Laboratory 2 Report

September 25, 2019

Authors: Andrei Neagu
Konstantinos Tatsis

Course code: 1DT301

Contents

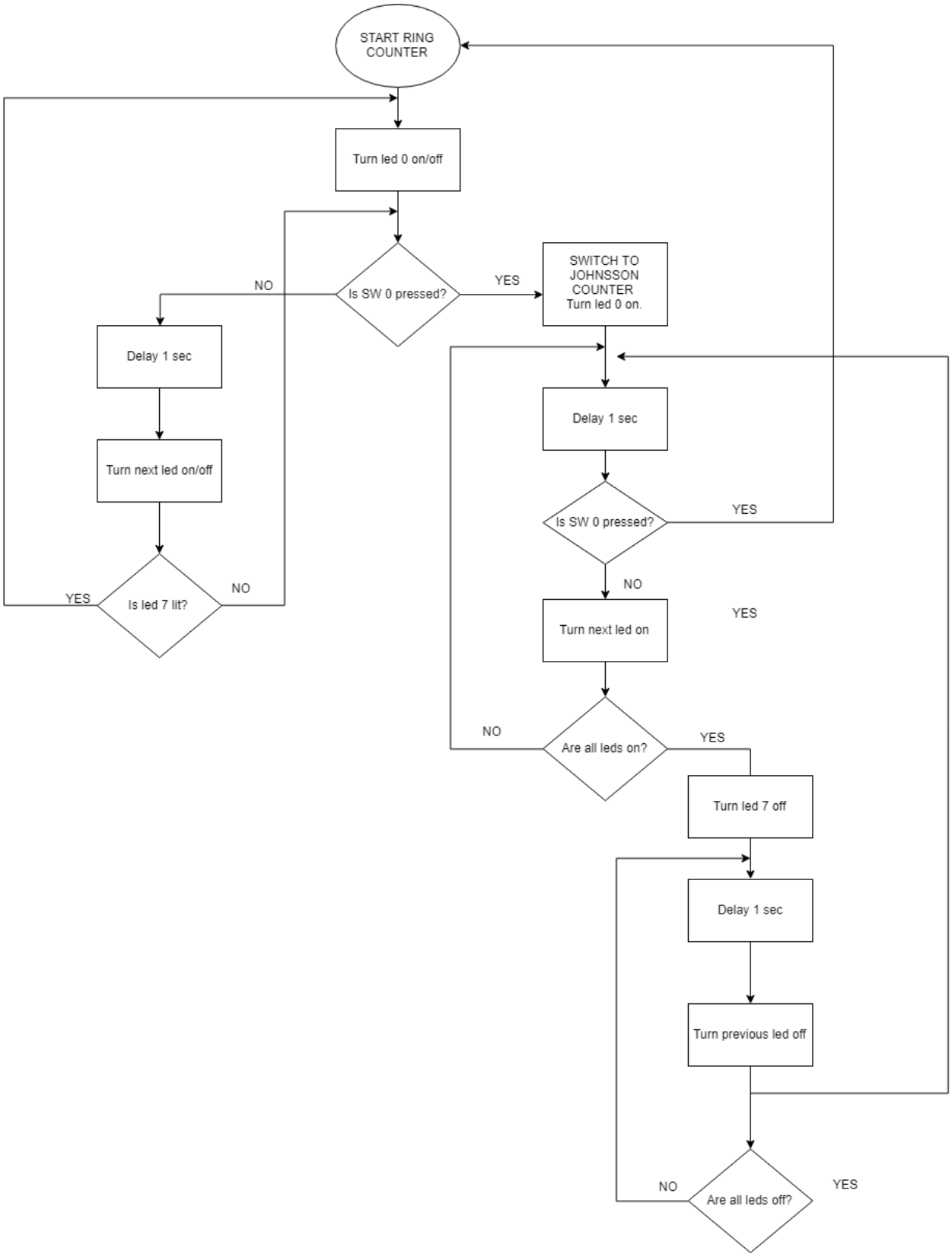
1 Task 1

Switch –Ring counter / Johnson counterWrite a program which switch between Ring counter and Johnson counter. You should not use Interruptin this lab. 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.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;    1DT301, Computer Technology I
;    Date: 2019-09-22
;    Author: Andrei Neagu (an223kj)
;            Konstantinos Tatsis (kt222iq)
;
;    Lab number:          2
;    Title:               Switch - Ring counter / Johnson counter
;
;    Hardware:            STK600, CPU ATmega2560
;
;    Function:            Switches between a ring counter and johnson counter
;
;    Input ports:                 PORTA
;
;    Output ports:        PORTB
;
;    Subroutines:              delay, loop, ring_counter, johnson_counter,
;                               switch_ring, switch_johnson
;
;    Included files:      m2560def.inc
;
;    Other information:
;
;    Changes in program: 2019-09-22
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"

