Task1:

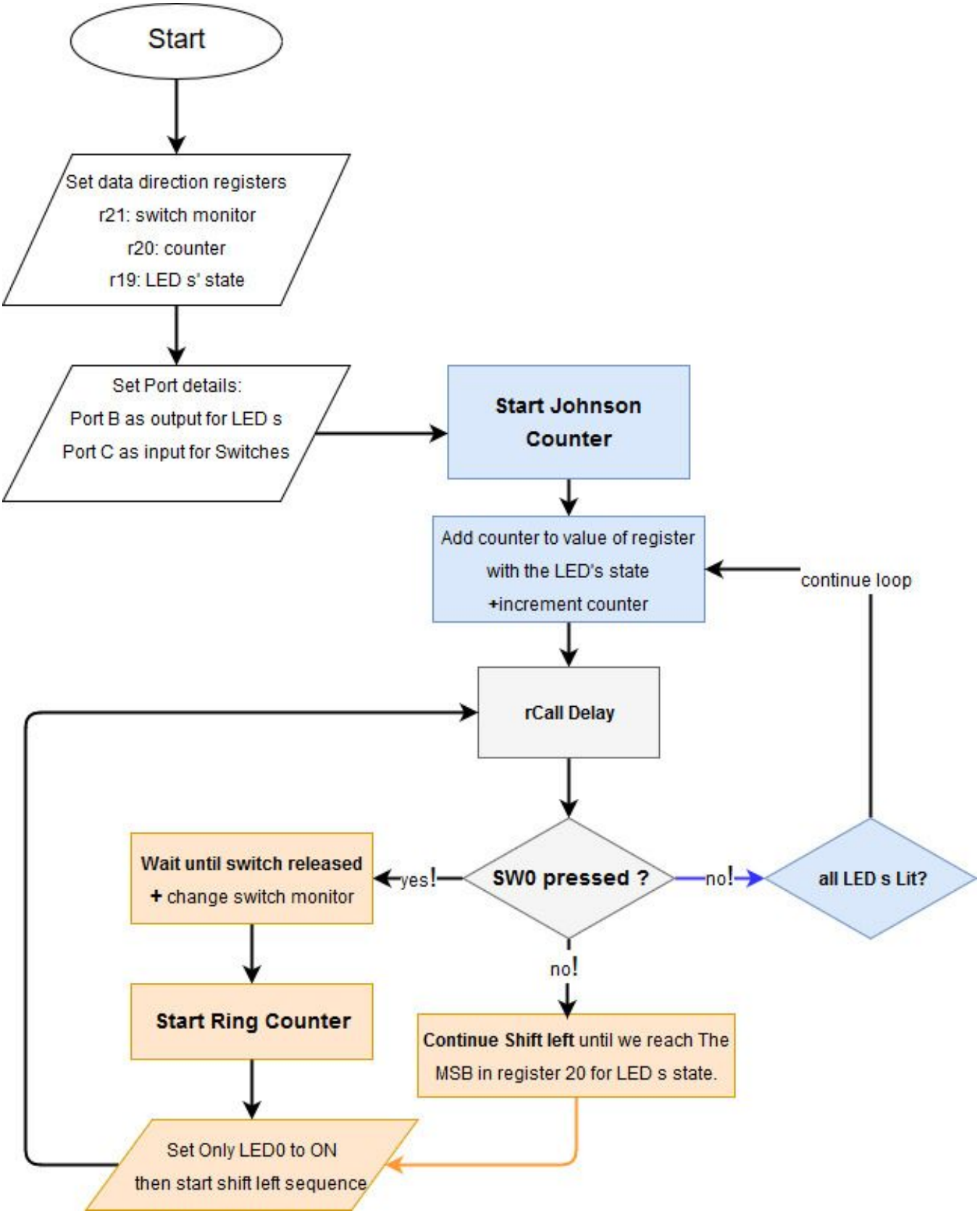
```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2016-09-15
; Authors:
; Mehdi Hmidi
; Jorian Wielink
;
; Lab number: 2
; Title: Task 1
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Switch Ring counter/ Johnson Counter,The pushbutton must be checked frequently, so there is no
;delay between the button is pressed and the change between Ring/Johnson. Use SW0 (PA0) for the button.
;Each time you press the button, the program should change counter
;
; Input ports: On board switches on portA
; Output ports: on-board LEDs connected to PORTB.
;
; Subroutines: delay, johnson, sub and ring counter.
; Included files: m2560def.inc
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"
;set initial states: Set data direction registers---------------
        ldi     r21, 0x00       ; switch value
        ldi     r20, 0x01
        ldi     r19, 0xFF       ; r19 used for portB state

;set port details:---------------------------------------------
        ldi     r16, 0xFF       ; set ddrb ports as output
        out     DDRB, r16
        out     PORTB, r16      ; 0XFF on the board translates to off state
        ldi     r16, 0x00       ; set ddrc port as input
        out     DDRC, r16
        rjmp johnson
switch:
        in      r23, PINC       ; Switch in pressed condition. User did not release button yet.
        cpi     r23, 0xFE       ; Check if switch has been released
        breq    switch          ; loop until release of switch.
        com     r21             ; we change the counters depending on previous state.
        brne    ring_counter    ; if not equal to zero, branch to ring counter.
                ; otherwise, continue with johnson counter below.
        ldi     r20, 0x01
        ldi     r19, 0xFF       ;  Johnson counter parameters reinitialized.
