main.asm(440): warning: Register r17 already defined by the .DEF directive
E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_11\temp_meas\temp_meas\main.asm(485): warning: Register r16 already defined by the .DEF directive
E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_11\temp_meas\temp_meas\main.asm(486): warning: Register r17 already defined by the .DEF directive
E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_11\temp_meas\temp_meas\main.asm(487): warning: Register r18 already defined by the .DEF directive
E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_11\temp_meas\temp_meas\main.asm(488): warning: Register r19 already defined by the .DEF directive
E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_11\temp_meas\temp_meas\main.asm(489): warning: Register r18 already defined by the .DEF directive
E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_11\temp_meas\temp_meas\main.asm(490): warning: Register r19 already defined by the .DEF directive
E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_11\temp_meas\temp_meas\main.asm(491): warning: Register r20 already defined by the .DEF directive
E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_11\temp_meas\temp_meas\main.asm(9): Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.3.300\avrasm\inc\m4809def.inc'


                                 ; temp_meas.asm
                                 ;
                                 ; Created: 11/17/2020 2:02:20 PM
                                 ; Author : Judah Ben-Eliezer
                                 ;

                                 .list

                                 .equ PERIOD_EXAMPLE_VALUE = 25

                                 .dseg
002800                           bcd_entries: .byte 4
002804                           led_display: .byte 4
002808                           digit_num: .byte 1


                                 .cseg

                                 reset:
```

```
000000 940c 002e                    jmp start

                                 .org TCA0_OVF_vect
00000e 940c 0056                    jmp post_display_ISR

                                 .org ADC0_RESRDY_vect
00002c 940c 006a                    jmp read_ISR

                                 start:
                                    ;configure inputs and outputs
00002e 9883                          cbi VPORTE_DIR, 3
00002f ef0f                         ldi r16, $FF
000030 b908                         out VPORTC_DIR, r16
000031 b90c                         out VPORTD_DIR, r16
000032 9500                         com r16
000033 b909                         out VPORTC_OUT, r16
000034 b90d                         out VPORTD_OUT, r16

                                    ;configure TCA0
000035 e000                         ldi r16, TCA_SINGLE_WGMODE_NORMAL_gc ;WGMODE normal
000036 9300 0a01                    sts TCA0_SINGLE_CTRLB, r16

                                    ;enable overflow interrupt
000038 e001                         ldi r16, TCA_SINGLE_OVF_bm
000039 9300 0a0a                    sts TCA0_SINGLE_INTCTRL, r16

                                    ;load period low byte then high byte
00003b e109                         ldi r16, LOW(PERIOD_EXAMPLE_VALUE)
00003c 9300 0a26                    sts TCA0_SINGLE_PER, r16
00003e e000                         ldi r16, HIGH(PERIOD_EXAMPLE_VALUE)
00003f 9300 0a27                    sts TCA0_SINGLE_PER + 1, r16

                                    ;set clock and start timer
000041 e00d                         ldi r16, TCA_SINGLE_CLKSEL_DIV256_gc | TCA_SINGLE_ENABLE_bm
000042 9300 0a00                    sts TCA0_SINGLE_CTRLA, r16

                                    ;set voltage reference
000044 e200                         ldi r16, VREF_ADC0REFSEL_2V5_gc
000045 9300 00a0                    sts VREF_CTRLA, r16

                                    ;select PE1/ AIN9
000047 e00b                         ldi r16, ADC_MUXPOS_AIN11_gc
000048 9300 0606                    sts ADC0_MUXPOS, r16

                                    ;enable internal reference and set prescaler to div 64
00004a e005                         ldi r16, ADC_PRESC_DIV64_gc |
    ADC_REFSEL_INTREF_gc
00004b 9300 0602                    sts ADC0_CTRLC, r16

                                    ;set resolution to 10 bit and enable adc
00004d e001                         ldi r16, ADC_RESSEL_10BIT_gc | ADC_ENABLE_bm;
00004e 9300 0600                    sts ADC0_CTRLA, r16

                                    ;start conversion
000050 e001                         ldi r16, ADC_STCONV_bm;
000051 9300 0608                    sts ADC0_COMMAND, r16

                                    ;enable interrupts
000053 9478                         sei
000054 940c 0065                    jmp wait_for_post

