Linnéuniversitetet
Kalmar Växjö

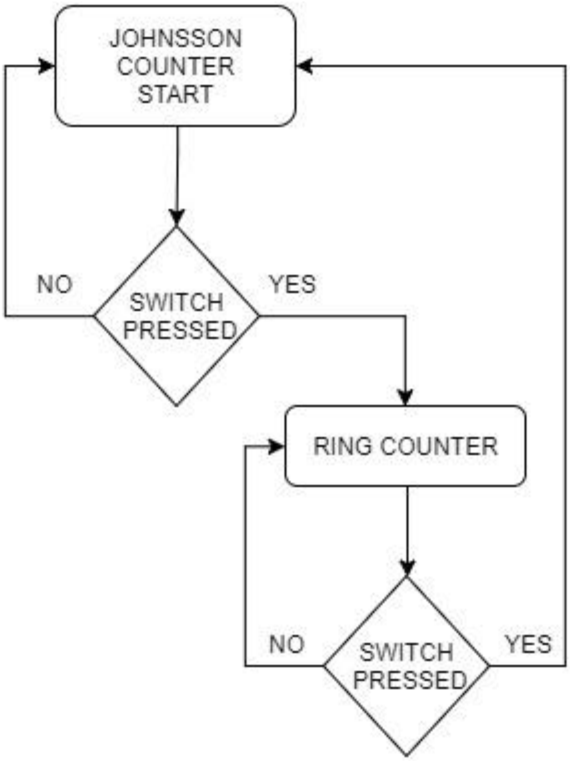
Linnaeus University

1DT301 - Computer Technology
Laboration 2

Students:
Alexander Risteski - ar222yu@student.lnu.se
Dimitrios Argyriou - da222kf@student.lnu.se

Task 1

Switch –Ring counter / Johnson counter
Write a program which switch between Ring counter and Johnson counter.
You should not use Interrupt in this lab. The push button 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: 08/09/2017
;           Authors:
;                                   Alexander Risteski
;                                   Dimitrios Argyriou
;
;           Hardware: STK600, CPU ATmega2560
;
;           Function: This program switches between Ring counter and Johnson counter
;           with the use of switch 0.
;
;           Input ports: On-board switches connected to PORTA.
;           Output ports: On-board LEDs connected to PORTB.
```

```
;
;                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
; Assigning names to the registers
.DEF mr = r16
.DEF mri = r17
.DEF switch =r18
;-----------------------<<<Johnson counter>>>----------------------------------------

; Here a Johnson counter starts and in each of its subroutines a call to the subroutine switch1
happens in order to verify if a switch has been pressed, so that it will change to the Ring Counter.

JohnsonCounter:

ldi mr, 0xff
out DDRB, mr

forward:                                 ;

rcall switch1
out PORTB, mr
lsl mr
call delay
cpi mr, 0b00000000
brne forward
rjmp reset

reset:
rcall switch1
out PORTB,mr
call delay
ldi mri,0b10000000
rjmp backwards

backwards:

lsr mr
add mr, mri
rcall switch1
out PORTB,mr
call delay
cpi mr, 0xFF
brne backwards

rjmp JohnsonCounter

;-----------------------<<< ring cycle>>>---------------------------
```

;Similarly in Ring counter the switch1 is called in “myloop” to check if a switch is pressed, so that it changes back to Johnson counter.

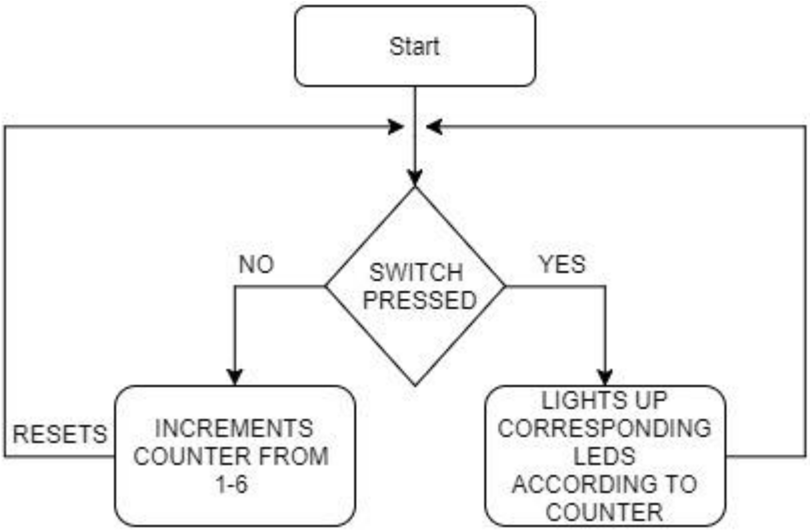
RingCounter:

```
start:
ldi mr, 0x01
com mr                  ; complements the rergister so that it will be showed correctly
out PORTB, mr
com mr                  ;complements again to return to its original form
rcall delay

myloop:

rcall switch1
lsl mr
com mr
out PORTB, mr
com mr

cpi mr, 0b00000000
breq equal
rcall delay
rjmp myloop

equal:
rjmp RingCounter



;-------------------methods-------------------------------------------------------
;Switch1 & 0 check if the input value is equal to change from one counter to another
;Switch 7 was used due to a problem on the board
switch0:
in r21, PIND
ldi r22, 0b01111111
cp r21, r22
breq buttonY
ret

buttonY:
rjmp JohnsonCounter

switch1:
in r23, PIND
ldi r24, 0b01111111
cp r23, r24
breq buttonY2
ret

buttonY2:
rjmp RingCounter
```

```
delay:
ldi  r18, 5
ldi  r19, 15
ldi  r20, 242
L1: dec  r20
brne L1
rcall switch0
dec  r19
brne L1
rcall switch0
dec  r18
brne L1
rcall switch0
ret
```

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.



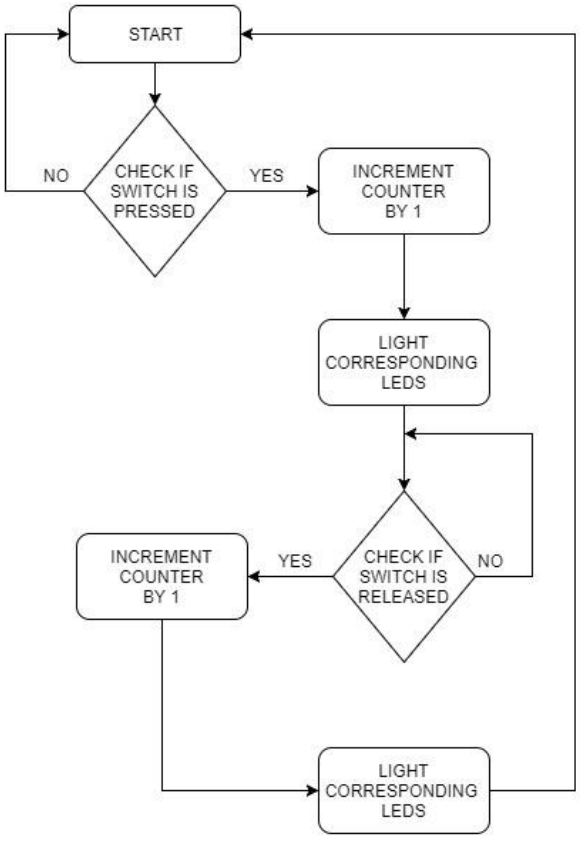

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;           1DT301, Computer Technology I
;           Date: 08/09/2017
;           Authors:
;                               Alexander Risteski
;                               Dimitrios Argyriou
;
;           Hardware: STK600, CPU ATmega2560
;
;           Function: Reads the switches and lights the corresponding LED.
;
;           Input ports: On-board switches connected to PORTA.
;           Output ports: On-board LEDs connected to PORTB.
;
;           Included files: m2560def.inc
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"

 .DEF mr = r16
 .DEF mr2 = r17
 .DEF counter = r22

clr counter
ldi mr,0xff
out DDRB,mr


start:
in r18, PINA
out PORTB,mr
```

```
cpi r18,0xfe            ; a loop that while the button is not pressed
breq pressed            ; runs and increases the counter.
rjmp notPressed         ; If it is pressed it branches to pressed

rjmp start


;<<<<< Increment counter if button is pressed >>>>>>

notPressed:
inc counter             ; increments counter by 1
cpi counter, 7          ; if it is 7
brge reset              ;or more than 7, it resets, otherwise
rjmp start              ; jumps to start

reset:
clr counter             ;clears counter
inc counter             ;increments counter by 1
rjmp start

;<<<<<<<     Check what number is the counter >>>>>>>>>>

pressed:
rcall delay             ;calls delay because without it was not showing the result “nicely”
cpi counter, 1
breq isOne              ; if it is one it will branch to isOne which will light up
brne checkIftwo         ; the corresponding leds. Likewise it will happen for every number up to
                        ; 6
checkIftwo:
cpi counter, 2
breq isTwo
brne checkIfthree

checkIfThree:
cpi counter, 3
breq isThree
brne checkIfFour

checkIfFour:
cpi counter, 4
breq isFour
brne checkIfFive

checkIfFive:
cpi counter, 5
breq isFive
brne checkIfSix

checkIfSix:
cpi counter, 6
breq isSix
rjmp start
```

```
;<<<<<<<   Light coresponding light    >>>>>>>>>>

isOne:
ldi mr,0b11111110          ; hex 1
out PORTB, mr
rjmp start

isTwo:
ldi mr,0b11111100          ; hex 2
out PORTB, mr
rjmp start

isThree:
ldi mr,0b11111000          ; hex 3
out PORTB, mr
rjmp start

isFour:
ldi mr,0b11110000          ; hex 4
out PORTB, mr
rjmp start

isFive:
ldi mr,0b11100000          ; hex 5
out PORTB, mr
rjmp start

isSix:
ldi mr,0b11000000          ; hex 6
out PORTB, mr
rjmp start

delay:

ldi  r18, 3
ldi  r19, 138
ldi  r20, 86
L1: dec  r20
brne L1
dec  r19
brne L1
dec  r18
brne L1
rjmp PC+1

                ret
```

Task 3

Change counter
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.

