Report

Laboratory 2

September 20, 2017

Author:
Caroline Nilsson *(cn222nd)*
Daniel Alm Grundström *(dg222dw)*
Term: HT 2017
Course: 1DT301 - Computer Technology I

Contents

1 Assignment 1

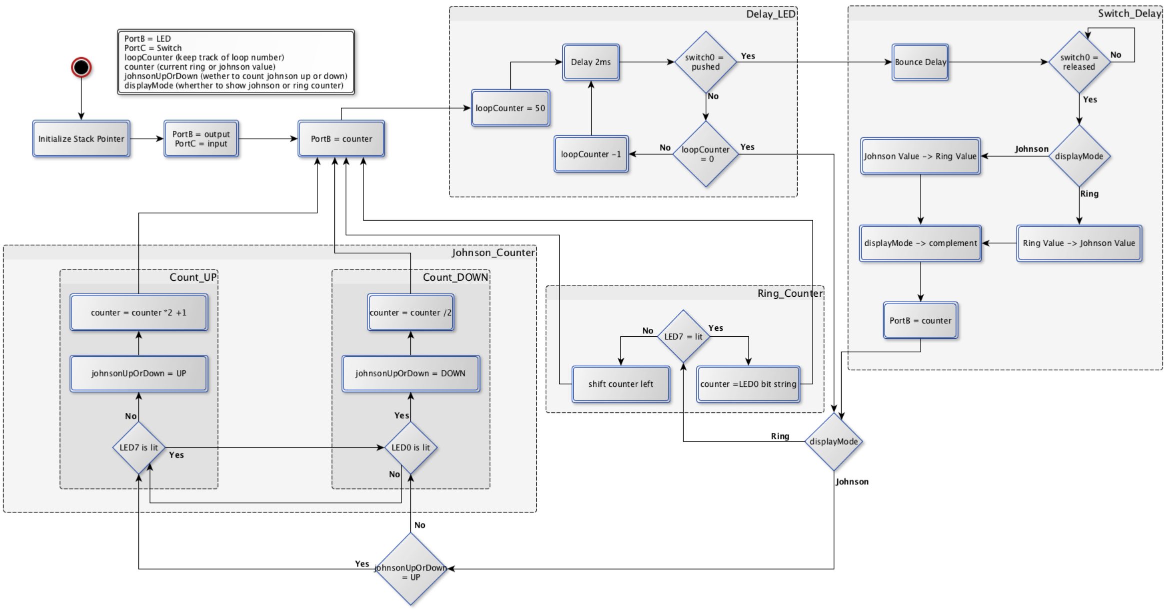

Figure 1: Switch between Johnson and Ring counter using switch0

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;   1DT301, Computer Technology I
;   Date: 2017-09-19
;   Author:
;                           Caroline Nilsson            (cn222nd)
;                           Daniel Alm Grundström       (dg222dw)
;
;   Lab number:             2
;   Title:                  Subroutines
;
;   Hardware:               STK600, CPU ATmega2560
;
;   Function:               Counts up a counter and display it's value as either a
;                           Ring counter or Johnson counter. The display mode can
;                           be toggled between ring/johnson by pressing switch SW0.
;
;   Input ports:            PIN0 on PORTC
;
;   Output ports:           PIN2 on PORTB
;
;   Subroutines:            led_out           - Outputs counter to LEDs
;                           delay_led         - Delay to make changes to LEDs
;                                               visible. Also continuously checks
;                                               if switch is pressed.
;                           ring_counter      - Counts up ring counter
;                           johnson_counter   - Counts johnson counter up/down
;                           check_switch      - Checks if switch gets pressed
;                           on_switch_pressed - Handles what should happen when
;                                               switch gets pressed
;                           delay_short       - Short delay of 2 ms used between
;                                               switch checks
;                           delay_switch      - Delay of 10 ms used after switch is
;                                               pressed down
;
;   Included files:         m2560def.inc
;
;   Other information:      N/A
;
;   Changes in program:
;                           2017-09-14:
;                           Implements flowchart design.
;
;                           2017-09-19:
;                           Refactors code by breaking down large subroutines
;                           into smaller subroutines. Fixes bug that caused
;                           counter to display wrong value when toggling display
;                           mode.
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