                                ;***************************************************************************
                                ;*
                                ;* "post_display" - title
                                ;*
                                ;* Description: toggles value for all PORTC pins.  Since PORTC is used to multiplex the led display, this will
                                ;*  turn the LED display on and off
                                ;* Author: Judah Ben-Eliezer
                                ;* Version:    1.0
                                ;* Last updated:   11/17
                                ;* Target: ATmega4809
                                ;* Number of words:    13
                                ;* Number of cycles:   6
                                ;* Low registers modified:
                                ;* High registers modified:
                                ;* Parameters: none
                                ;* Returns:        none
                                ;*
                                ;* Notes:
                                ;*
                                ;***************************************************************************
                                post_display_ISR:
000056 930f                         push r16
000057 b70f                         in r16, CPU_SREG
000058 930f                         push r16
000059 931f                         push r17

00005a ef1f                         ldi r17, $FF
00005b b109                         in r16, VPORTC_OUT
00005c 2701                         eor r16, r17
```

```
00005d b909                         out VPORTC_OUT, r16

                                    ;ldi r16, TCA_SINGLE_OVF_bm ;clear OVF flag
                                    ;sts TCA0_SINGLE_INTFLAGS, r16

00005e 911f                         pop r17
00005f 910f                         pop r16
000060 bf0f                         out CPU_SREG, r16
000061 910f                         pop r16

000062 9478                         sei
000063 940c 0067                    jmp main_loop

                                 wait_for_post:
000065 0000                         nop
000066 cffe                         rjmp wait_for_post

                                 main_loop:
000067 d037                         rcall multiplex_display
000068 d04b                         rcall mux_digit_delay
000069 cffd                         rjmp main_loop

                                 ;***************************************************************************
                                 ;*
                                 ;* "read_ISR" - title
                                 ;*
                                 ;* Description: loads ADC0_RES into r17:r16    and calls bin16_to_BCD
                                 ;*
                                 ;* Author: Judah Ben-Eliezer
                                 ;* Version:    1.0
                                 ;* Last updated:    11/17/2020
                                 ;* Target: ATmega4809
                                 ;* Number of words:
                                 ;* Number of cycles:
                                 ;* Low registers modified: none
                                 ;* High registers modified:        r17:r16
                                 ;*
                                 ;* Parameters: ADC0_RES
                                 ;* Returns:        r17:r16
                                 ;*
                                 ;* Notes:
                                 ;*
                                 ;*********************************************************************************
                                 read_ISR:
00006a 930f                         push r16
00006b b70f                         in r16, CPU_SREG
```

```
00006c 930f                        push r16
00006d 2f0b                        mov r16, XH
00006e 930f                        push r16
00006f 2f0a                        mov r16, XL
000070 930f                        push r16
000071 2f0f                        mov r16, ZH
000072 930f                        push r16
000073 2f0e                        mov r16, ZL
000074 930f                        push r16
000075 931f                        push r17
000076 932f                        push r18
000077 933f                        push r19

000078 e039                        ldi r19, $09    ;load multiplier 2500 into r19:r18 for multiplication
000079 ec24                        ldi r18, $C4
00007a 9110 0611                   lds r17, ADC0_RESH
00007c 9100 0610                   lds r16, ADC0_RESL
00007e d07f                        rcall mpy16u
00007f 9526                        lsr r18     ;finish division by 1024 by taking middle two bytes and shifting right twice
000080 9517                        ror r17
000081 9526                        lsr r18
000082 9517                        ror r17
000083 2f01                        mov r16, r17    ; move output to r17:r16
000084 2f12                        mov r17, r18
000085 5f04                        subi r16, $F4   ;subtract 500 from result
000086 4011                        sbci r17, $01
000087 d054                        rcall bin16_to_BCD
000088 d032                        rcall packed_to_bcd_entries
000089 d041                        rcall bcd_to_led

                                   ;reset interrupt flag
00008a e001                        ldi r16, ADC_RESRDY_bm;
00008b 9300 060a                   sts ADC0_INTCTRL, r16

                                   ;restart conversion
00008d e001                        ldi r16, ADC_STCONV_bm;
00008e 9300 0608                   sts ADC0_COMMAND, r16

000090 913f                        pop r19
000091 912f                        pop r18
000092 911f                        pop r17
000093 910f                        pop r16
000094 2fe0                        mov ZL, r16
000095 910f                        pop r16
000096 2ff0                        mov ZH, r16
000097 910f                        pop r16
000098 2fa0                        mov XL, r16
```