johnson:
        out     PORTB, r19
        com     r19             ; one´s complement is used to invert the bits (to successfully add
        add     r19, r20        ; r20 to r19).
        lsl     r20             ; logical shift left to light the LED to the consequent left
```

```
        com     r19             ; for output purposes, one´s complement again.
        rcall   delay
        cpi     r19, 0x00       ; conditional statement, checks if end has been reached(LEds Lit).
        brne    johnson         ; if end not reached continue "while" loop.
j_return:
        out     PORTB, r19
        com     r19             ; same principle as above, only now we only need to shift right
        lsr     r19             ; after doing a one's complement.
        com     r19
        rcall   delay
        cpi     r19, 0xFF       ; if all bits are set, skip looping this sub
        brne    j_return
        ldi     r20, 0x01       ; reset r20 to start adding to r19 (output to LED) from the right
        rjmp    johnson
ring_counter:
        ldi     r20, 0xFE       ; all leds except led 0 are off
   loop:
        out     PORTB, r20      ; light the leds
        rcall   delay
        com     r20
        lsl     r20
        brcs    ring_counter
        com     r20             ; complement back AFTER possible branch to
   rjmp  loop                   ; reset, to account for carry flag in brcs!

delay:
        ldi     r16, 3
delay_1:
        ldi     r17, 10
delay_2:
        ldi     r18, 150
delay_3:
        in      r23, PINC
        cpi     r23, 0xFE
        breq    switch   ;check while in the delay for input(press of switch)
        dec     r18        ;continue delay sequence.
        brne    delay_3
        dec     r17
        brne    delay_2
        dec     r16
        brne    delay_1
        ret
```