.def displayMode = r16              ;determines whether to output ring or johnson
.def counter = r17                  ;keeps track of output value
.def dataDir = r18                  ;use to set input and output on PORTs
.def loopCounter = r19              ;counts number of loops in delay led
.def johnUpOrDown = r20             ;whether to count johnson value up or down
```

```
.def complement = r21                   ;temp, to output counters complement
.equ UP = 0x01                          ;constant: value of up
.equ DOWN = 0x00                        ;constant: value of down
.equ JOHNSON = 0x00                     ;constant: Johnson display mode
.equ RING = 0xFF                        ;constant: Ring display mode
.equ SWITCH = PINC0                     ;constant: PIN of switch to check

;Initialize stack pointer
ldi r18, HIGH(RAMEND)
out SPH, r18
ldi r18, LOW(RAMEND)
out SPL, r18

;set PORTB to output
ldi dataDir, 0xFF
out DDRB, dataDir

;set PORTC to input
ldi dataDir, 0x00
out DDRC, dataDir

;initialize starting state
ldi displayMode, JOHNSON
ldi counter, 0x01
ldi johnUpOrDown, UP

main_loop:
    cpi displayMode, JOHNSON            ;if displaymode = johnson
        breq johnson1                   ;then jump to johnson branch

    ring1:                               ;else jump to ring
        rcall ring_counter
        rjmp main_loop

    johnson1:
        rcall johnson_counter

    rjmp main_loop

;Outputs complement of the current value of counter to LEDs
led_out:
    mov complement, counter
    com complement
    out PORTB, complement
    ret

;Delay with continuous switch checking
delay_led:
    ldi loopCounter, 50

    loop_led:
        rcall delay_short
        rcall check_switch

        cpi loopCounter, 0              ;if loopcounter = 0
        breq delay_led_end              ;then jump to end

        dec loopCounter
        rjmp loop_led

    delay_led_end:
        ret

;Creates the ring counter by writing the complement of counter
;to PORTB and then increments the ring counter
ring_counter:
    sbis PORTB, PINB7                   ;if the 7th led is lit
        ldi counter, 0x01               ;then set counter to one

    sbic PORTB, PINB7                   ;else
        lsl counter
                                ;shift counter to the left
    rcall led_out
    rcall delay_led

    ret

;Creates the johnson counter by writing the complement of counter
;to PORTB and then checks wheter to count up or down
johnson_counter:
    cpi johnUpOrDown, UP                ;if count up is active
    breq count_up                       ;then jump to count up

    rjmp count_down                     ;else jump to count down

    ;checks whether to continue to count up and
    ;increments the johnson value
    count_up:
        sbis PORTB, PINB7               ;if the 7th led is lit
            rjmp count_down             ;then jump to count down

        ldi johnUpOrDown, UP
        lsl counter                     ;shift to the left
        inc counter
                                        ;add one

        rjmp end

    ;checks whether to continue to count down and
    ;decrese the johnson value
    count_down:
```

