Prelab 5

Author: Rajith Radhakrishnan

ID: 109061463
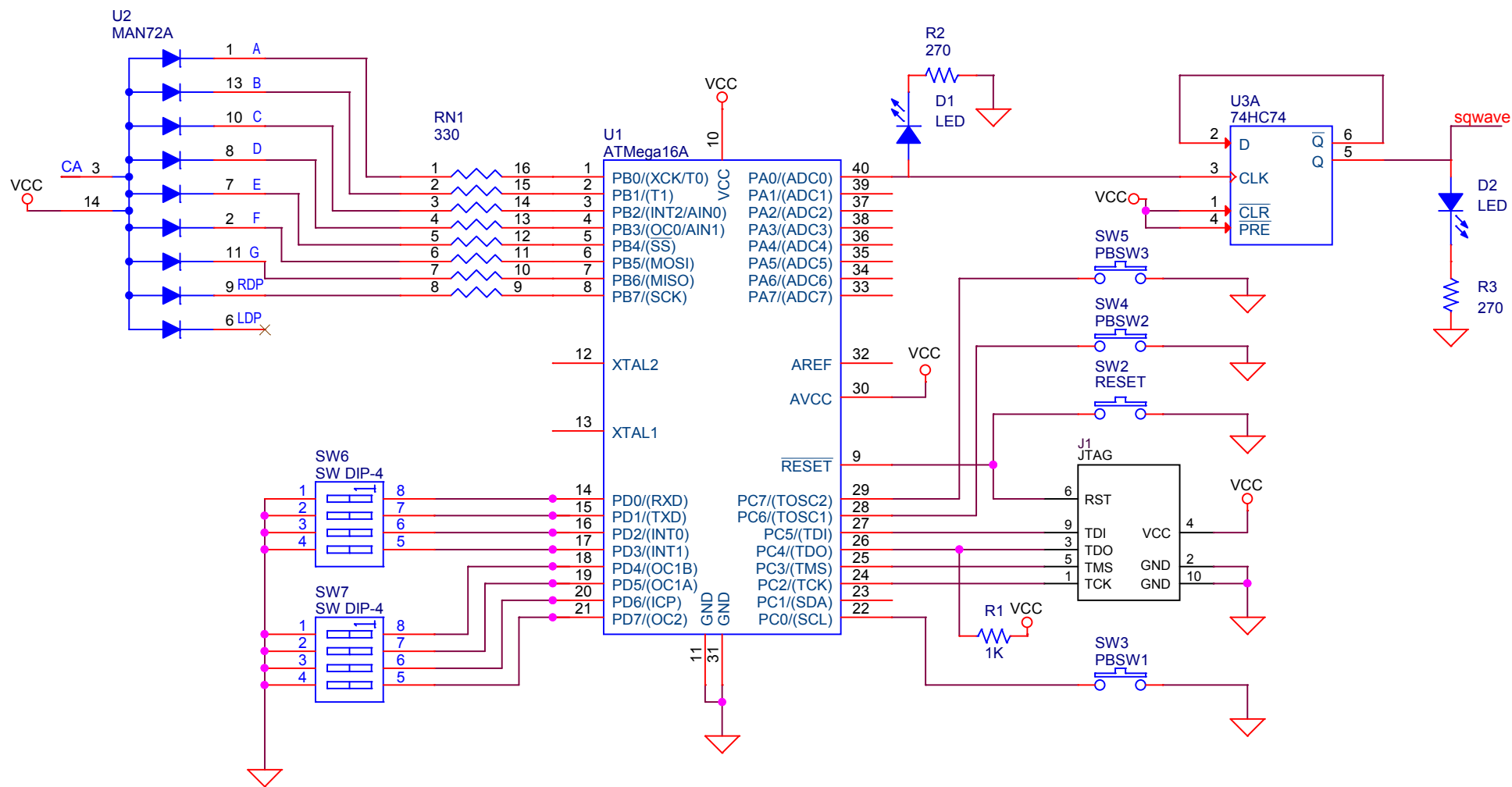
Partner's Name: Raymond Ng

ID: 109223276

Course and section: ESE 380 L01

Bench Number: 6

Due Date/time : 10/06/14 9:00 PM

Date of the Lab: 10/07/14

Title: Lab3 - task 4

Size: A
Document Number: Ng, Radhakrishnan
Rev: 1V

Date: Tuesday, October 07, 2014
Sheet 1 of 1

AVRASM ver. 2.1.52 C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014
notes and files\ese 380 lab\lab 5\duty_cycle\duty_cycle\duty_cycle.asm Tue Oct 07
19:21:32 2014