ldi r21, HIGH(RAMEND)           ;initialize the stack pointer
out SPH,R21
ldi R21, low(RAMEND)
out SPL,R21

ldi r16, 0xFF                          ; set the DDRB as output
out DDRB, r16

ldi r20, 0b11111110                    ; register used to check if SW0 is pressed

loop:
        rcall switch_johnson    ; if SW0 is pressed switch to johnson

ring_counter:
        ldi r18, 0b11111110            ; the program starts with the ring counter
        call ring_loop

ring_loop:
        out PORTB, r18                 ; LED0 lights up
        call Delay                             ; delay of 500ms
        com r18
        LSL r18                                ; shift the bits to left
        com r18

        ldi r24,0xFF
        cp r24, r18                            ; if the lights reach the end start again with the ring
        breq ring_counter

        rcall switch_johnson    ; check if SW0 is pressed and go to johnson
        rjmp ring_loop

rjmp loop                                      ; if this is reached go to the start again

johnson_counter :
```

```
        ldi r19, 0b1111_1110
        ldi r22, 0b0000_0000

johnson_loop:
        out PORTB, r19                  ; light first led
        LSL r19                                 ; shift the bits to left
        call Delay                              ; delay of 500ms
        cp r19, r22                             ; if the end was reached go to forward
        breq johnson_forward

        rcall switch_ring               ; if SW0 is pressed switch to ring
        rjmp johnson_loop               ; if this is reached go to the start of johnson

johnson_forward :                       ; this is the johnson main logic for the second loop
        out PORTB, r22
        ldi r22, 0b11111111
        call Delay
        ldi r19,0b10000000

        johnson_secondary:
                out PORTB, r19
                ASR r19                         ; switch the bits to the right
                call Delay
                cp r19, r22
                breq johnson_counter

                rcall switch_ring               ; if SW0 is pressed switch to ring
                rjmp johnson_secondary  ; go back to the secondary loop

 switch_johnson:                                        ; checks if SW0 is pressed and if so goes to
                                                ;johnson_counter
                in r16, PINA
                cp r20,r16
                breq johnson_counter
                ret

 switch_ring:                                   ; checks if SW0 is pressed and if so goes to ring_counter
                in r16, PINA
                cp r20,r16
                breq ring_counter
                ret