```
158             sbic PORTB, PINB0               ;if the right most led is not lit
159                 rjmp count_up               ;then jump to count up
160
161             ldi johnUpOrDown, DOWN
162             lsr counter
163                             ;shift to the right
164
165         end:
166             rcall led_out
167             rcall delay_led
168
169             ret
170
171
172     ;Checks if the switch is pressed and in that case calls on_switch_pressed
173     check_switch:
174         sbic PINC, SWITCH                   ;if switch is not pressed
175             rjmp check_switch_end           ;then jump to end of subroutine
176
177         switch_pressed_down:
178             rcall delay_switch
179
180             ;wait until button is released
181             loop_switch:
182                 sbis PINC, SWITCH           ;if switch to the right most is still pressed
183                 rjmp loop_switch            ;then jump to loop switch
184
185             ;When the button has been released we consider the switch pressed
186             rcall on_switch_pressed
187
188         check_switch_end:
189         ret
190
191
192     ;Handles what should happen when the switch gets pressed
193     on_switch_pressed:
194         cpi displayMode, JOHNSON            ;if displaymode = johnson
195         breq johnson_to_ring                ;then jump to johnson to ring
196
197         cpi displayMode, RING               ;if displaymode = ring
198         breq ring_to_johnson                ;then jump to ring to johnson
199
200         ;convert ring value to johnson value
201         ring_to_johnson:
202
203             lsl counter
204             dec counter
205
206             rjmp switch_end
207
208         ;convert johnson value to ring value
209         johnson_to_ring:
210             lsr counter
211             inc counter
212
213         switch_end:
214             com displayMode
215             ;TODO: needed? rcall led_out
216             rcall led_out
217                                     ;toogle displaymode between ring and johnson
218         ret
219
220     ;Delay for 2 ms
221     delay_short:
222         ldi  r31, 13
223         ldi  r30, 252
224         L1:
225             dec  r30
226         brne L1
227         dec  r31
228         brne L1
229         nop
230
231         ret
232
233     ;Delay 10 ms to avoid bouncing when switch is pressed
234     delay_switch:
235         ldi  r31, 13
236         ldi  r30, 252
237     L2: dec  r30
238         brne L2
239         dec  r31
240         brne L2
241         nop
242
243         ret
```

2 Assignment 2

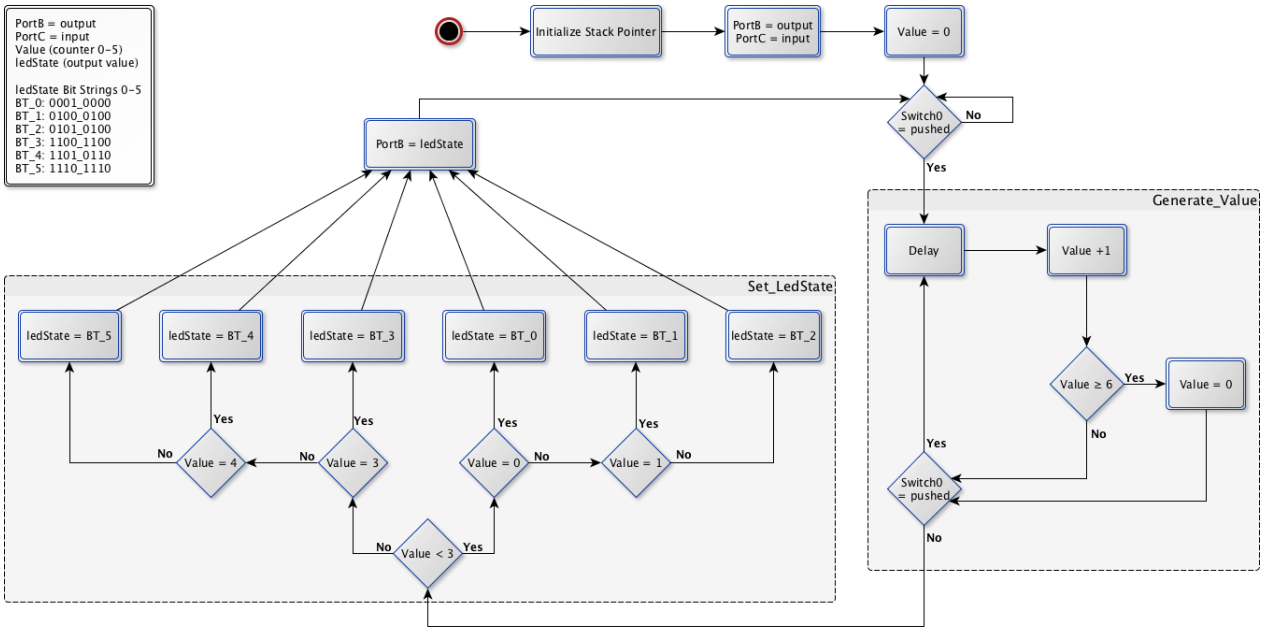

Figure 2: Simulating electronic dice

