

## Report

# Laboratory 3



September 27, 2017

Author:

Caroline Nilsson (cn222nd)

Daniel Alm Grundström (dg222dw)

Term: HT 2017

Course: 1DT301 - Computer

Technology I

### Contents

1	Assignment 1	1
2	Assignment 2	3
3	Assignment 3	7
4	Assignment 4	11

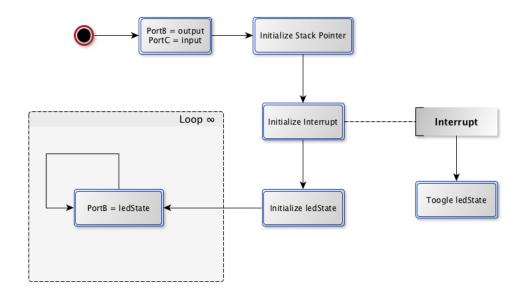


Figure 1: Toggle LED0 on Switch pressed using interrupt

```
1DT301, Computer Technology I
Date: 2017-09-25
Author:
                                                    (cn222nd)
(dg222dw)
                          Caroline Nilsson
Daniel Alm Grundström
        Lab number:
                          Interrupts
        Title:
Hardware:
                          STK600, CPU ATmega2560
                          Toogle LED0 ON and OFF by pressing Switch0
        Function:
                          PORTD
       Input ports:
        Output ports:
                          PORTB
        Subroutines:
                          delay
                                         -delay to avoid bouncing-within interrupts
        Included files:
                          m2560def.inc
        Other information: N/A
        Changes in program: 2017-09-20
                           Implementation of flowchart design
                          2017\!-\!09\!-\!21
                          Bugfixes during laboratory
                          2017-09-25
                          Overview code and commentary, missing commentary added
    .include "m2560def.inc"
    . def temp = r16
. def ledState = r17
. def xorComparison = r18
                                  ; stores output value ; used to toogle bit 0 in ledState
    .equ XOR_BIT_STRING = 0b0000_0001
    .cseg
    . org 0x00 rjmp reset
    ;>>>>>>
    ; initialize interrupt start
    .org int0addr
rjmp interrupt
```

```
59
60
61
62
    . org 0x72
    reset:
    63
64
     initialize Stack Pointer
    65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
    \begin{array}{c} \textbf{Idi} & temp \;,\;\; LOW(\textcolor{red}{RAMEND}) \end{array}
    out SPL, temp
    ldi temp, HIGH(RAMEND)
out SPH, temp
    ser temp
out DDRB, temp
    ; set portD as input ; <<<<<<<<<<<><<<<>><<<>><<<>><<<>><</>;
    clr temp

    ser
    ledState
    ; set default value for ledState

    ldi
    xorComparison , XOR_BIT_STRING
    ; initialize xorComparison

82
83
84
85
    86
87
    ; enable external interrupt on PINDO
    .<<<<<<
    ldi temp, (1<<int0)
out EIMSK, temp
88
89
90
91
    ;>>>>>>>>>>>>>>>>>
    set interrupt sence control to "on rising edge"
92
93
94
95
96
97
98
99
    ldi temp, (3<<ISC00)
sts EICRA, temp
    100
    ; Sets portB to ledState in an infinite loop
101
    102
    main_loop:
out PORTB, ledState
103
104
       rjmp main_loop
106
    ; interrupt start
; performas XOR to toogle LED0 between "ON" and "OFF" and
; calls delay to avoid bouncing
108
    110
    interrupt:
       eor ledState, xorComparison
112
       rcall delay
114
       reti
   ;>>>>>>>; Generated by delay loop calculator ; at "http://www.bretmulvey.com/avrdelay.html"
116
118
119
    ; Delay 200 000 cycles
120
121
    ; 200ms at 1 MHz
122
123
     Delay for 200 ms
124
    ······
125
126
   ldi r19, 4
ldi r20, 187
L1: dec r20
127
128
129
130
       brne L1
131
132
       dec r19
brne L1
133
       dec r30
brne L1
       nop
ret
135
```