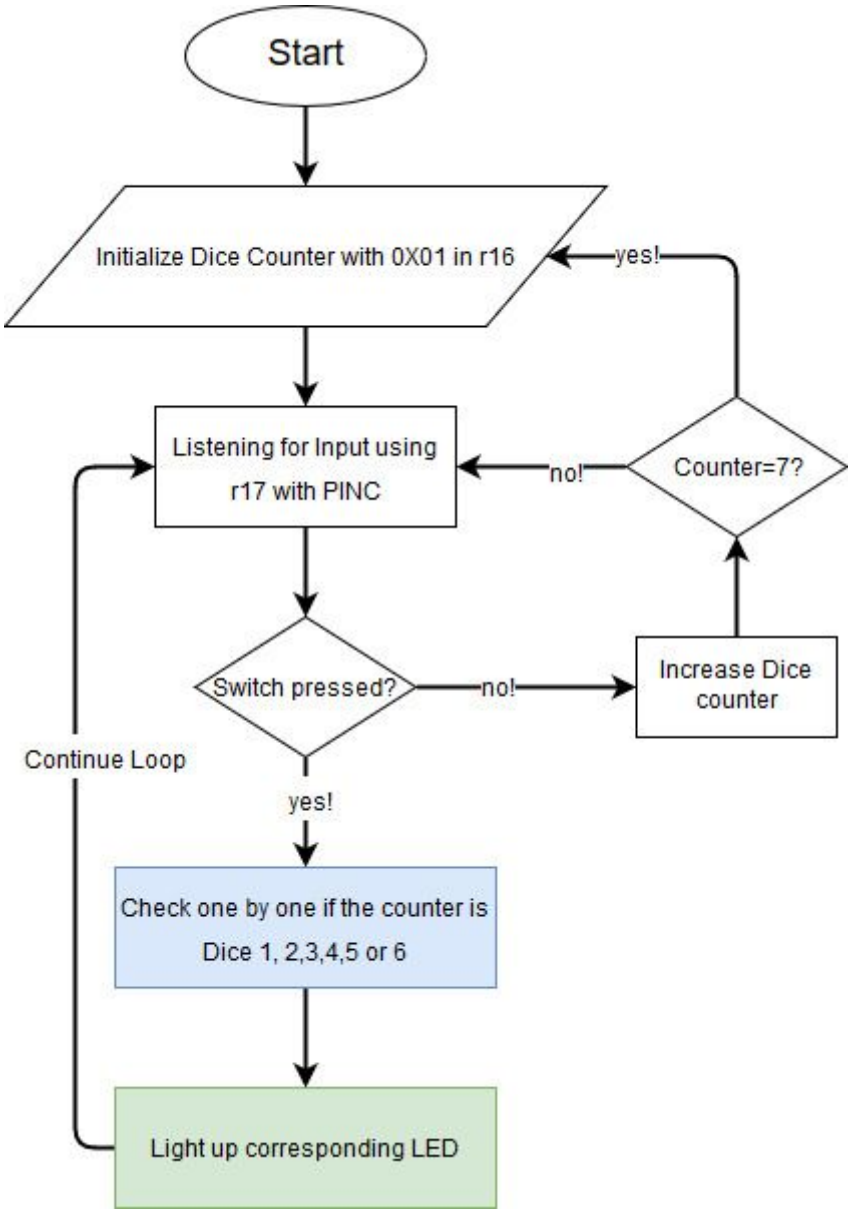

Start
Set data direction registers
r21: switch monitor
r20: counter
r19: LED s' state
Set Port details:
Port B as output for LED s
Port C as input for Switches
Start Johnson Counter
Add counter to value of register with the LED's state
+increment counter
continue loop
rCall Delay
SW0 pressed ?
yes!
no!
Wait until switch released
+ change switch monitor
all LED s Lit?
no!
Start Ring Counter
Continue Shift left until we reach The MSB in register 20 for LED s state.
Set Only LED0 to ON
then start shift left sequence

Task2:

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2016-09-15
; Authors:
; Mehdi Hmidi
; Jorian Wielink
;
; Lab number: 2
; Title: Task 2
;
; Hardware: STK600, CPU ATmega2560
;
; Function: An Electronic Dice, the number 1 to 6 should be generated randomly. use the fact that the time you press the button ; ;varies in length.
;
; Input ports: On board switches on portC
; Output ports: on-board LEDs connected to PORTB.
;
; Included files: m2560def.inc
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"
init:
        ldi r16, 0          ; random output value
        rjmp start
loop:
        in r17, PINC
        cpi r17, 0xff
        brne loop
start:
        inc r16
        cpi r16, 7
        breq init

        in r17, PINC
        cpi r17, 0xff
        brne output
        rjmp start          ; start over
output:
        cpi r16, 1
        breq dis_1
        cpi r16, 2
        breq dis_2
        cpi r16, 3
        breq dis_3
        cpi r16, 4
        breq dis_4
        cpi r16, 5
        breq dis_5
        cpi r16, 6
        breq dis_6
```