C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files\ese 380
lab\lab 5\duty_cycle\duty_cycle\duty_cycle.asm(22): Including file 'C:\Program Files (x86)
\Atmel\Atmel Toolchain\AVR Assembler\Native\2.1.39.1005\avrassembler\Include\m16def.inc'

```
                      * duty_cycle.asm
                      ; components used:
                      ; 4 bit nibble DIP switches(2), 7 seg dispaly, 3 leds with 270 ohm resisotr
                      ; 3 pushbuttons
                      ;
                      * This program will read the lower DIP switches and output a PWM signal depending
                      ; on the BCD number entered using the switch. The MSB will be on the top and LSB
                      ; on the bottom of the switch. The first pushbutton on top will be utilized for
                      ; as the load button and led as output, and the duty cycle will be displayed on
                      ; the 7seg.
                      ; output 0 for all the values above 9
                      ;
                      * inputs used: PD4-7(switches), PC 7(PBSW)
                      * outputs used: PB 0-7(7seg), PA0 (LED)
                      *
                      ;register r20 and 21 used for delay timers
                      ;r18 - to store switch values
                      *    Author: radra_000
                      */
                      .list

                      ;equates for delay loop countes
                      .equ outer = 100                ;delays 241 clk cyles
                      .equ inner = 33                 ;delays 13 clk cycles

                  //loads the registers and ports required, run and repated when powere up or reset
                   reset:
                      //initizling the stack pointer
000000 e50f          ldi r16, LOW(RAMEND)            ;load SPL with low byte of
000001 bf0d          out SPL, r16                    ;RAMEND adress
000002 e004          ldi r16, HIGH(RAMEND)           ;load SPH with low byte of
000003 bf0e          out SPH, r16                    ;RAMEND adress

000004 e000          LDI r16, $00                    ;load register 16 with 1's
                  // OUT PORTB, r16                   ;load the 7seg diplay
000005 bb07          OUT DDRB, r16                   ;turn on all leds in 7seg
000006 e001          ldi r16, 1                      ;load r16 with 1
000007 bb0a          OUT DDRA, r16                   ;set up led as output
000008 e70f          ldi r16, 0b01111111             ;set only the first switch
000009 bb04          out DDRC, r16                   ;set the pc7 as the input
                  // ldi r16, 0b11111111
                  // out DDRD, r16                    ;set the lower DIP Switch
00000a ef0f          LDI r16, $ff                    ;load register 16 with 0's
00000b bb02          OUT PORTD, r16                  ;load the pull ups for dip switch
00000c bb05          OUT PORTC, r16                  ;load the pull ups for PBSW
00000d bb08          OUT PORTB, r16                  ;output 0 on 7seg

                  //check if the load button is pressed

                  main_loop:
00000e e034          ldi r19, 4                      ;r19 used as counter
00000f 999f          SBIC PINC, 7                    ;check if the PB switch is pressed
000010 cffd          rjmp main_loop                  ;repeat until button is pressed
000011 d029          rcall delay                     ;wait for 10ms(debounce)
000012 9b9f          SBIS PINC, 7                    ;check if PB is still pressed
000013 c001          rjmp read_switch                ; jump to read switch
```

```
000014 cff9        rjmp main_loop                 ;repeat the main loop

                   //read values of switch
                   read_switch:
000015 b320          in r18, PIND                 ;input the switch values to r18
                   // ldi r16, 0                  ;load with 0 to compare with swtich
000016 3020          CPI r18, 0                   ;check weather r18 is 0
000017 f0d1          BREQ clear                   ;jump to clear if equal
000018 ef06          ldi r16, $f6                     ;load r16 with 90 in hex, to check if its 9+
000019 0f02          add r16,r18                  ;if there is a negative value then
00001a f0b8          brcs clear                   ;output 0 for anything greater than 9

00001b d008          rcall hex_7seg               ;go to hex7seg and return
00001c d000          rcall duty_pwm               ;go to dutypwm and return

                   //output the pwm singal
                   duty_pwm:
00001d 999f          SBIC PINC, 7                 ;check if load button is pressed
00001e cfef          rjmp main_loop               ;if pressed restart
00001f 9ad8          SBI PORTA, 0                 ;set port a to high
000020 d013          rcall delay_on               ;turn on the led for cetain period
000021 98d8          CBI PORTA, 0                 ;set port a to low
000022 d014          rcall delay_off              ;turn off the led for certain period
000023 cff9          rjmp duty_pwm                ;output the signal until button press



                   //subroutines
                   //-----------
                   /*
                   //get r18 to 4 digits from 8 digits
                   hex_7seg:
                     lsr r18                      ;shift lsb to carry
                     dec r19                      ;decrement r19 for 4 times
                     BRNE hex_7seg                ;after shifting 4 digits go to bcd_7seg
                     */

                   //display the duty cycle value
                   hex_7seg:
000024 2f12          mov r17, r18                 ;copy r18 to r17
000025 e0f0          ldi ZH, HIGH(table*2)
000026 e5ea          ldi ZL, LOW(table*2)         ;set z to point to start of the table
000027 e000          ldi r16, $00                 ;clear for later use
000028 0fe1          add ZL, r17                  ;add low byte
000029 1ff0          adc ZH, r16                  ;add in the carry
00002a 9114          lpm r17, z                   ;load bid pattern from table into r18
                   display:
00002b bb18          out PORTB,r17                ;output patter for 7 seg display
00002c 9508          ret
00002d 7940
00002e 3024
00002f 1219
000030 7803
000031 1800      table: .db $40, $79, $24, $30, $19, $12, $03, $78,$0, $18
                        //  0    1    2    3    4    5    6    7  8    9


                   //output 0 in the signal
                   clear:
000032 98d8          CBI PORTA, 0                 ;set port a to 0
000033 cfda          rjmp main_loop

                   //turn on the signal for the switch value

                   delay_on:
```

```
000034 e241          ldi r20, 33                    ;load r20 with 100
000035 2f52          mov r21, r18                   ;load r21 with switch values
000036 c006          rjmp inner_loop                ;delay for r18*100 cycles

                     //turn off the signal for 10 minus switch value
                     delay_off:
000037 e241          ldi r20, 33                    ;load r20 with 100
000038 e05a          ldi r21, 10                    ;load r21 with 10
000039 1b52          sub r21, r18                   ;subract 10 - switch values
00003a c002          rjmp inner_loop                ;delay for r21*100 cycles

                     delay:
00003b e241          ldi r20, inner                 ;set r20 to 240
00003c e654          ldi r21, outer                 ;   set r21 to 13
                     inner_loop:
00003d 954a          dec r20                        ;decrements 240
00003e f7f1          brne inner_loop                ;repeat until r20 is 0
                     outer_loop:
00003f e241          ldi r20, inner                 ;reset the r20 to 240
000040 955a          dec r21                        ;decrement for 13 cycles
000041 f7d9          brne inner_loop                ;repeat until r21 is 0
000042 e654          ldi r21, outer                 ;reset r21 for next delay
000043 9508          ret                            ;return
                     /*
                     delay:
                         ldi r20, inner             ;set r20 to 240
                         //ldi r21, outer           ;   set r21 to 13
                     inner_loop:
                         ldi r21, outer
                     outer_loop:
                         dec r21                     ;decrements 240
                         brne outer_loop             ;repeat until r20 is 0
                         dec r20
                         brne inner_loop
                         ret                         ;return

                     */
```