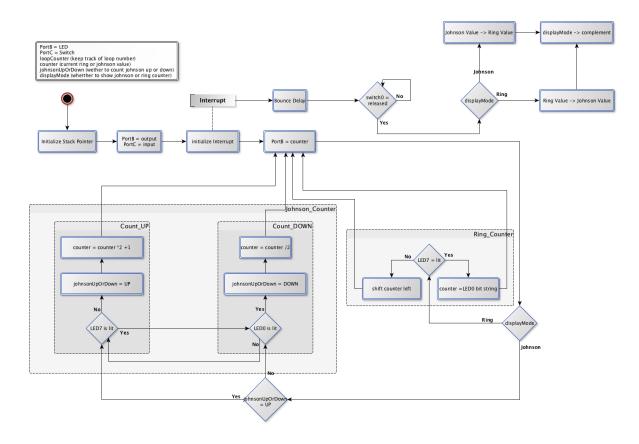


Figure 2: Johnson/Ring counter toggle using interrupt

```
1DT301, Computer Technology I
           Date: 2017-09-25
Author:
                                      Caroline Nilsson
Daniel Alm Grundström
                                                                            (cn222nd)
(dg222dw)
           Lab number:
           Title:
                                      Interrupts
STK600, CPU ATmega2560
           Hardware:
                                      Counts up a counter and display it's value as either a Ring counter or Johnson counter. The display mode can
           Function:
                                      be toggled between ring/johnson by pressing switch SWO.
                                      PORTD
           Input ports:
                                      PORTB
           Output ports:
           Subroutines:
                                      led_out
                                                              - Outputs counter to LEDs

    Outputs counter to LEDs
    Delay to make changes to LEDs
    visible. Also continuously checks if switch is pressed.
    Counts up ring counter
    Counts johnson counter up/down
    Delay of 10 ms used after switch is

                                      delay_led
                                       ring_counter
                                      johnson_counter
delay_switch
                                                                  pressed down
           Included files:
                                      m2560def.inc
           Other information: N/A
           Changes in program: 2017-09-20
                                      Implementation of flowchart design
                                      2017 - 09 - 21
                                      Bugfixes during laboratory
                                      Overview code and commentary,
                                      missing commentary added
      :<<<<<<
```

```
.include "m2560def.inc"
     . def displayMode = r16
                                                 ; determines whether to output ring or johnson ; keeps track of output value
48
49
50
51
52
53
54
55
56
57
58
     def counter = r17
                                                ;keeps track of output value
;use to set input and output on PORTs
;whether to count johnson value up or down
;temp, to output counters complement
;constant: value of up
;constant: value of down
;constant: Johnson display mode
;constant: Ring display mode
;constant: PIN of switch to check
     . def temp = r18
. def johnUpOrDown = r20
     . def complement = r21
. equ UP = 0x01
      . equ DOWN = 0x00
     .equ JOHNSON = 0x00
.equ RING = 0xFF
.equ SWITCH = PIND0
59
60
61
     ;>>>>>
     ; initialize starting point for program ;<<<<<<<<<<<<<<<<<<<<><<<<>.org 0x00
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
     rjmp start
     : initialize interrupt start
     ;<<<<<<<<<<<<></></>
     .org intOaddr rjmp interrupt
     .org 0x72
     ; initialize Stack Pointer
.<<<<<<<<<<<<<<<<<<>>
79
80
     ldi r18, HIGH(RAMEND)
     out SPH, r18
ldi r18, LOW(RAMEND)
out SPL, r18
81
82
83
84
     ;>>>>>>>>>>>>>>>>>
      ; set PORTB to output ; <<<<<<<>>
85
86
87
88
89
90
91
     \textcolor{red}{\textbf{Idi}} \hspace{0.2cm} temp \hspace{0.1cm}, \hspace{0.1cm} 0xFF
     out DDRB, temp
     ;>>>>>>>>>>>>>>>>>
      : set PORTD to input

<
92
93
94
95
96
97
     ldi temp, 0x00
out DDRD, temp
     ; initialize starting state
;<<<<<<td>; initialize starting state
;
98
99
     ldi counter, 0x01
ldi johnUpOrDown, UP
101
103
     ;>>>>>>>>>>>>>>>>>
      ; enable external interrupt on PINDO
       ......
105
106
107
     ldi temp, (1<<int0)
out EIMSK, temp
108
109
     :>>>>>>>>>>
110
111
      ; set interrupt sence control to "falling edge"
;<<<<<<<<<<<<>>
112
113
     ldi temp, (2<<ISC00)
sts EICRA, temp
114
115
116
     ;>>>>>>>>>>>>>>>>>
118
      ; calls Ring/Johnson counter subrutine depending on ; the state of displayMode
119
120
      121
     main loop:
          cpi displayMode, JOHNSON
breq johnson1
122
                                                 ; if displaymode = johnson