```
1   ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
2   ;   1DT301, Computer Technology I
3   ;   Date: 2017-09-19
4   ;   Author:
5   ;                       Caroline Nilsson            (cn222nd)
6   ;                       Daniel Alm Grundström       (dg222dw)
7   ;
8   ;   Lab number:         2
9   ;   Title:              Subroutines
10  ;
11  ;   Hardware:           STK600, CPU ATmega2560
12  ;
13  ;   Function:           Generates a random value between 1 and 6 when the user
14  ;                       presses down switch SW0 and, when the user releases
15  ;                       the switch, outputs a representation of a dice value
16  ;                       to the LEDs
17  ;
18  ;   Input ports:        PIN0 on PORTC
19  ;
20  ;   Output ports:       PORTB
21  ;
22  ;   Subroutines:        generate_value    - Generate a pseudorandom value
23  ;                                           between 1 and 6
24  ;                       set_led_state     - Set value to output to LEDs
25  ;                       delay_switch      - Delay of 10 ms used after switch is
26  ;                                           pressed down
27  ;
28  ;   Included files:     m2560def.inc
29  ;
30  ;   Other information:  N/A
31  ;
32  ;   Changes in program:
33  ;                       2017-09-14:
34  ;                       Implements flowchart design.
35  ;
36  ;                       2017-09-19:
37  ;                       Adds constants for LED dice states and comments. Adds
38  ;                       outputting of complement to make the correct LEDs light
39  ;
40  ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
41  .include "m2560def.inc"
42
43  .def dataDir = r16
44  .def randomValue = r17
45  .def ledState = r18
46  .def complement = r19
47
48  .equ LED_DICE_1 = 0b0001_0000
49  .equ LED_DICE_2 = 0b0100_0100
50  .equ LED_DICE_3 = 0b0101_0100
51  .equ LED_DICE_4 = 0b1100_1100
52  .equ LED_DICE_5 = 0b1101_0110
53  .equ LED_DICE_6 = 0b1110_1110
54
```

```
55  ldi r16, HIGH(RAMEND)
56  out SPH, r16
57  ldi r16, LOW(RAMEND)
58  out SPL, r16
59
60  ldi dataDir, 0xFF
61  out DDRB, dataDir
62
63  ldi dataDir, 0x00
64  out DDRC, dataDir
65
66  ldi complement, 0xFF
67  out PORTB, complement
68
69  loop:
70      sbic PINC, PINC0                    ;Wait until switch is pressed down
71          rjmp loop
72
73      rcall generate_value
74      rcall set_led_state
75      mov complement, ledState
76      com complement
77      out PORTB, complement
78
79      rjmp loop
80
81  ;Generate a pseudorandom value by repeatedly incrementing a counter for as long
82  ;as the switch is pressed down
83  generate_value:
84      ldi ledState, 0xFF                  ;Reset LEDs
85      out PORTB, ledState
86      start:
87
88          rcall delay_switch              ;Delay to avoid bouncing effects
89
90          inc randomValue
91          cpi randomValue, 6
92          brge reset_value
93          rjmp end
94
95      reset_value:
96          ldi randomValue, 0
97
98      end:
99          sbis PINC, PINC0                ;If switch is still pressed down
100             rjmp start                  ;   then jump to start
101
102         ret
103
104 ;Set LED output value to bit pattern representing different dice values
105 ;depending of value of the pseudorandomly generated value
106 set_led_state:
107     cpi randomValue, 3
108     brlo less
109     rjmp more
110
111     less:
112         cpi randomValue, 0
113         breq one
114
115         cpi randomValue, 1
116         breq two
117
118         rjmp three
119
120         one:
121             ldi ledState, LED_DICE_1
122             rjmp end_led_state
123
124         two:
125             ldi ledState, LED_DICE_2
126             rjmp end_led_state
127
128         three:
129             ldi ledState, LED_DICE_3
130             rjmp end_led_state
131
132     more:
133         cpi randomValue, 3
134         breq four
135
136         cpi randomValue, 4
137         breq five
138
139         rjmp six
140
141         four:
142             ldi ledState, LED_DICE_4
143             rjmp end_led_state
144
145         five:
146             ldi ledState, LED_DICE_5
147             rjmp end_led_state
148
149         six:
150             ldi ledState, LED_DICE_6
151             rjmp end_led_state
152
153     end_led_state:
154         ret
155
```