```
000099 910f                 pop r16
00009a 2fb0                 mov XH, r16
00009b 910f                 pop r16
00009c bf0f                 out CPU_SREG, r16
00009d 910f                 pop r16

00009e 9518                 reti

                         ;***************************************************************************
                         ;*
                         ;* "multiplex_display" - title
                         ;*
                         ;* Description:    outputs values from led_display array to 7 segment display on PORTD driven by highest two bits of PORTC
                         ;*
                         ;* Author:  Judah Ben-Eliezer
                         ;* Version:    1.0
                         ;* Last updated:    11/10/2020
                         ;* Target:  ATmega4809
                         ;* Number of words:
                         ;* Number of cycles:
                         ;* Low registers modified:
                         ;* High registers modified:
                         ;*
                         ;* Parameters:
                         ;* Returns:
                         ;*
                         ;* Notes:
                         ;*
                         ;***************************************************************************
                         multiplex_display:
00009f e2d8                 ldi YH, HIGH(led_display)
0000a0 e0c4                 ldi YL, LOW(led_display)
0000a1 9110 2808            lds r17, digit_num
0000a3 7013                 andi r17, $03
0000a4 2f41                 mov r20, r17
0000a5 0fc1                 add YL, r17
0000a6 8128                 ld r18, Y
0000a7 e850                 ldi r21, $80
0000a8 9543                 inc r20
                         loop:
0000a9 9556                 lsr r21
0000aa 954a                 dec r20
0000ab f7e9                 brne loop
0000ac 0f55                 lsl r21
0000ad 9550                 com r21
```

```
0000ae b959                  out VPORTC_OUT, r21
0000af b92d                  out VPORTD_OUT, r18
0000b0 9513                  inc r17
0000b1 9310 2808             sts digit_num, r17
0000b3 9508                  ret

                          ;****************************************************************************
                          ;*
                          ;* "mux_digit_delay" - title
                          ;*
                          ;* Description: delays 0.1 * r23
                          ;*
                          ;* Author: Judah Ben-Eliezer
                          ;* Version:    1.0
                          ;* Last updated:
                          ;* Target:
                          ;* Number of words:
                          ;* Number of cycles:
                          ;* Low registers modified:
                          ;* High registers modified:
                          ;*
                          ;* Parameters:
                          ;* Returns:
                          ;*
                          ;* Notes:
                          ;*
                          ;****************************************************************************
                          mux_digit_delay:
0000b4 e078                  ldi r23, $08 ; 0.1 * r23 = delay
                          outer_loop:
0000b5 e086                  ldi r24, $06
                          inner_loop:
0000b6 958a                  dec r24
0000b7 f7f1                  brne inner_loop
0000b8 957a                  dec r23
0000b9 f7d9                  brne outer_loop
0000ba 9508                  ret

                          ;****************************************************************************
                          ;*
                          ;* "packed_to_bcd_entries" - title
                          ;*
                          ;* Description:    Converts bcd input to 7seg output, from bcd_entries array to led_display array
                          ;*
                          ;* Author:
```

```
                         ;* Version:
                         ;* Last updated:
                         ;* Target:
                         ;* Number of words:
                         ;* Number of cycles:
                         ;* Low registers modified:
                         ;* High registers modified:
                         ;*
                         ;* Parameters:
                         ;* Returns:
                         ;*
                         ;* Notes:
                         ;*
                         ;*****************************************************************************
                         packed_to_bcd_entries:
0000bb e2b8                 ldi XH, HIGH(bcd_entries)
0000bc e0a0                 ldi XL, LOW(bcd_entries)
0000bd 2f37                 mov r19, r23
0000be 7f30                 andi r19, $F0
0000bf 9532                 swap r19
0000c0 933d                 st X+, r19
0000c1 2f37                 mov r19, r23
0000c2 703f                 andi r19, $0F
0000c3 933d                 st X+, r19
0000c4 2f36                 mov r19, r22
0000c5 7f30                 andi r19, $F0
0000c6 9532                 swap r19
0000c7 933d                 st X+, r19
0000c8 2f36                 mov r19, r22
0000c9 703f                 andi r19, $0F
0000ca 933c                 st X, r19


                         ;*****************************************************************************
                         ;*
                         ;* "bcd_to_led" - title
                         ;*
                         ;* Description:    Converts bcd input to 7seg output, from bcd_entries array to led_display array
                         ;*
                         ;* Author:
                         ;* Version:
                         ;* Last updated:
                         ;* Target:
                         ;* Number of words:
                         ;* Number of cycles:
                         ;* Low registers modified:
```