                                                ; then jump to johnson branch
123
124
          ring1:
                                                ; else jump to ring
            rcall ring_counter
rjmp main_loop
126
128
         johnson1:
reall johnson_counter
130
131
132
         rimp main loop
134
135
      : Outputs complement of the current value of counter to LEDs
136
       138
     led out:
         mov complement, counter
com complement
139
140
141
142
          out PORTB, complement
143
144
145
146
     ; Delay to show LED output
```

```
148
     delay_led:
        push johnUpOrDown
149
150
        ldi r18, 3
ldi r19, 138
ldi r20, 86
dec r20
151
153
154
    L1: dec
155
        brne L1
156
157
        dec r19
brne L1
158
159
        dec r18
brne L1
        rjmp PC+1
161
        pop johnUpOrDown
163
165
166
167
    ; Creates the ring counter by writing the complement of counter ; to PORTB and then increments the ring counter
169
     170
     ring_counter:
        counter:
sbis PORTB, PINB7
ldi counter, 0x01
                                         if the 7th led is lit
171
                                         ; then set counter to one
173
        sbic PORTB, PINB7
Isl counter
                                        ; else
174
175
                                        ; shift counter to the left
        rcall led_out
177
178
179
        rcall delay_led
180
181
182
183
     Creates the johnson counter by writing the complement of counter to PORTB and then checks wheter to count up or down
184
185
186
      187
    johnson_counter:
        cpi johnUpOrDown, UP
breq count_up
                                        ; if count up is active
188
                                        ; then jump to count up
190
191
        rjmp count_down
                                         ; else jump to count down
192
        ;>>>>>>
        ; checks whether to continue to count up and ; increments the johnson value
194
195
196
         count_up:
sbis PORTB, PINB7
                                         ; if the 7th led is lit
198
                                        ; then jump to count down
              rjmp count_down
200
            ldi johnUpOrDown, UP
                                        : shift to the left
            lsl counter
inc counter
202
203
204
                                         : add one
206
            rjmp end
207
208
        ; checks whether to continue to count down and ; decrese the johnson value ; <<<<<<<<<<<<<<<<<c>count_down:
210
211
212
            sbic PORTB, PINBO
rjmp count_up
213
214
                                         ; if the right most led is not
                                         ; lit then jump to count up
215
            ldi johnUpOrDown, DOWN lsr counter
216
                                        ; shift to the right
217
219
220
            rcall led_out
221
            rcall delay_led
223
            ret
224
225
227
228
229
230
231
     ; Delay 10 ms to avoid bouncing when switch is pressed
      delay_switch:
    ldi r31, 13
ldi r30, 252
L2: dec r30
232
233
234
235
        brne L2
        dec
        brne L2
236
237
        nop
238
239
        ret
240
241
     242
243
     ; interrupt start
; calls delay to avoid bouncing
244
245
      <<<<></></></></></></t>
     interrupt:
246
247
        rcall delay_switch
```

```
248 249 250 251 252 253 254 255 256 257 262 263 264 265 267 271 272 273 274 275 276 277 280 281 282 283 284 285 286
           ; wait until switch is released then branch depending ; on display mode ; <<<<<<<<<>>>>
           switch_release:
sbis PIND, SWITCH
rjmp switch_release
          cpi displayMode , JOHNSON
breq johnson_to_ring
           ;>>>>>>>>>>>>>>>>
           ; convert ring value to johnson value ; call LED output and delay
            ring_to_johnson:
               lsl counter
dec counter
rcall led_out
rcall delay_led
               rjmp switch_end
           ;>>>>>>>>>; convert johnson value to ring value;
           johnson_to_ring:
               lsr counter inc counter
           ;>>>>>>>>>; toogle displayMode and remove PC from Stack, set interruption
           switch_end:
com displayMode
287
288
               pop temp
sei
               rjmp main_loop
```