```
156 ;Delay 10 ms to avoid bouncing effects when a switch is first pressed down
157 delay_switch:
158     ldi   r31, 13
159     ldi   r30, 252
160     L1:
161         dec   r30
162     brne L1
163     dec   r31
164     brne L1
165     nop
166
167     ret
```

3 Assignment 3

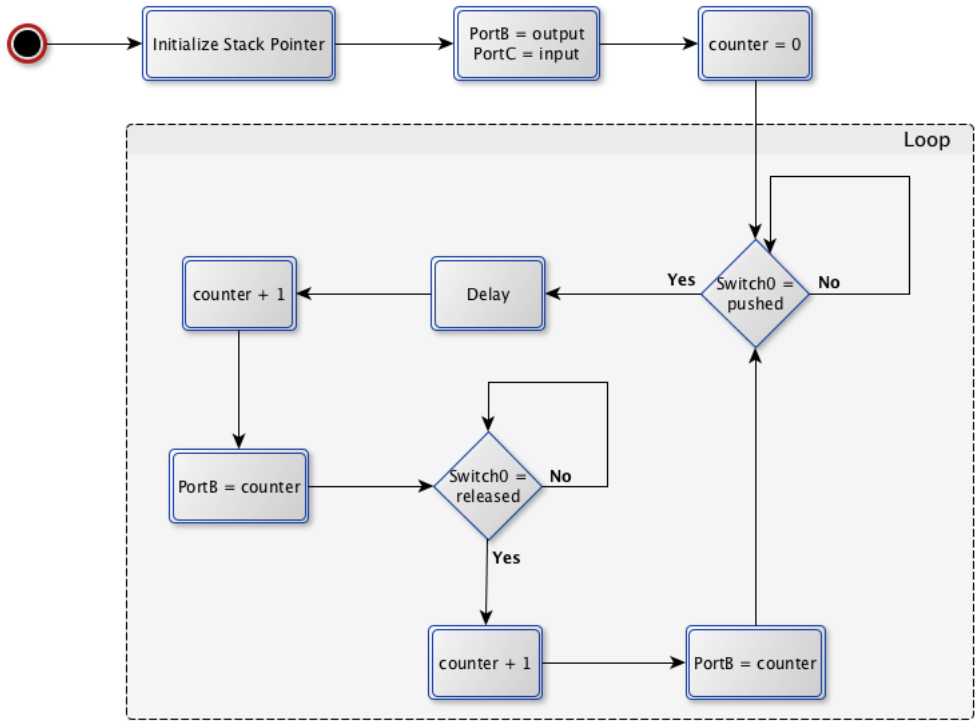

Figure 3: Change counter on switch0

```
1  ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
2  ;   1DT301, Computer Technology I
3  ;   Date: 2017-09-19
4  ;   Author:
5  ;                       Caroline Nilsson            (cn222nd)
6  ;                       Daniel Alm Grundström       (dg222dw)
7  ;
8  ;   Lab number:         2
9  ;   Title:              Subroutines
10 ;
11 ;   Hardware:           STK600, CPU ATmega2560
12 ;
13 ;   Function:           Counts number of times switch SW0 changes values, i.e.
14 ;                       how many times the switch goes from 0 to 1 and 1 to 0.
15 ;
16 ;                       The counter is outputted to the LEDs in binary form
17 ;                       each time the counter gets incremented.
18 ;
19 ;   Input ports:        PIN0 on PORTC
20 ;
21 ;   Output ports:       PORTB
22 ;
23 ;   Subroutines:        wait_for_switch_press   - Delays execution until SW0 is
24 ;                                                 pressed.
25 ;                       on_switch_down          - Handles what should happen
26 ;                                                 when SW0 gets pressed.
27 ;                       wait_for_switch_release - Delays execution until SW0 is
28 ;                                                 released.
29 ;                       on_switch_up            - Handles what should happen
30 ;                                                 when SW0 gets released.
31 ;                       led_out                 - Ouputs complement of counter
32 ;                                                 to LEDs.
33 ;                       delay_switch            - Delay 10 ms to avoid effects
34 ;                                                 of bouncing.
35 ;
36 ;   Included files:     m2560def.inc
37 ;
38 ;   Other information:  N/A
39 ;
40 ;   Changes in program:
41 ;                       2017-09-14:
42 ;                       Implements flowchart design.