```
                        ;* High registers modified:
                        ;*
                        ;* Parameters:
                        ;* Returns:
                        ;*
                        ;* Notes:
                        ;*
                        ;***************************************************************************

                        bcd_to_led:
0000cb e2b8                ldi XH, HIGH(bcd_entries)
0000cc e0a0                ldi XL, LOW(bcd_entries)
0000cd 931c                st X, r17
0000ce e044                ldi r20, $04
                        conversion_loop:
0000cf 954a                dec r20
0000d0 e2b8                ldi XH, HIGH(bcd_entries)
0000d1 e0a0                ldi XL, LOW(bcd_entries)
0000d2 e2d8                ldi YH, HIGH(led_display)
0000d3 e0c4                ldi YL, LOW(led_display)
0000d4 0fa4                add XL, r20
0000d5 0fc4                add YL, r20
0000d6 912c                ld r18, X
0000d7 d035                rcall hex_to_7seg
0000d8 8328                st Y, r18
0000d9 3040                cpi r20, $00
0000da f7a1                brne conversion_loop
0000db 9508                ret

                        ;***************************************************************************
                        ;*
                        ;* "bin16_to_BCD" - 16-bit Binary to BCD Conversion
                        ;*
                        ;* Description: Converts a 16-bit unsigned binary number to a five digit
                        ;* packed BCD number. Uses subroutine div16u from Atmel application note AVR200
                        ;*
                        ;* Author:                 Ken Short
                        ;* Version:                    0.0
                        ;* Last updated:           111320
                        ;* Target:                 ATmega4809
                        ;* Number of words:
                        ;* Number of cycles:
                        ;* Low registers modified:  r14, r15
                        ;* High registers modified: r16, r17, r18,
                    r19, r20, r22, r23, r24
                    ;*
                    ;* Parameters: r17:r16 16-bit unsigned right justified number to be converted.
                    ;* Returns:        r24:r23:r22 five digit packed BCD result.
                    ;*
                    ;* Notes:
                    ;* Subroutine uses repeated division by 10 to perform conversion.
                    ;***************************************************************************
                    bin16_to_BCD:
0000dc e030            ldi r19, 0          ;high byte of divisor for div16u
0000dd e02a            ldi r18, 10         ;low byte of the divisor for div16u

0000de d00c            rcall div16u        ;divide original binary number by 10
0000df 2d6e            mov r22, r14        ;result is BCD digit 0 (least significant digit)
0000e0 d00a            rcall div16u        ;divide result from first division by 10, gives digit 1
0000e1 94e2            swap r14            ;swap digit 1 for packing
0000e2 296e            or r22, r14         ;pack

0000e3 d007            rcall div16u        ;divide result from second division by 10, gives digit 2
0000e4 2d7e            mov r23, r14        ;place in r23
0000e5 d005            rcall div16u        ;divide result from third division by 10, gives digit 3
0000e6 94e2            swap r14            ;swap digit 3 for packing
0000e7 297e            or r23, r14         ;pack

0000e8 d002            rcall div16u        ;divide result from fourth division by 10, gives digit 4
0000e9 2d8e            mov r24, r14        ;place in r24