```
        rjmp loop
dis_1:
        ldi r16, 0b00010000
        out DDRB, r16
        rjmp loop
dis_2:
        ldi r16, 0b01000100
        out DDRB, r16
        rjmp loop
dis_3:
        ldi r16, 0b01010100
        out DDRB, r16
        rjmp loop
dis_4:
        ldi r16, 0b11000110
        out DDRB, r16
        rjmp loop
dis_5:
        ldi r16, 0b11010110
        out DDRB, r16
        rjmp loop
dis_6:
        ldi r16, 0b11101110
        out DDRB, r16
        rjmp loop
```



Task3:

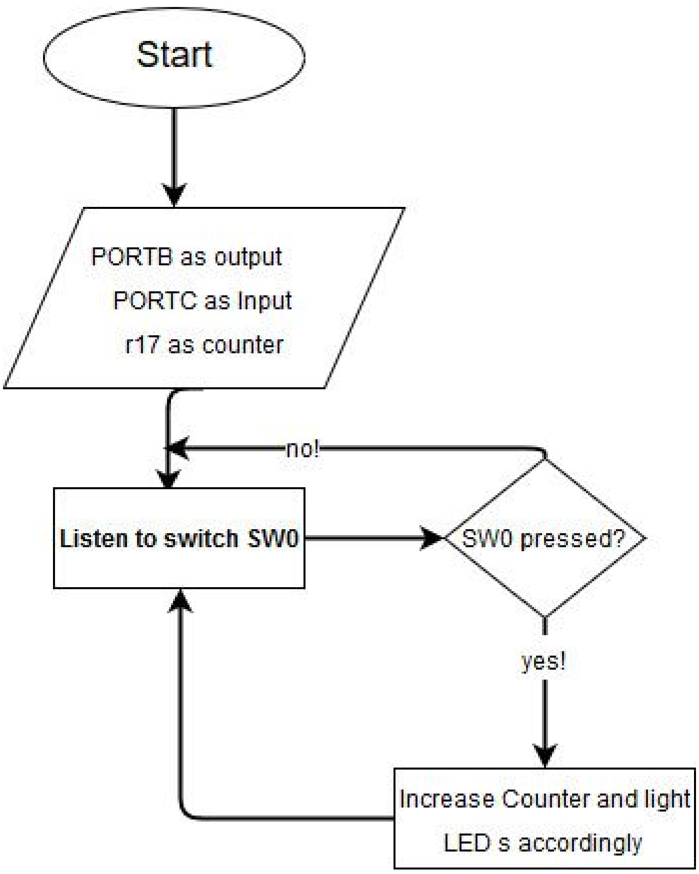
```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2016-09-15
; Authors:
; Mehdi Hmidi
; Jorian Wielink
;
; Lab number: 2
; Title: Task 3
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Count the number of changes on a switch. As a change we count when the switch SW0 goes from 0
to 1 and from 1 to 0, we expect therefore positive and negative edges. We calculate the changes in a byte
variable and display its value on PORTB.
;
; Input ports: On board switches on portA
; Output ports: on-board LEDs connected to PORTB.
;
; Subroutines:; Included files: m2560def.inc
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

    ldi     r16, 0xff               ; count in binary
    out     DDRB, r16               ; set data direction as output
    Out     PORTB, r16
    ldi     r16, 0x00               ; set portC as input
    out     DDRC, r16

    ldi     r17, 0x00               ; used as counter
    rjmp    start

switch:
    inc     r17                     ; pushed
    com     r17
    out     PORTB, r17
    com     r17
    rcall delay
    loop:
            in      r16, PINC       ;PORT C in r16
            cpi  r16, 0xFF          ; switch released.
            brne loop
    rcall delay
    inc     r17                     ; released
    com     r17
    out     PORTB, r17
    com     r17