 Delay :                                                ;this delay is approx 500ms
        ldi  r21, 5
    ldi  r23, 15
    ldi  r24, 242
L1: dec  r24
    brne L1
    dec  r23
    brne L1
    dec  r21
    brne L1
        ret
```

1.1 Flowchart 1



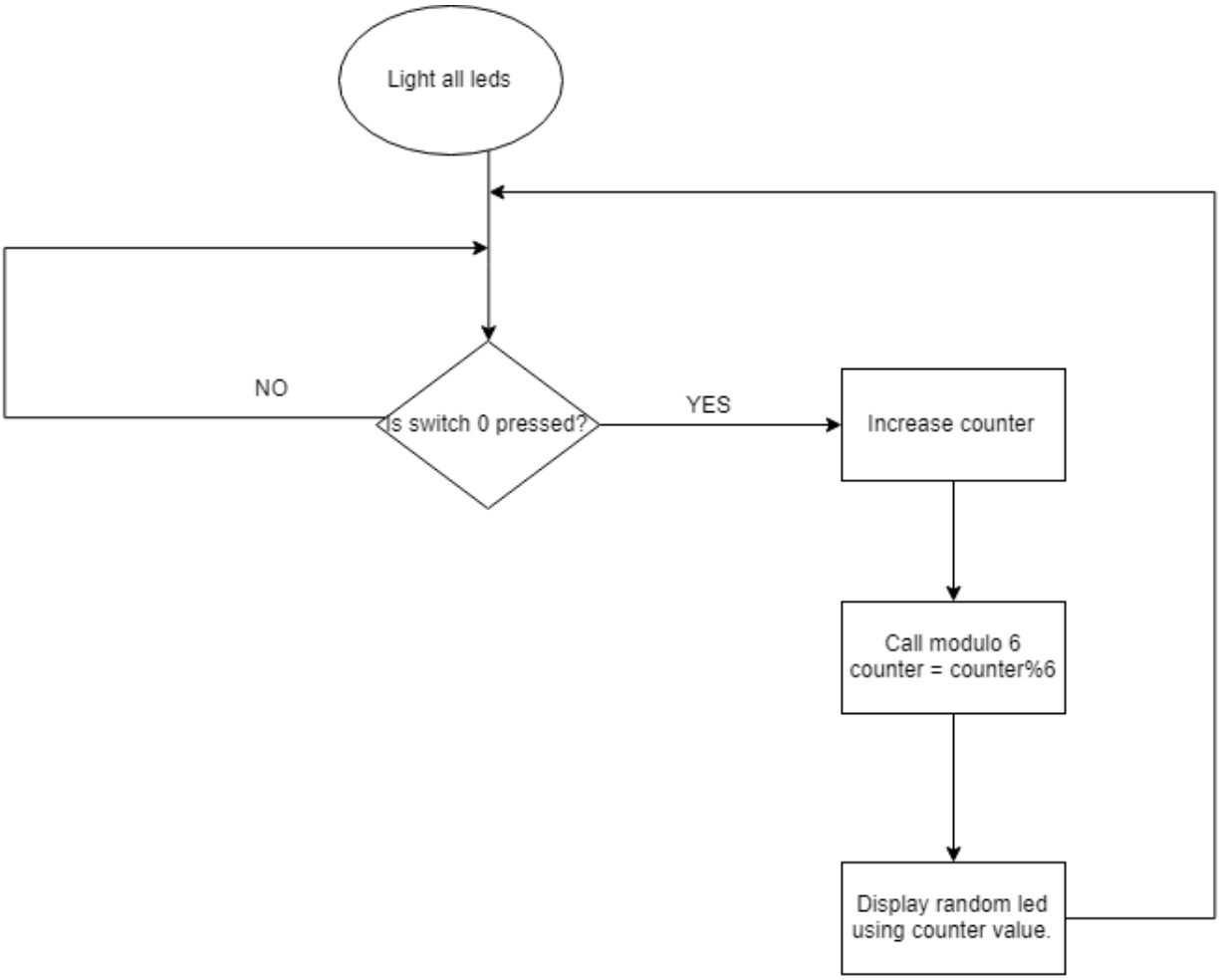
2 Task 2

Electronic dice.You should create an electronic dice. Think of the LEDs placed as in the picture below. The number 1 to 6 should be generated randomly. You could use the fact that the time you press the button varies in length.

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;    1DT301, Computer Technology I
;    Date: 2019-09-21 14:20:40
;    Author: Andrei Neagu (an223kj)
;            Konstantinos Tatsis (kt222iq)
;
;    Lab number:         2
;    Title:              Electronic Dice
;
;    Hardware:           STK600, CPU ATmega2560
;
;    Function:           Lights the led in a random generated way when you press switch (0).
;
;    Input ports:                PORTA
;
;    Output ports:       PORTB
;
;    Subroutines:                Display, Button, one, two..
;
;    Included files:     m2560def.inc
;
;    Other information: -
;
;    Changes in program: 2019-09-21 14:20:40
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

ldi r20, 0b1111_1111 ; Load to r20 the binary value
out DDRB, r20 ; Output the loaded value to PORTB

ldi r23,0b0000_0001 ;load to r23 the binary value

; Dice loop
loop:
in r22, PINA ; Read PINA
cpi r22, 0b1111_1110 ; IF PINA equals to " 0b1111_1110"
breq button ; Then branch to button subroutine
rjmp loop;


;PRESSING SWITCH
button:
inc r23 ; it increases the value in r23 --> 0b0000_0001 which is 1 in decimal
cpi r23, 7 ; if the load value in r23 is 7 then
breq modulo6 ; branch to subroutine modulo6

check:
in r22, PINA ; read PORTA and write to r22
cpi r22, 0b1111_1111 ; IF r22 equals to "0b1111_1111"
breq display ; then branch to display subroutine

rjmp button

modulo6:
ldi r23, 0b0000_0001  ; load to r23 the binary value of 1
rjmp check  ; jump to check subroutine


;OUTPUT RESULT
display:
cpi r23, 1 ; if r23 equals to 1
breq one ; go to subroutine one

cpi r23, 2 ; same as subroutine "one"
breq two

cpi r23, 3
breq three

cpi r23, 4
```

```
breq four

cpi r23, 5
breq five

cpi r23, 6
breq six


; DISPLAY RESULT
one:
ldi r21, 0b0001_0000 ; Display random value 1
out DDRB, r21 ;  Output the loaded value to PORTA
rjmp loop

two:
ldi r21, 0b1001_1000 ; Display random value 2
out DDRB, r21 ; Output the loaded value to PORTA
rjmp loop

three:
ldi r21, 0b0011_0000 ; etc
out DDRB, r21
rjmp loop

four:
ldi r21, 0b1111_1000
out DDRB, r21
rjmp loop

five:
ldi r21, 0b0001_1100
out DDRB, r21
rjmp loop

six:
ldi r21, 0b1001_0010
out DDRB, r21
rjmp loop
```