0000ea 9508            ret


                    ;Subroutine div16u is from Atmel application note AVR200

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

```
                          ;*
                          ;* "div16u" - 16/16 Bit Unsigned Division
                          ;*
                          ;* This subroutine divides the two 16-bit numbers
                          ;*# "dd16uH:dd16uL" (dividend) and "dv16uH:dv16uL" (divisor).
                          ;* The result is placed in "dres16uH:dres16uL" and the remainder in
                          ;* "drem16uH:drem16uL".
                          ;*
                          ;* Number of words  :19
                          ;* Number of cycles     :235/251 (Min/Max)
                          ;* Low registers used   :2 (drem16uL,drem16uH)
                          ;* High registers used  :5 (dres16uL/dd16uL,dres16uH/dd16uH,dv16uL,dv16uH,
                          ;*            dcnt16u)
                          ;*
                          ;****************************************************************************

                          ;***** Subroutine Register Variables

                          .def   drem16uL=r14
                          .def   drem16uH=r15
                          .def   dres16uL=r16
                          .def   dres16uH=r17
                          .def   dd16uL  =r16
                          .def   dd16uH  =r17
                          .def   dv16uL  =r18
                          .def   dv16uH  =r19
                          .def   dcnt16u =r20

                          ;***** Code

0000eb 24ee               div16u:    clr drem16uL    ;clear remainder Low byte
0000ec 18ff                   sub drem16uH,drem16uH;clear remainder High byte and carry
0000ed e141                   ldi dcnt16u,17   ;init loop counter
0000ee 1f00               d16u_1:    rol dd16uL      ;shift left dividend
0000ef 1f11                   rol dd16uH
0000f0 954a                   dec dcnt16u      ;decrement counter
0000f1 f409                   brne    d16u_2      ;if done
0000f2 9508                   ret         ;    return
0000f3 1cee               d16u_2:    rol drem16uL    ;shift dividend into remainder
0000f4 1cff                   rol drem16uH
```

```
0000f5 1ae2                   sub drem16uL,dv16uL ;remainder = remainder
  - divisor
0000f6 0af3                   sbc drem16uH,dv16uH ;
0000f7 f420                   brcc    d16u_3      ;if result negative
0000f8 0ee2                   add drem16uL,dv16uL ;    restore remainder
0000f9 1ef3                   adc drem16uH,dv16uH
0000fa 9488                   clc         ;    clear carry to be shifted
  into result
0000fb cff2                   rjmp    d16u_1      ;else
0000fc 9408                d16u_3:    sec         ;    set carry to be
  shifted into result
0000fd cff0                   rjmp    d16u_1

                           ;*********************************************
             ******************************
                           ;*
                           ;* "mpy16u" - 16x16 Bit Unsigned
             Multiplication
                           ;*
                           ;* This subroutine multiplies the two 16-bit
             register variables
                           ;* mp16uH:mp16uL and mc16uH:mc16uL.
                           ;* The result is placed in
             m16u3:m16u2:m16u1:m16u0.
                           ;*
                           ;* Number of words  :14 + return
                           ;* Number of cycles    :153 + return
                           ;* Low registers used   :None
                           ;* High registers used  :7
             (mp16uL,mp16uH,mc16uL/m16u0,mc16uH/m16u1,m16u2,
                           ;*                          m16u3,mcnt16u)
                           ;*
                           ;*********************************************
             ******************************

                           ;***** Subroutine Register Variables

                           .def    mc16uL   =r16        ;multiplicand low
             byte
                           .def    mc16uH   =r17        ;multiplicand high
             byte
                           .def    mp16uL   =r18        ;multiplier low
             byte
                           .def    mp16uH   =r19        ;multiplier high
             byte
                           .def    m16u0    =r18        ;result byte 0
             (LSB)
                           .def    m16u1    =r19        ;result byte 1
                           .def    m16u2    =r20        ;result byte 2
```

```
                        .def    m16u3   =r21        ;result byte 3 (MSB)
                        .def    mcnt16u =r22        ;loop counter

                        ;***** Code

0000fe 2755             mpy16u:     clr m16u3       ;clear 2 highest bytes of result
0000ff 2744                 clr m16u2
000100 e160                 ldi mcnt16u,16  ;init loop counter
000101 9536                 lsr mp16uH
000102 9527                 ror mp16uL

000103 f410             m16u_1:     brcc    noad8       ;if bit 0 of multiplier set
000104 0f40                 add m16u2,mc16uL    ;add multiplicand Low to byte 2 of res
000105 1f51                 adc m16u3,mc16uH    ;add multiplicand high to byte 3 of res