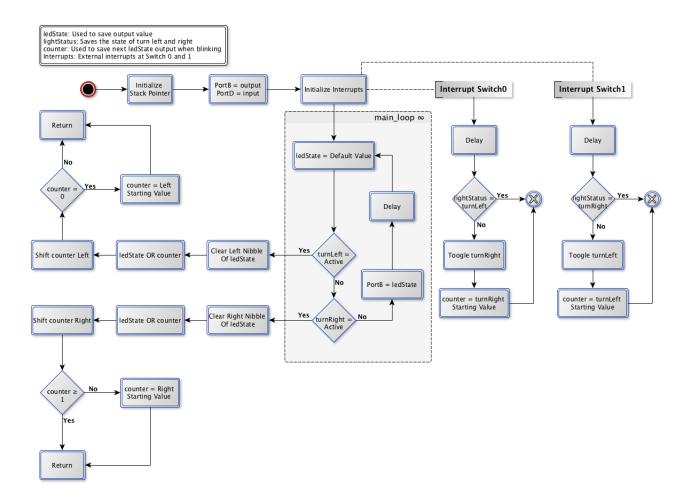


Figure 3: Simulating rear lights blink right/blink left

```
1DT301, Computer Technology I
Date: 2017-09-25
             Author:
                                                                                         (cn222nd)
                                             Caroline Nilsson
                                             Daniel Alm Grundström
                                                                                         (dg222dw)
             Lab number:
Title:
                                             Interrupts
                                             STK600, CPU ATmega2560
             Hardware:
12
13
14
15
16
17
18
19
                                             Simulates the rear lights on a car using LED.

Default light -> back lights

Holding down Switch0 -> blinking right

Holding down Switch1 -> blinking left

but not possible to blink right and left
              Function:
                                             at the same time
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
                                             PORTD
             Input ports:
                                             PORTR
             Output ports:
                                             blink_left
blink_right

    toogle LED for left blinking
    toogle LED for right blinking
    outputs ledState to PORTB

             Subroutines:
                                             led out
                                             delay_led
                                                                       -delay to show changes made to LEDs
                                                                        500 ms
                                             delay_switch
                                                                       -delay to avoid bouncing
                                                                        100 ms
             Included files:
                                             m2560def.inc
                                             back lights LED = 0.1, 6.7 is lit blink right LED = 6.7 lit and toogle between having LED = 3.2, 1.0 lit blink left LED = 0.1 lit
             Other information:
                                             and toogle between having LED 4,5,6,7 lit
```

```
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
          Changes in program: 2017-09-20
                                Implementation of flowchart design
                               2017 - 09 - 21
                                Bugfixes during laboratory
                                Overview code and commentary,
                                missing commentary added
     . def temp = r16
. def ledState = r17
     def ledState = r17
def lightStatus = r18
def counter = r19
def complement = r20
     .def xorComparison = r21 ; used to toogle bits in lightStatus
     .equ TURN_RIGHT = 0 ; mask for right blinking bit
.equ TURN_LEFT = 7 ; mask for left blinking bit
.equ COUNTER_RIGHT_RESET = 0b0000_1000 ; blink right starting state
.equ COUNTER_LEFT_RESET = 0b0001_0000 ; blink left starting state
60
61
63
64
65
66
67
     ; initialize starting point for program ;:
68
69
      org 0x00
 70
71
     rjmp start
     74
75
      ; initialize interrupt starting point for Switch0
     rjmp interrupt_right
78
79
80
     82
83
84
     rjmp interrupt_left
     . org 0x72
     ;>>>>>>>>>>>>>>>>>
88
89
90
91
92
       initialize Stack Pointer
      ldi r16, HIGH(RAMEND)
     out SPH, r16
93
94
95