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

"ATmega16" register use summary:
```
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:   0 r15:   0
r16:  18 r17:   4 r18:   6 r19:   1 r20:   5 r21:   6 r22:   0 r23:   0
r24:   0 r25:   0 r26:   0 r27:   0 r28:   0 r29:   0 r30:   2 r31:   2
x  :   0 y  :   0 z  :   1
```
Registers used: 9 out of 35 (25.7%)

"ATmega16" instruction use summary:
```
.lds  :   0 .sts  :   0 adc   :   1 add   :   2 adiw  :   0 and   :   0
andi  :   0 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
brcc  :   0 brcs  :   1 break :   0 breq  :   1 brge  :   0 brhc  :   0
brhs  :   0 brid  :   0 brie  :   0 brlo  :   0 brlt  :   0 brmi  :   0
```

```
brne  :   2 brpl  :   0 brsh  :   0 brtc  :   0 brts  :   0 brvc  :   0
brvs  :   0 bset  :   0 bst   :   0 call  :   0 cbi   :   2 cbr   :   0
clc   :   0 clh   :   0 cli   :   0 cln   :   0 clr   :   0 cls   :   0
clt   :   0 clv   :   0 clz   :   0 com   :   0 cp    :   0 cpc   :   0
cpi   :   1 cpse  :   0 dec   :   2 eor   :   0 fmul  :   0 fmuls :   0
fmulsu:   0 icall :   0 ijmp  :   0 in    :   1 inc   :   0 jmp   :   0
ld    :   0 ldd   :   0 ldi   :  18 lds   :   0 lpm   :   2 lsl   :   0
lsr   :   0 mov   :   2 movw  :   0 mul   :   0 muls  :   0 mulsu :   0
neg   :   0 nop   :   0 or    :   0 ori   :   0 out   :   9 pop   :   0
push  :   0 rcall :   5 ret   :   2 reti  :   0 rjmp  :   8 rol   :   0
ror   :   0 sbc   :   0 sbci  :   0 sbi   :   1 sbic  :   2 sbis  :   1
sbiw  :   0 sbr   :   0 sbrc  :   0 sbrs  :   0 sec   :   0 seh   :   0
sei   :   0 sen   :   0 ser   :   0 ses   :   0 set   :   0 sev   :   0
sez   :   0 sleep :   0 spm   :   0 st    :   0 std   :   0 sts   :   0
sub   :   1 subi  :   0 swap  :   0 tst   :   0 wdr   :   0
Instructions used: 20 out of 113 (17.7%)

"ATmega16" memory use summary [bytes]:
Segment    Begin    End       Code    Data    Used    Size    Use%
-------------------------------------------------------------
[.cseg] 0x000000 0x000088     126      10     136   16384    0.8%
[.dseg] 0x000060 0x000060       0       0       0    1024    0.0%
[.eseg] 0x000000 0x000000       0       0       0     512    0.0%


Assembly complete, 0 errors, 0 warnings
```

AVRASM ver. 2.1.52 C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files
\ese 380 lab\lab 5\dutycycl_freq\dutycycl_freq\dutycycl_freq.asm Sat Oct 04 21:59:12 2014