2.1 Flowchart 2

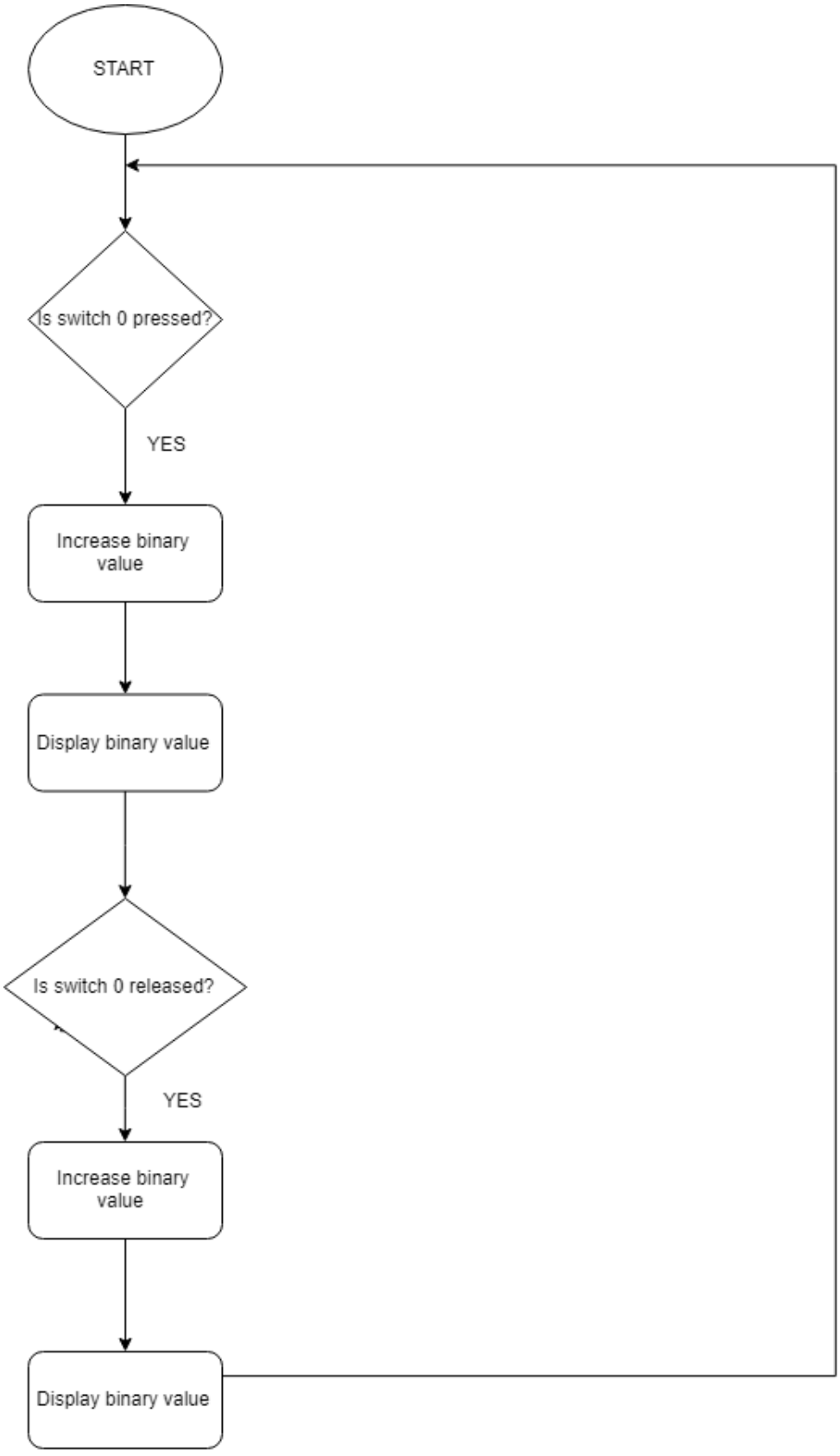


3 Task 3

Write a program that is able to count the number of changes on a switch. As a change we count when the switch SW0 goes from 0 to1 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.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;   1DT301, Computer Technology I
;   Date: 2019-09-22 16:47:08
;   Author: Andrei Neagu (an223kj)
;           Konstantinos Tatsis (kt222iq)
;
;   Lab number:         2
;   Title:              Change Counter
;
;   Hardware:           STK600, CPU ATmega2560
;
;   Function:                   Displays and increases the binary value every time led 0 is pressed on/off
;
;   Input ports:                PORTA
;
;   Output ports:       PORTB
;
;   Subroutines:        Activated, main
;
;   Included files:     m2560def.inc
;
;   Other information:  Ring counter with a delay of 500ms
;
;   Changes in program: 2019-09-22
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


.include "m2560def.inc"

; Initialize Input/Output
ldi r20, 0b0000_0000 ;load 0 (Decimal value) to r20
out DDRA, r20 ; Set PORTA as an input with the value of r20
ldi r20, 0b1111_1111  ;load 255 (Decimal value) to r20
out DDRB, r20 ; Set PORTB as an input with the value of r20
ldi r25, 0b0000_0000 ;load 0 (Decimal value) to r25

;Move to main subroutine check switch
main:
        in r24, PINA  ; Read from PINA and write it to r24
        cpi r24, 0b1111_1110 ; IF r24 equals that binary value "0b1111_1110" then
        breq activated ; branch to pressed subroutine
rjmp main

activated:
        inc r25 ; increase r25 by 1 decimal value
        com r25 ; complement/ flip the value of r25
        out PORTB, r25 ; Output the value of r25 to PORTB
        com r25 ; complement/ flip the value of r25

loop:
        in r24, PINA ; Read from PINA and write it to r24
        cpi r24, 0b1111_1111  ; IF r24 not equals that binary value "0b1111_1111" then
        brne loop ; branch to loop subroutine

        inc r25 ; increase r25 by 1 decimal value
        com r25  ; complement/ flip the value of r25