96
97
98
99
100
     ldi r16, LOW(RAMEND)
     out SPL, r16
     set PORTB to output
       ......
     ldi temp, 0xFF
     out DDRB, temp
102
     103
104
      set PORTD to input
     ldi temp, 0x00
out DDRD, temp
105
106
107
     clr lightStatus
                                            ; initialize starting state
108
109
110
     ;>>>>>>>>>>>>>>>>>
       enable external interrupts on PINDO and PIND1
113
     ldi temp, (3<<int0)
out EIMSK, temp</pre>
115
     116
117
119
     ldi temp, (1<<ISC00)|(1<<ISC10)
sts EICRA, temp
121
123
124
125
     ;>>>>>
      ; sets backlight strings to ledState
; calls subroutine blink_left/blink_right depending on
127
      : lightStatus state
       calls led_out and delay_led
129
      ·-----
      main_loop:
          clr ledState
131
         sbrs lightStatus, TURN_LEFT ; if TURN_LEFT bit is clear
sbr ledState, 0b1100_0000 ; set back lights bits left
133
135
136
137
         sbrs lightStatus , TURN_RIGHT
    sbr ledState , 0b0000_0011
                                           ; if TURN_RIGHT bit is clear
; set back lights bits right
         sbrc lightStatus, TURN_LEFT
```

```
rcall blink_left
sbrc lightStatus, TURN_RIGHT
rcall blink_right
140
141
142
143
        rcall led_out rcall delay_led
144
146
        rjmp main_loop
    ;>>>>>
148
    150
152
     blink left:
        cbr ledState, 0xF0
        or ledState, counter clc
lsl counter
brcc end_left
154
                                          ; clear carry flag
156
157
                                          ; if carry is clear jump
158
                                           to end left
159
160
        ldi counter. COUNTER LEFT RESET
                                          ; else reset counter to
                                           ; starting value
        end left:
162
163
164
165
    ; clears right side of ledState, rotate counter bit in range ; 3, 2, 1, 0 and adds it to ledState
166
167
168
      blink_right:
    cbr ledState, 0x0F
    or ledState, counter
lsr counter
170
171
172
        cpi counter, 1
brge end_right
173
                                          ; if counter >=1 jump to
175
                                           ; end_right
176
                                          ; else reset counter to ; starting value
        ldi counter, COUNTER_RIGHT_RESET
178
179
180
        end_right:
181
183
    ; outputs complement of ledstates current value to PORTB
185
      led_out:
        mov complement, ledState com complement
187
189
        out PORTB, complement
191
        ·>>>>>
193
     ; 500 ms delay to show the LED output
     ; Generated by delay loop calculator
195
196
197
     ; at http://www.bretmulvey.com/avrdelay.html
198
199
     ; Delay 500 000 cycles
; 500ms at 1 MHz
200
201
      </r/>
     delay led:
        ldi r31, 3
ldi r30, 138
203
204
205
    ldi r29, 86
L1: dec r29
        brne L1
        dec r30
brne L1
207
208
209
        dec r31
brne L1
210
211
        rjmp PC+1
212
213
214
    ;>>>>>>
    ; 100 ms delay to avoid bouncing
216
    ; Generated by delay loop calculator
; at http://www.bretmulvey.com/avrdelay.html
217
218
220
221
     ; Delay 100 000 cycles
; 100ms at 1 MHz
222
      223
224
     delay_switch:
    ldi r30, 130
ldi r29, 222
L2: dec r29
225
226
        brne L2
        dec
        brne L2
229
230
        nop
ret
231
232
233
234
     ; calls delay, break if turn left is active otherwise toogle; lightStatus TURN_RIGHT bit ON/OFF and reset counter
      interrupt_right:
    reall delay_switch
237
239
240
        sbrc lightStatus, TURN_LEFT
```