C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files\ese 380 lab\lab 5\
dutycycl_freq\dutycycl_freq\dutycycl_freq.asm(22): Including file 'C:\Program Files (x86)\Atmel\Atmel
 Toolchain\AVR Assembler\Native\2.1.39.1005\avrassembler\Include\m16def.inc'


```
                        * duty_cycle_freq.asm
                        ; components used:
                        ; 4 bit nibble DIP switches(2), 7 seg dispaly, 3 leds with 270 ohm resisotr
                        ; 3 pushbuttons
                        ;
                        * This program will read the lower DIP switches and output a PWM signal depending
                        ; on the BCD number entered using the switch. The MSB will be on the top and LSB
                        ; on the bottom of the switch. The first pushbutton on top will be utilized for
                        ; as the load button and led as output, and the duty cycle will be displayed on
                        ; the 7seg. The upper DIP Switches will be read and affect the frequency
                        ; output 0 for all the values above 9
                        ;
                        * inputs used: PD4-7(switches), PC 7(PBSW)
                        * outputs used: PB 0-7(7seg), PA0 (LED)
                        *
                        ;register r20 and 21 used for delay timers
                        ;r18 - to store switch values
                        *   Author: radra_000
                        */
                        .list

                        ;equates for delay loop countes
                        .equ outer = $f1                ;delays 241 clk cyles
                        .equ inner = $d                 ;delays 13 clk cycles

                //loads the registers and ports required, run and repated when powere up or reset
                 reset:
                        //initizling the stack pointer
000000 e50f             ldi r16, LOW(RAMEND)            ;load SPL with low byte of
000001 bf0d             out SPL, r16                    ;RAMEND adress
000002 e004             ldi r16, HIGH(RAMEND)           ;load SPH with low byte of
000003 bf0e             out SPH, r16                    ;RAMEND adress

000004 e000             LDI r16, $00                    ;load register 16 with 1's
                // OUT PORTB, r16                        ;load the 7seg diplay
000005 bb07             OUT DDRB, r16                   ;turn on all leds in 7seg
000006 e001             ldi r16, 1                      ;load r16 with 1
000007 bb0a             OUT DDRA, r16                   ;set up led as output
000008 e70f             ldi r16, 0b01111111             ;set only the first switch
000009 bb04             out DDRC, r16                   ;set the pc7 as the input
00000a ef0f             ldi r16, 0b11111111
                // out DDRD, r16                         ;set the lower DIP Switch
00000b ef0f             LDI r16, $ff                    ;load register 16 with 0's
00000c bb02             OUT PORTD, r16                  ;load the pull ups for dip switch
00000d bb05             OUT PORTC, r16                  ;load the pull ups for PBSW
00000e bb08             OUT PORTB, r16                  ;output 0 on 7seg

                //check if the load button is pressed

                main_loop:
00000f e034             ldi r19, 4                      ;r19 used as counter
000010 999f             SBIC PINC, 7                    ;check if the PB switch is pressed
000011 cffd             rjmp main_loop                  ;repeat until button is pressed
000012 d038             rcall delay                     ;wait for 10ms(debounce)
000013 9b9f             SBIS PINC, 7                    ;check if PB is still pressed
000014 c001             rjmp read_switch                ; jump to read switch
000015 cff9             rjmp main_loop                  ;repeat the main loop
```

```
                  //read values of switch
                  read_switch:
000016 b320          in r18, PIND                    ;input the switch values to r18
000017 2f92          mov r25, r18                    ;copy value from r18 to r25
                  // ldi r16, 0                       ;load with 0 to compare with swtich
000018 3020          CPI r18, 0                       ;check weather r18 is 0
000019 f0f1          BREQ clear                       ;jump to clear if equal
00001a e900          ldi r16, $90                        ;load r16 with 9
00001b 0f02          add r16,r18                      ;if there is a negative value then
00001c f0d8          brcs clear                       ;output 0 for anything greater than 9
00001d d01c          rcall frequency
00001e d008          rcall hex_7seg                   ;go to hex7seg and return
00001f d000          rcall duty_pwm                   ;go to dutypwm and return