start:
        in          r16, PINC
```



```
        cpi     r16, 0xFE           ; SW0 is pushed
        breq    switch
        rjmp    start
Delay:
        ldi  r18, 2                 ; accounts for the delay. Equals 1 ms (including all overhead calls)
        ldi  r19, 74
L1:
        dec  r19
        brne L1
        dec  r18
        brne L1
ret
```

Task4:

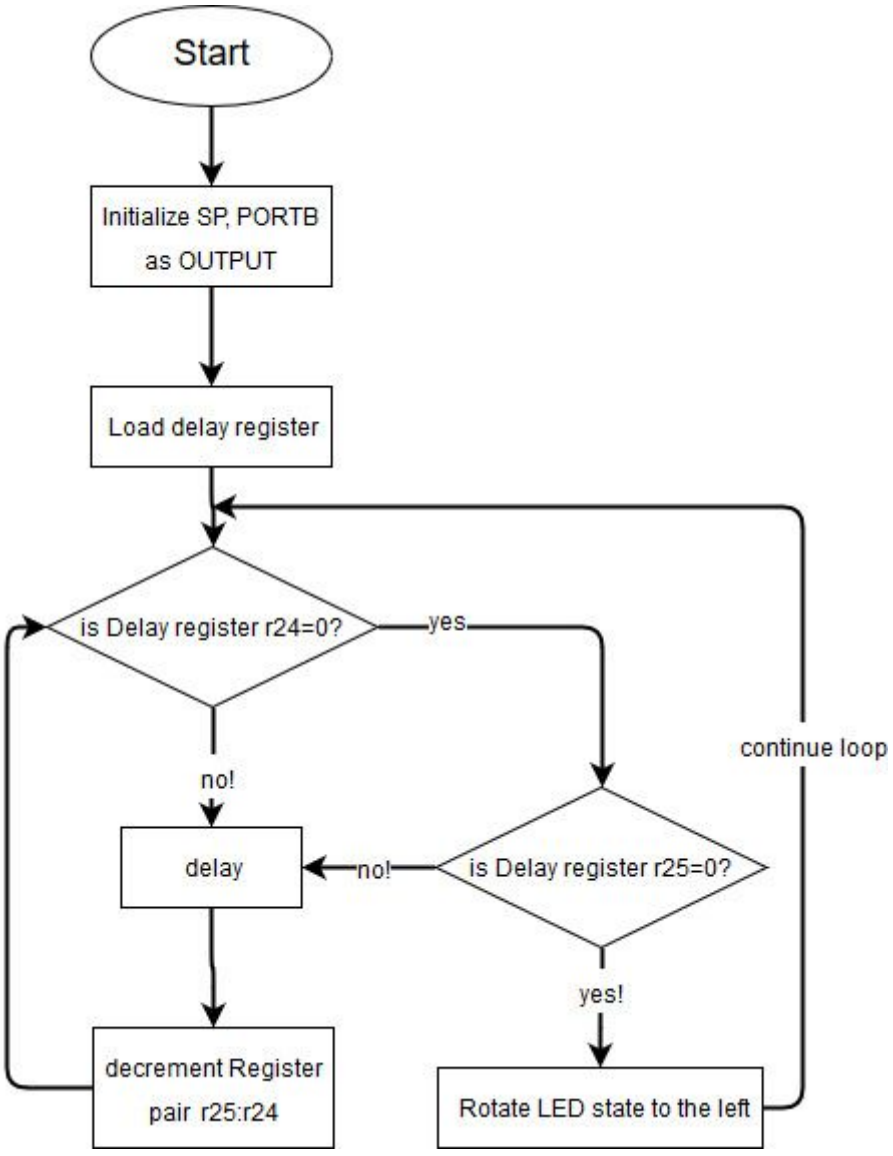
```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2016-09-15
; Authors:
; Mehdi Hmidi
; Jorian Wielink
;
; Lab number: 2
; Title: Task 4
;
; Hardware: STK600, CPU ATmega2560
;
; Function: a general delay routine that can be called from other programs. It should be named
;wait_milliseconds. The number of milliseconds should be transferred to register pair R24, R25.
;
;; Output ports: on-board LEDs connected to PORTB.
;
; Subroutines:; Included files: m2560def.inc
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"
        ldi     r20, HIGH(RAMEND)           ; R20 = high part of RAMEND address
        out     SPH, R20                    ; SPH = high part of RAMEND address
        ldi     R20, low(RAMEND)            ; R20 = low part of RAMEND address
        out     SPL, R20                    ; SPL = low part of RAMEND address

   ldi      r16, 0xff
        out     ddrb, r16       ; ddrb as output
        out     portb, r16      ; switch off leds
        ldi     r17, 0xff       ; r17 drives leds
loop:
        out     PORTB, r17              ; Write state to LEDs
        ldi     r25, HIGH(500)
        ldi     r24, LOW(500)
        rcall   wait_milliseconds       ; Delay to make changes visible
        rol     r17             ; Rotate LED state to the left
        rjmp    loop
```

```
wait_milliseconds:
 waitfor_lowzero:
  cpi r24, 0                ; wait for lower nibble to clear to 0 (contains least significant bits)
  breq waitfor_highzero             ; if it´s decreased to zero, start wait for high zero. repeat this until
high zero
                                                                        ; has reached zero too.
  rjmp decrement_delay

 waitfor_highzero:                  ; contains most significant bits (bit 9 - 15)
           cpi r25, 0               ; if this also zero, the time is up and we branch to subroutine zero.
           breq zero
           rjmp decrement_delay
   zero:                    ; returns to the loop driving the LEDS.
          ret
decrement_delay:                    ; this decreases the 16-bit value by using subtract immediate (SBIW)
          sbiw r25:r24, 1           ; this delay is called as many times defined in the register pair in ms.
          ldi  r18, 2               ; accounts for the delay. Equals 1 ms (including all overhead calls)
          ldi  r19, 74
L1:
          dec  r19
          brne L1
          dec  r18
          brne L1

          rjmp waitfor_lowzero      ;
```


Start
Initialize SP, PORTB as OUTPUT
Load delay register
is Delay register r24=0?
yes
no!
delay
no!
is Delay register r25=0?
yes!
continue loop
decrement Register pair r25:r24
Rotate LED state to the left