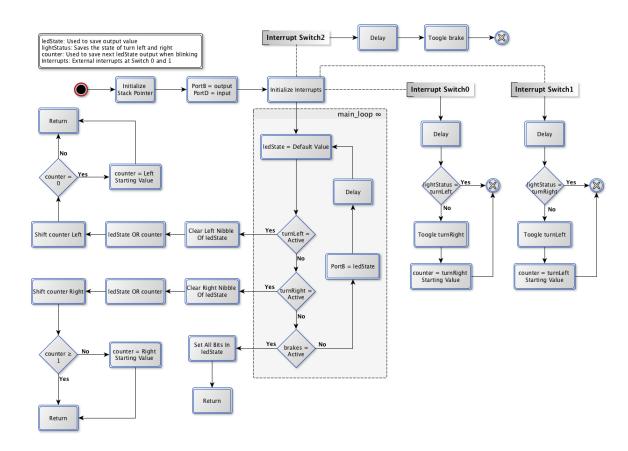


Figure 4: Simulating rear lights blink right/blink left/brake

```
1DT301, Computer Technology I
Date: 2017-09-25
             Author:
                                            Caroline Nilsson
                                                                                      (cn222nd)
                                            Daniel Alm Grundström
                                                                                      (dg222dw)
             Lab number:
8
9
10
                                            Interrupts
             Hardware:
                                            STK600, CPU ATmega2560
Function:
                                            Simulates the rear lights on a car using LED.
                                            Default light —> back lights
Holding down Switch0 —> blinking right
Holding down Switch1 —> blinking left
Holding down Switch2 —> brake
but not possible to blink right and left
at the same time but possible to brake and
blink
                                            blink
             Input ports:
                                           PORTD
                                           PORTB
             Output ports:
                                            blink_left
                                                                    -toogle LED for left blinking
             Subroutines:
                                                                    -toogle LED for right blinking
-toogle LEDs for braking
                                            blink_right
                                            brake
                                                                    -outputs ledState to PORTB
-delay to show changes made to LEDs
                                            led_out
                                            delay_led
                                                                    500 ms
-delay to avoid bouncing 100 ms
                                            delay_switch
             Included files:
                                            m2560def.inc
                                                                   \begin{array}{cccc} LED - & 0\,,1\,,6\,,7 & is & lit \\ LED - & 6\,,7 & lit \end{array}
             Other\ information:
                                            back\ lights -\!\!\!>
                                            blink right ->
                                            and toogle between having LED 3,2,1,0 lit blink left -> LED - 0,1 lit and toogle between having LED 4,5,6,7 lit brake -> lit all LEDs
             Changes in program: 2017-09-20
```

```
Implementation of flowchart design
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
                                 2017-09-21
                                 Bugfixes during laboratory
                                  2017-09-25
                                 Overview code and commentary, missing commentary added
      ......
      . def temp = r16
      . def ledState = r17
                                     ;LED output value
     . def lightStatus = r18
. def counter = r19
                                   ; masking register for backlight modes
; used in blink subroutines
      . def complement = r20
61
62
63
64
65
66
67
68
69
70
71
72
73
      .equ TURN_RIGHT = 0 ; mask for right blinking bit
.equ TURN_LEFT = 7 ; mask for left blinking bit
.equ BRAKES = 2 ; mask for brake bit
.equ COUNTER_RIGHT_RESET = 0b0000_1000 ; blink right starting state
.equ COUNTER_LEFT_RESET = 0b0001_0000 ; blink left starting state
      : initialize starting point for program
 74
75
      ;<<<<<<<<<<<<<<<<<<<><
      org 0x00
 76
77
      ; initialize interrupt starting point for Switch0 ;<<<<<<<<<><<<>><<
80
81
      .org intOaddr
 82
      rjmp interrupt_right
      :>>>>>>>>>>>>>>>>
84
85
      ; initialize interrupt starting point for Switch1
86
      :<<<<<<<
88
      rjmp interrupt_left
89
90
      ; initialize interrupt starting point for Switch2
91
92
93
94
95
96
97
98
      rjmp interrupt_brake
      . org 0x72
      start:
99
100
      ; Initialize stack pointer
;<<<<<<<<<<<<>>
     ldi r16, HIGH(RAMEND)
out SPH, r16
ldi r16, LOW(RAMEND)
out SPL, r16
102
104
105
106
      ; set PORTB to output
108
109
110
      ;<<<<<<<<<<</td>
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;
    ;

      out DDRB, temp
112
113
      ;>>>>>>>>>>>>>>>>
      ; set PORTD to input
114
115
       ......
116
      ldi temp, 0x00
117
118