                  //output the pwm singal
                  duty_pwm:
000020 999f          SBIC PINC, 7                     ;check if load button is pressed
000021 cfed          rjmp main_loop                   ;if pressed restart
000022 9ad8          SBI PORTA, 0                     ;set port a to high
000023 d020          rcall delay_on                   ;turn on the led for cetain period
000024 98d8          CBI PORTA, 0                     ;set port a to low
000025 d021          rcall delay_off                  ;turn off the led for certain period
000026 cff9          rjmp duty_pwm                    ;output the signal until button press




                  //subroutines
                  //-----------

                  //get r18 to 4 digits from 8 digits
                  hex_7seg:
000027 9526          lsr r18                          ;shift lsb to carry
000028 953a          dec r19                          ;decrement r19 for 4 times
000029 f7e9          BRNE hex_7seg                    ;after shifting 4 digits go to bcd_7seg

                  //display the duty cycle value
                  bcd_7seg:
00002a 2f12          mov r17, r18                     ;copy r18 to r17
00002b e0f0          ldi ZH, HIGH(table*2)
00002c e6e6          ldi ZL, LOW(table*2)             ;set z to point to start of the table
00002d e000          ldi r16, $00                     ;clear for later use
00002e 0fe1          add ZL, r17                      ;add low byte
00002f 1ff0          adc ZH, r16                      ;add in the carry
000030 9114          lpm r17, z                       ;load bid pattern from table into r18
                  display:
000031 bb18          out PORTB,r17                    ;output patter for 7 seg display
000032 9508          ret
000033 7940
000034 3024
000035 1219
000036 7803
000037 1800     table: .db $40, $79, $24, $30, $19, $12, $03, $78,$0, $18
                      //  0    1    2    3    4    5    6    7  8    9


                  //output 0 in the signal
                  clear:
000038 98d8          CBI PORTA, 0                     ;set port a to 0
000039 cfd5          rjmp main_loop
                  frequency:
                  // ldi r16, $f                       ;load r16 with 1's
00003a 709f          andi r25, 0b00001111             ;obtain only the 4 lsb
00003b e100          ldi r16, 16                       ;load r16 with 16
00003c 1b09          sub r16, r25                      ;subtract r25 from 16
```

```
00003d 2f90          mov r25, r16                   ;move r16 to r25
00003e e006          ldi r16, 6                     ;load r16 with 6
00003f 9f90          mul r25, r16                   ;multiply r25 with 6
000040 e694          ldi r25, 100                   ;load 100
000041 1990          sub r25, r0                    ;subtract r0 from 100
                // ldi r16, 60
                // mul r25, r16                      ;multiple r25 with 100
                // ldi r16, 1000
                // sub r25, r16                      ;subtract r25 from 1000
000042 939f          push r25                       ;store r25
000043 939f          push r25


                //turn on the signal for the switch value

                delay_on:
000044 914f          pop r20                        ;load r20 with values of r25
                // ldi r20, 100                      ;load r20 with 100
000045 2f52          mov r21, r18                   ;load r21 with switch values
000046 c006          rjmp inner_loop                ;delay for r18*100 cycles

                //turn off the signal for 10 minus switch value
                delay_off:
000047 914f          pop r20                        ;load r20 with value of r25
                // ldi r20, 100                      ;load r20 with 100
000048 e05a          ldi r21, 10                    ;load r21 with 10
000049 1b52          sub r21, r18                   ;subract 10 - switch values
00004a c002          rjmp inner_loop                ;delay for r21*100 cycles

                //delay for 10 ms
                delay:
00004b e04d          ldi r20, inner                 ;set r20 to 240
00004c ef51          ldi r21, outer                 ;set r21 to 13
                inner_loop:
00004d 954a          dec r20                        ;decrements 240
00004e f7f1          brne inner_loop                ;repeat until r20 is 0
                outer_loop:
00004f e04d          ldi r20, inner                 ;reset the r20 to 240
000050 955a          dec r21                        ;decrement for 13 cycles
000051 f7d9          brne inner_loop                ;repeat until r21 is 0
000052 ef51          ldi r21, outer                 ;reset r21 for next delay
000053 9508          ret                            ;return
```