000106 9557             noad8:  ror m16u3       ;shift right result byte 3
000107 9547                 ror m16u2       ;rotate right result byte 2
000108 9537                 ror m16u1       ;rotate result byte 1 and multiplier High
000109 9527                 ror m16u0       ;rotate result byte 0 and multiplier Low
00010a 956a                 dec mcnt16u     ;decrement loop counter
00010b f7b9                 brne    m16u_1      ;if not done, loop more
00010c 9508                 ret




                        ;****************************************************************************
                        ;*
                        ;* "hex_to_7seg" - Hexadecimal to Seven Segment Conversion
                        ;*
                        ;* Description: Converts a right justified hexadecimal digit to the seven
                        ;* segment pattern required to display it. Pattern is right justified a
                        ;* through g. Pattern uses 0s to turn segments on ON.
                        ;*
                        ;* Author:                      Ken Short
                        ;* Version:                         1.0
```

```
                                  ;* Last updated:              101620
                                  ;* Target:                    ATmega4809
                                  ;* Number of words:               8
                                  ;* Number of cycles:          13
                                  ;* Low registers modified:    none
                                  ;* High registers modified:       r19, r18, ZL, ZH
                                  ;*
                                  ;* Parameters: r18: right justified hex digit, high nibble 0
                                  ;* Returns: r18: segment values a through g right justified
                                  ;*
                                  ;* Notes:
                                  ;*
                                  ;***********************************************************************************

                                  hex_to_7seg:
00010d 702f                          andi r18, 0x0F              ;clear ms nibble
00010e e0f2                           ldi ZH, HIGH(hextable * 2)  ;set Z to point to start of table
00010f e2ea                           ldi ZL, LOW(hextable * 2)
000110 e030                           ldi r19, $00                ;add offset to Z pointer
000111 0fe2                           add ZL, r18
000112 1ff3                           adc ZH, r19
000113 9124                           lpm r18, Z                  ;load byte from table pointed to by Z
000114 9508                          ret

                                      ;Table of segment values to display digits 0 - F
                                      ;!!! seven values must be added - verify all values
000115 4f01
000116 0612
000117 244c
000118 0f20
000119 0400
00011a 6008
00011b 4231
00011c 3830                       hextable: .db $01, $4F, $12, $06, $4C, $24, $20, $0F, $00, $04, $08, $60, $31, $42, $30, $38


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

"ATmega4809" register use summary:
x  :   6 y  :   2 z  :   1 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:   0
r13:   0 r14:  11 r15:   5 r16:  66 r17:  21 r18:  18 r19:  24 r20:  13
r21:   8 r22:   6 r23:   6 r24:   3 r25:   0 r26:   6 r27:   5 r28:   4
r29:   2 r30:   4 r31:   4
Registers used: 20 out of 35 (57.1%)

"ATmega4809" instruction use summary:
.lds  :   0 .sts  :   0 adc   :   3 add   :   6 adiw  :   0 and   :   0
andi  :   6 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
brcc  :   2 brcs  :   0 break :   0 breq  :   0 brge  :   0 brhc  :   0
brhs  :   0 brid  :   0 brie  :   0 brlo  :   0 brlt  :   0 brmi  :   0
brne  :   6 brpl  :   0 brsh  :   0 brtc  :   0 brts  :   0 brvc  :   0
brvs  :   0 bset  :   0 bst   :   0 call  :   0 cbi   :   1 cbr   :   0
clc   :   1 clh   :   0 cli   :   0 cln   :   0 clr   :   3 cls   :   0
clt   :   0 clv   :   0 clz   :   0 com   :   2 cp    :   0 cpc   :   0
cpi   :   1 cpse  :   0 dec   :   6 des   :   0 eor   :   1 fmul  :   0
fmuls :   0 fmulsu:   0 icall :   0 ijmp  :   0 in    :   3 inc   :   2
jmp   :   5 ld    :   2 ldd   :   0 ldi   :  37 lds   :   3 lpm   :   2
lsl   :   1 lsr   :   4 mov   :  18 movw  :   0 mul   :   0 muls  :   0
mulsu :   0 neg   :   0 nop   :   1 or    :   2 ori   :   0 out   :   9
pop   :  12 push  :  12 rcall :  12 ret   :   7 reti  :   1 rjmp  :   4
rol   :   4 ror   :   7 sbc   :   1 sbci  :   1 sbi   :   0 sbic  :   0
sbis  :   0 sbiw  :   0 sbr   :   0 sbrc  :   0 sbrs  :   0 sec   :   1
seh   :   0 sei   :   2 sen   :   0 ser   :   0 ses   :   0 set   :   0
sev   :   0 sez   :   0 sleep :   0 spm   :   0 st    :   6 std   :   0
sts   :  13 sub   :   2 subi  :   1 swap  :   4 tst   :   0 wdr   :   0

Instructions used: 42 out of 114 (36.8%)

"ATmega4809" memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
---------------------------------------------------------------
[.cseg] 0x000000 0x00023a    474     16    490   49152   1.0%
[.dseg] 0x002800 0x002809      0      9      9    6144   0.1%
[.eseg] 0x000000 0x000000      0      0      0     256   0.0%
```

```

Assembly complete, 0 errors, 9 warnings

```