      out DDRD, temp
119
      clr lightStatus
                                                   ; initialize starting state
120
121
      ; enable external interrupt on PINDO, PIND1 and PIND2
123
       ldi temp, (7<<int0)
out EIMSK, temp
125
127
      ;>>>>>>>>>>>>>>>>>
      ; set interrupt sense control to "Any edge" ;:
129
130
131
      | Idi temp, (1<<ISC00) | (1<<ISC10) | (1<<ISC20) | sts EICRA, temp
      sei
133
135
      ; sets backlight strings to ledState; calls subroutine blink_left/blink_right/brake depending on; lightStatus state; calls led_out and delay_led
137
138
139
140
141
       ............
      main loop:
          ldi ledState, 0b1100_0011
                                            ; set default LED state
144
          sbrc lightStatus , BRAKES
145
               rcall brake
```

```
146
        sbrc lightStatus, TURN_LEFT
        rcall blink_left
sbrc lightStatus, TURN_RIGHT
rcall blink_right
148
149
150
152
153
        rcall led_out rcall delay_led
154
        rjmp main_loop
156
    ; clears left side of ledState, rotate counter bit in range; 4, 5, 6, 7 and adds it to ledState
158
      blink_left:
cbr ledState, 0xF0
160
        or ledState, counter clc
ls1 counter
162
                                          ; clear carry flag
164
165
166
                                          ; if carry is clear jump ; to end_left
        brcc end_left
        ldi counter, COUNTER_LEFT_RESET
168
                                          :else reset counter to
169
                                          ; starting value
170
        end_left:
172
            ret
173
174
    ; clears right side of ledState, rotate counter bit in range ; 3,\ 2,\ 1,\ 0 and adds it to ledState
176
177
178
      ......
     blink_right:
        cbr ledState, 0x0F
or ledState, counter
lsr counter
cpi counter, 1
179
180
181
182
                                          ; if counter >=1 jump to ; end_right
183
        brge end_right
185
        ldi counter, COUNTER_RIGHT_RESET
                                          ; else reset counter to
187
                                           ; starting value
188
189
        end\_right:
190
191
    193
195
     brake:
        ser ledstate
197
198
199
    ;>>>>>>>; outputs complement of ledstates current value to PORTB ;
201
203
     led_out:
        mov complement, ledState
com complement
out PORTB, complement
205
206
207
        ret
209
    210
211
     ; 500 ms delay to show the LED output
212
213
     ; Generated by delay loop calculator ; at http://www.bretmulvey.com/avrdelay.html
214
215
    ; Delay 500 000 cycles
     216
217
    delay_led:
    ldi r31, 3
    ldi r30, 138
    ldi r29, 86
218
219
220
    L1: dec r29
222
223
        brne L1
224
        dec
        brne L1
226
227
        dec r31
brne L1
228
        rjmp PC+1
230
231
232
        ·>>>>>
     ; 100 ms delay to avoid bouncing
    ; Generated by delay loop calculator ; at http://www.bretmulvey.com/avrdelay.html
234
235
236
237
238
     ; Delay 100 000 cycles
; 100ms at 1 MHz
239
240
      </r/>
     delay switch:
        ldi r30, 130
ldi r29, 222
    L2: dec r29
brne L2
245
246
        dec r30
brne L2
```

```
247
248
249
250
251
252
253
254
255
256
257
258
259
      ; calls delay, break if turn left is active otherwise toogle; lightStatus TURN_RIGHT bit ON/OFF and reset counter
      :<<<>><>>
interrupt_right:
reall delay_switch
          sbrc lightStatus , TURN_LEFT
    reti
          cbr lightStatus, 0b1000_0000
ldi xorComparison, 0b0000_0001
eor lightStatus, xorComparison
ldi counter, COUNTER_RIGHT_RESET
260
261
262
263
264
265
266
267
268
270
271
272
273
274
275
                                                      ;RIGHT_TURN bit toogle
                                                      ; counter = starting bit string
      ; calls delay, break if turn right is active otherwise toogle
; lightStatus TURN_LEFT bit ON/OFF and reset counter
       interrupt_left:
    reall delay_switch
           {\bf sbrc} lightStatus , TURN_RIGHT
               reti
276
277
          cbr lightStatus , 0b0000_0001
ldi xorComparison , 0b1000_0000
eor lightStatus , xorComparison
278
279
                                                      ;LEFT_TURN bit toogle
           ldi counter, COUNTER_LEFT_RESET
                                                      ; counter = starting bit string
282
283
      284
285
286
287
288
          ldi xorComparison , 0b0000_0100
eor lightStatus , xorComparison
reti
290
                                                      ;BRAKES bit toogle
```

14