43 ;
44 ;                       2017-09-19:
45 ;                       Refactors the code by breaking smaller subroutines into
46 ;                       multiple smaller ones. Adds comments
47 ;
48 ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

```
49  .include "m2560def.inc"
50
51  .def dataDir = r16
52  .def counter = r17
53  .def complement = r18
54
55  ldi r16, HIGH(RAMEND)
56  out SPH, r16
57  ldi r16, LOW(RAMEND)
58  out SPL, r16
59
60  ldi dataDir, 0xFF
61  out DDRB, dataDir
62
63  ldi dataDir, 0x00
64  out DDRC, dataDir
65
66  ldi counter, 0x00
67  rcall led_out
68
69  main_loop:
70      rcall wait_for_switch_press
71      rcall on_switch_down
72
73      rcall wait_for_switch_release
74      rcall on_switch_up
75
76      rjmp main_loop
77
78  ;Pauses execution of program until SW0 is pressed down
79  wait_for_switch_press:
80      loop:
81          sbic PINC, PINC0                ;If SW0 is not pressed down
82              rjmp loop                   ;   then continue waiting
83      ret                                 ;return when SW0 gets pressed down
84
85  ;Handles what should happen when SW0 gets pressed down
86  on_switch_down:
87      rcall delay_switch                  ;Delay to avoid bouncing effects
88      inc counter
89      rcall led_out                       ;Output new counter value
90      ret
91
92  ;Pauses execution of program until SW0 is released
93  wait_for_switch_release:
94      loop_2:
95          sbis PINC, PINC0                ;If SW0 is still pressed down
96              rjmp loop_2                 ;   then continue waiting
97      ret                                 ;return when SW0 gets released
98
99  ;Handles what should happen when SW0 gets released
100 on_switch_up:
101     inc counter
102     rcall led_out                       ;Output new counter value
103     ret
104
105 ;Outputs a binary representation of the current counter value to the LEDs
106 led_out:
107     mov complement, counter
108     com complement
109     out PORTB, complement
110
111 ;Delay of 10 ms. Used to avoid effects of switch bouncing
112 delay_switch:
113     ldi  r31, 13
114     ldi  r30, 252
115     L1:
116         dec  r30
117     brne L1
118     dec  r31
119     brne L1
120     nop
121
122     ret
```