        out PORTB, r25
        com r25
rjmp main ; jump back to main
```

3.1 Flowchart 3



4 Task 4

Modify the program in task 5 in Lab 1to 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.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;    1DT301, Computer Technology I
;    Date: 2019-09-22
;    Author: Andrei Neagu (an223kj)
;            Konstantinos Tatsis (kt222iq)
;
;    Lab number:         2
;    Title:              Learn how to program and use subroutines with Assembly Language
;
;    Hardware:           STK600, CPU ATmega2560
;
;    Function:           Delay subroutine with variable delay time
;
;    Input ports:
;
;    Output ports:       PORTB
;
;    Subroutines:        delay, wait_miliseconds
;
;    Included files:     m2560def.inc
;
;    Other information:  Ring counter with a delay of 500ms
;
;    Changes in program:
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"


; INITIALIZE STACK
ldi r18, HIGH (RAMEND)
out SPH, r18
ldi r18, LOW (RAMEND)
out SPL, r18

; Set DDRB as output
ldi r17, 0b1111_1111
out DDRB, r17

ldi r16, 0b1111_1111                    ; r16 is used in the ring counter

main:
        ring:                   ; 1000 fits in 2 8bit registers
                ldi r24, LOW(1000)      ; load the low part of 1000 in r24
                ldi r25, HIGH(1000) ; load the high part of 1000 in r25
                cpi r16, 0b1111_1111    ; if equal go to again
                breq again

                out PORTB, r16

                com r16
                lsl r16                         ; shift the bits to the left
                com r16

                rcall wait_milliseconds ; call the subroutine with r25:24
    rjmp ring

        again:
        ldi r16, 0b1111_1110    ; this is used when the counter has reached the end

rjmp main

wait_milliseconds:


delay:

    ldi  r18, 2
    ldi  r19, 75
L1: dec  r19
```

```
    brne L1
    dec  r18
    brne L1
    rjmp PC+1


        sbiw r25:r24,1          ; subtract 1 from r25:24,compare and if not equal then branch to delay
        brne delay

RET
```

4.1 Flowchart 4