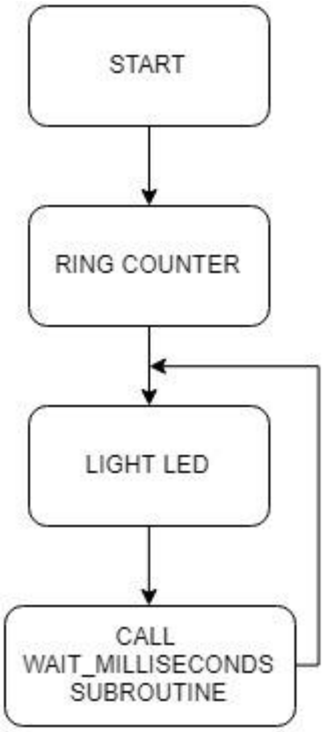


```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;		1DT301, Computer Technology I
;		Date: 08/09/2017
;		Authors:
;				Alexander Risteski
;				Dimitrios Argyriou
;
;		Hardware: STK600, CPU ATmega2560
;
;		Function: This program counts the number of changes on a switch from 0 to 1 and
;		from 1 to 0.
;
;		Input ports: On-board switches connected to PORTA.
;		Output ports: On-board LEDs connected to PORTB.
;
;		Included files: m2560def.inc
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<



.include "m2560def.inc"

 .DEF mr = r16
 .DEF mr2 = r17
 .DEF counter = r22

clr counter
ldi mr,0xff
out DDRB,mr
out PORTB, mr

start:
in mr2, PINA
cpi mr2,0b11111110	;branches to pressed when the SW0 is pressed
breq pressed
rjmp start



pressed:
inc counter		; increases the counter by 1
com counter		; complements the counter so that it will be showed correctly
rcall delay
out PORTB , counter
com counter		; complements again to return it to its original form
```

```
release:
in mr2, PINA            ; checks if the button is released so that it will branch to released
cpi mr2,0b11111111      ; which will increase the counter in the same way that it happens to
breq released           ; pressed subroutine
rjmp release

released:
inc counter
com counter
rcall delay
out PORTB , counter
com counter
rjmp start

delay:
   ldi  r18, 3
   ldi  r19, 138
   ldi  r20, 86
   L1: dec  r20
   brne L1
   dec  r19
   brne L1
   dec  r18
   brne L1
   rjmp PC+1

ret
```

Task 4

Delay subroutine with variable delay time
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: 08/09/2017
;		Authors:
;			Alexander Risteski
;			Dimitrios Argyriou
;
;		Hardware: STK600, CPU ATmega2560
;
;		Function: This program creates a Ring Counter and displays the values with the LEDs
;		by using Shift instructions(lsl/lsr) with a delay time of 1000 milliseconds (1 millisec-
		ond
;		that repeats for 1000 times).
;
;		Input ports: On-board switches connected to PORTA.
;		Output ports: On-board LEDs connected to PORTB.
```

```
;
;               Included files: m2560def.inc
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"
; initialize the Stack Pointer (SP) tothe highest address in SRAMs (RAMEND)
ldi r25, HIGH(RAMEND)           ; MSB part av address to 100 miliseconds
out SPH, r25                    ; store in SPH
ldi r24, LOW(RAMEND)            ; LSB part av address to 100 miliseconds
out SPL, r24                    ; store inSPL

; Delay NaN cycles
; at 8.0 MHz
.DEF mr = r16                   ; Assigning register r16 onto mr


start:
ldi mr, 0b00000001              ; Make the lower 1 bit output
out DDRB, mr                    ;               for Port B.
                        ; Relative call on delay subroutine

myloop:

ldi r25, HIGH(100)
ldi r24, LOW(100)

lsl mr
rcall wait_milliseconds         ; Logical shift of register r16 left.
out DDRB, mr
                                ; Make the lower 1 bit output for Port B.

cpi mr, 0b00000000              ; Compare register r16 to 0
breq equal                      ; and if equal, branch to equal
                                ; Relative call to delay subroutine
rjmp myloop

equal:
rjmp start                      ; Relative jump to start

wait_milliseconds:
push r16                        ; Store data in r16 and r17 to the stack first
push r17

        ; These instructions take approx 1 ms to complete on ATMEGA2560
L0:
ldi  r16, 2
ldi  r17, 75

L1:
dec  r17
brne L1
dec  r16
brne L1
```

```
                ; To gain N delay, repeat above instructions N times with this
sbiw r25:r24, 1                 ; 16-bit decrementation by subtraction
brne L0                         ; Continue until the 16-bit value is 0x00

pop r17                         ; Return stored data to r16 and r17 from stack
pop r16
ret
```