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

"ATmega16" register use summary:
```
r0 :   1 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:   0 r15:   0
r16:  24 r17:   4 r18:   8 r19:   2 r20:   5 r21:   6 r22:   0 r23:   0
r24:   0 r25:   9 r26:   0 r27:   0 r28:   0 r29:   0 r30:   2 r31:   2
x :   0 y :   0 z :   1
```

Registers used: 11 out of 35 (31.4%)

"ATmega16" instruction use summary:
```
.lds   :    0 .sts   :    0 adc    :    1 add    :    2 adiw   :    0 and    :    0
andi   :    1 asr    :    0 bclr   :    0 bld    :    0 brbc   :    0 brbs   :    0
brcc   :    0 brcs   :    1 break  :    0 breq   :    1 brge   :    0 brhc   :    0
brhs   :    0 brid   :    0 brie   :    0 brlo   :    0 brlt   :    0 brmi   :    0
brne   :    3 brpl   :    0 brsh   :    0 brtc   :    0 brts   :    0 brvc   :    0
brvs   :    0 bset   :    0 bst    :    0 call   :    0 cbi    :    2 cbr    :    0
clc    :    0 clh    :    0 cli    :    0 cln    :    0 clr    :    0 cls    :    0
clt    :    0 clv    :    0 clz    :    0 com    :    0 cp     :    0 cpc    :    0
cpi    :    1 cpse   :    0 dec    :    3 eor    :    0 fmul   :    0 fmuls  :    0
fmulsu:     0 icall  :    0 ijmp   :    0 in     :    1 inc    :    0 jmp    :    0
ld     :    0 ldd    :    0 ldi    :   20 lds    :    0 lpm    :    2 lsl    :    0
lsr    :    1 mov    :    4 movw   :    0 mul    :    1 muls   :    0 mulsu  :    0
neg    :    0 nop    :    0 or     :    0 ori    :    0 out    :    9 pop    :    2
push   :    2 rcall  :    6 ret    :    2 reti   :    0 rjmp   :    8 rol    :    0
ror    :    0 sbc    :    0 sbci   :    0 sbi    :    1 sbic   :    2 sbis   :    1
sbiw   :    0 sbr    :    0 sbrc   :    0 sbrs   :    0 sec    :    0 seh    :    0
sei    :    0 sen    :    0 ser    :    0 ses    :    0 set    :    0 sev    :    0
sez    :    0 sleep  :    0 spm    :    0 st     :    0 std    :    0 sts    :    0
sub    :    3 subi   :    0 swap   :    0 tst    :    0 wdr    :    0
```
Instructions used: 25 out of 113 (22.1%)

"ATmega16" memory use summary [bytes]:

Segment	Begin	End	Code	Data	Used	Size	Use%
[.cseg]	0x000000	0x0000a8	158	10	168	16384	1.0%
[.dseg]	0x000060	0x000060	0	0	0	1024	0.0%
[.eseg]	0x000000	0x000000	0	0	0	512	0.0%

Assembly complete, 0 errors, 0 warnings