4 Assignment 4

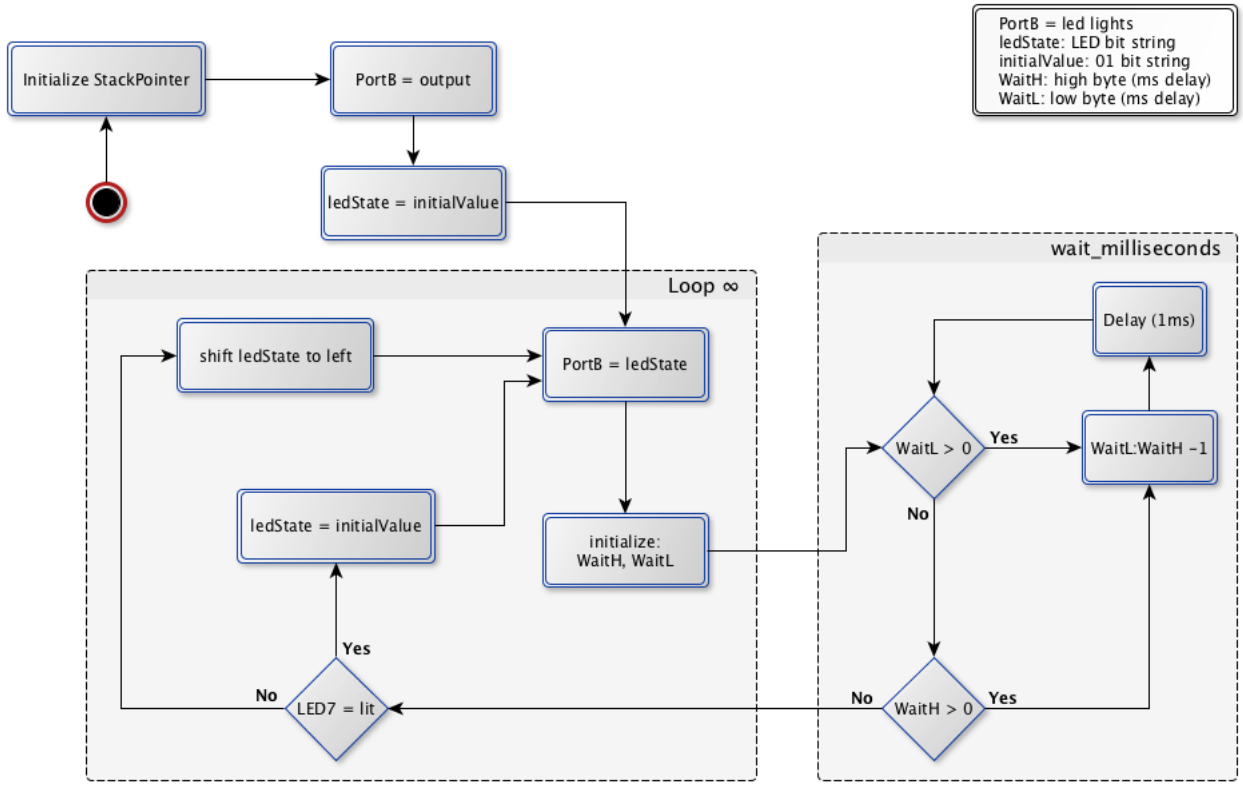

Figure 4: Ring counter with "wait-milliseconds" delay

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;   1DT301, Computer Technology I
;   Date: 2017-09-07
;   Author:
;                       Caroline Nilsson            (cn222nd)
;                       Daniel Alm Grundström       (dg222dw)
;
;   Lab number:         2
;   Title:              Subroutines
;
;   Hardware:           STK600, CPU ATmega2560
;
;   Function:           Repeatedly lights LEDs sequentially right to left.
;
;                       I.e:
;                       0000 0001 -> 0000 0010 -> 0000 0100 -> ... ->
;                       1000 0000 -> 0000 0001 -> 0000 0010 -> ...
;
;   Input ports:        N/A
;
;   Output ports:       PORTB
;
;   Subroutines:        wait_milliseconds - Delays executions n milliseconds.
;   Included files:     m2560def.inc
;
;   Other information:  N/A
;
;   Changes in program:
;                       2017-09-14:
;                       Implements flowchart design
;
;                       2017-09-19:
;                       Adds header, comments and some minor refactoring
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"
.def dataDir = r16
.def ledState = r17
.def complement = r18
.def waitH = r25
.def waitL = r24
.equ INITIAL_LED_STATE = 0x01

; Initialize SP, Stack Pointer
ldi r20, HIGH(RAMEND)                   ; R20 = high part of RAMEND address
```

```
46  out SPH,R20                         ; SPH = high part of RAMEND address
47  ldi R20, low(RAMEND)                ; R20 = low part of RAMEND address
48  out SPL,R20                         ; SPL = low part of RAMEND address
49
50  ; Set PORTB to output
51  ldi dataDir, 0xFF
52  out DDRB, dataDir
53
54  ldi ledState, INITIAL_LED_STATE     ;Set initial LED state
55
56  loop1:
57      mov complement, ledState
58      com complement
59      out PORTB, complement           ;Write complement of LED state to LEDs
60
61      ldi waitH, HIGH(1000)
62      ldi waitL, LOW(1000)
63      rcall wait_milliseconds         ;Delay to make changes visible
64
65      sbis PORTB, PINB7               ;If leftmost LED is lit
66          ldi ledstate, INITIAL_LED_STATE ;   then reset LED State
67      sbic PORTB, PINB7               ;Else
68          lsl ledState                ;   Shift LED state to the left
69      rjmp loop1
70
71  ;Wait n milliseconds. The number of milliseconds to wait is provided through
72  ;registers 25:24.
73  wait_milliseconds:
74      loop2:
75          cpi waitL, 0x00             ;If lower bit of register pair 'wait' is 0
76          breq low_zero               ;   then jump to low_zero
77          rjmp wait                   ;Else jump to wait
78
79      low_zero:
80          cpi waitH, 0x00             ;If higher bit of register pair 'wait' is 0
81          breq high_zero              ;   then jump to high_zero
82          rjmp wait                   ;Else jumpt to wait
83
84      high_zero:
85          ret
86
87      wait:
88          sbiw waitH:waitL, 0x01      ;Decrement register pair 'wait'
89
90          ;Delay 1 ms
91          ldi  r20, 2
92          ldi  r19, 74
93      L1: dec  r19
94          brne L1
95          dec  r20
96          brne L1
97          rjmp PC+1
98
99          rjmp loop2
```