;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

; ED2022, Computer Technology I

; Lab 3, task 1

; Hardware: STK600, CPU ATmega2560

; Date: 2015-11-26

; Authors: Ivan Hussein

; Function:

; --------

; This program use interrupt to turn ON/OFF LEDs

; Used subroutine:

; Global subroutines (that can be used from other programs): Included

; Other information: Stock Pointer

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.INCLUDE"m2560def.inc"

.CSEG

.org 0

jmp start

.org INT0addr

jmp interrupt\_1 ; interrupt subrotine

.org 0x50

start:

ldi r20, HIGH(RAMEND) ; st-pointer

out SPH,R20

ldi R20, low(RAMEND)

out SPL,R20

ldi r17, 0xFF

out DDRB, r17 ; output port

out PORTB, r17

ldi r16, 0b11111110 ; input port

out DDRD, r16

ldi r16, 0b00000001 ; interrupt enabled

out EIMSK, r16

ldi r16, 0b00000011

sts EICRA, r16

sei ; global interrupt enabled

ldi r20, 0b10000000 ; LED-OFF

wait:

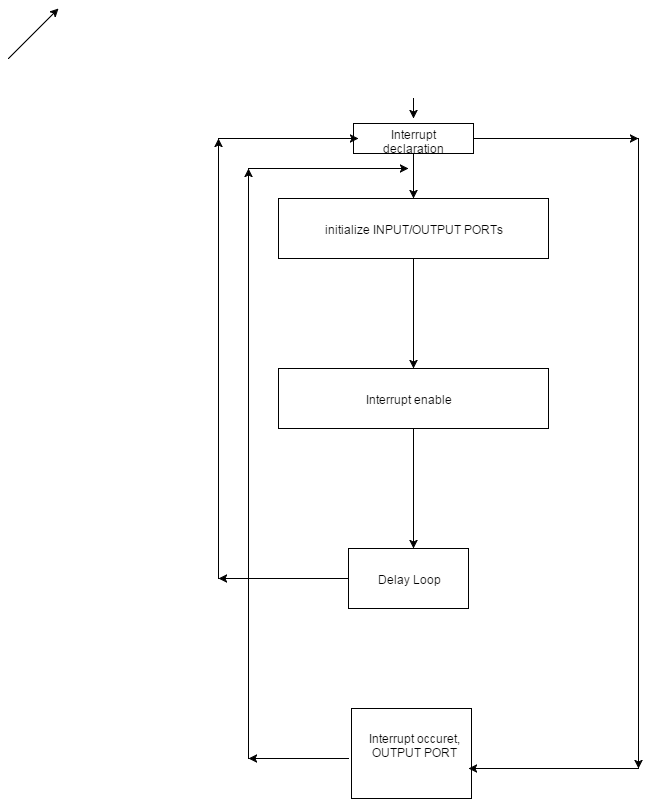
rjmp wait

interrupt\_1: ; change r20 from HIgh to LOW each time interrupt occure

com r20

out PORTB, r20

reti



;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

; ED2022, Computer Technology I

; Lab 3, task 2

; Hardware: STK600, CPU ATmega2560

; Date: 2015-11-26

; Authors: Ivan Hussein

; Function:

; --------

; This program use interrupt to switch between RingCounter and JohnsonCounter

; Used subroutine: Delay as a subroutine

; Global subroutines (that can be used from other programs): Included

; Other information: Stock Pointer

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.INCLUDE"m2560def.inc"

.CSEG

.org 0

jmp start

.org INT0addr

jmp interrupt\_1 ; interrupt subrotine

.org 0x50

start:

ldi r20, HIGH(RAMEND) ; st-pointer

out SPH,R20

ldi R20, low(RAMEND)

out SPL,R20

ldi r17, 0xFF

out DDRA, r17

out PORTA, r17 ; output port

ldi r16, 0b11111110

out DDRD, r16; ; input port

ldi r16, 0b00000001 ; interrupt enabled

out EIMSK, r16

ldi r16, 0b00000011

sts EICRA, r16

sei ; global interrupt enabled

ldi r29, 0b10000000

RingCounter: ; Ringcode starts

ldi r16, 0x00

ldi r17, 0xff

out DDRA, R17;

ldi r20, HIGH(0x45F) ; R20 = high part of RAMEND address

out 0x3E,R20 ; SPH = high part of RAMEND address

ldi R20, low(0x45F) ; R20 = low part of RAMEND address

out 0x3D,R20 ; SPL = low part of RAMEND address

loop:

cpi r29, 0b01111111 ; if interrupt ocurre then it jump to JohnssonCounter

breq JohnssonCounter

dec r17

call Delay\_Tva ; delay after decrimenting r17

out PORTA, r17

inc r17

lsl r17

cpi r17, 0

brne loop

jmp RingCounter

Delay\_Tva: ; delay

ldi r18, 255 ; r18 = 255

ldi r19, 0 ; r19 = loop counter

ldi r23, 255 ; r23 = 255

ldi r24, 0

del\_0:

del\_3:

inc r24 ; r19 = r19+ 1

cp r24, r23 ; compare r18 - 19

brne del\_3 ; if greater or equal, go back

ldi r24, 0

dec r18

cp r19, r18

brne del\_0

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

JohnssonCounter: ; JohnssonCode starts

ldi r16, 0x00

ldi r17, 0xff

out DDRA, r17

jmp loop\_mot\_v ; jomp to left ringCode

ldi r16, 0xff

ldi r17, 0xff

out DDRA, r17

jmp loop\_mot\_h ; jomp to right ringCode

ldi r20, HIGH(0x45F) ; R20 = high part of RAMEND address

out 0x3E,R20 ; SPH = high part of RAMEND address

ldi R20, low(0x45F) ; R20 = low part of RAMEND address

out 0x3D,R20 ; SPL = low part of RAMEND address

loop\_mot\_v:

cpi r29, 0b10000000 ; if interrupt ocurre it jomps to RingCounter

breq RingCounter

call Delay\_Ett ;delay

lsl r17

out PORTA, r17

cpi r17, 0

brne loop\_mot\_v

jmp loop\_mot\_h

loop\_mot\_h:

cpi r29, 0b10000000 ; if interrupt ocurre it jomps to RingCounter

breq RingCounter

cpi r17, 0b11111111

breq loop\_mot\_v

call Delay\_Ett ;delay

com r17

lsr r17

com r17

out PORTA, r17

cp r17, r16

brne loop\_mot\_h

test:

jmp RingCounter

testTva:

jmp JohnssonCounter

Delay\_Ett:

ldi r18, 255

ldi r19, 0

ldi r23, 255

ldi r24, 0

del\_1:

del\_2:

inc r24

cp r24, r23

brne del\_2

ldi r24, 0

dec r18

cp r19, r18

brne del\_1

ret

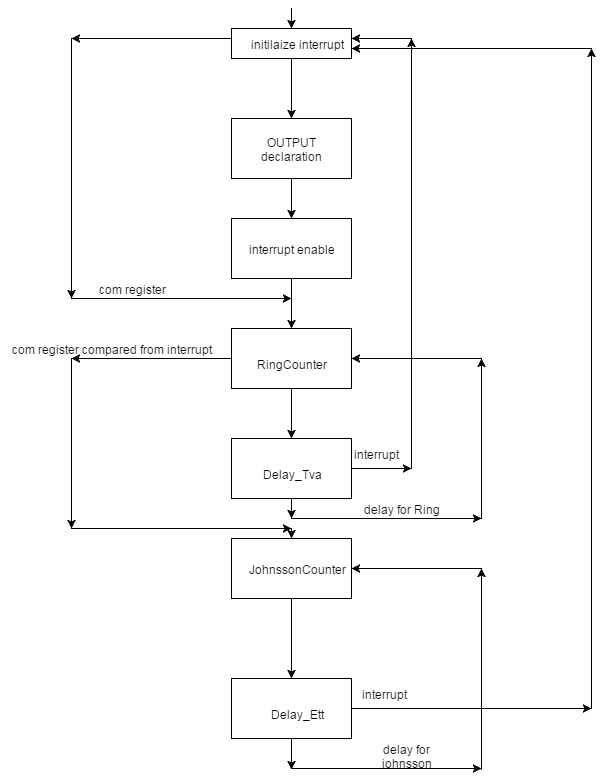
delay:

rjmp delay

interrupt\_1:

com r29

reti



;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

; ED2022, Computer Technology I

; Lab 3, task 3

; Hardware: STK600, CPU ATmega2560

; Date: 2015-11-26

; Authors: Ivan Hussein

; Function:

; --------

; This program use interrupt to simulates the rear lights on a car.

; Normal light: LED 0, 1, 6 and 7 ’ON’.

; Turning right: LED 6 – 7 on, LED 0 – 3 blinking as RING counter.

; Turning left: LED 0 – 1 on, LED 4 – 7 blinking as RING counter.

; Used subroutine: Delay as a subroutine

; Global subroutines (that can be used from other programs): Included

; Other information: Stock Pointer

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.INCLUDE"m2560def.inc"

.CSEG

.org 0

jmp start

.org INT0addr

jmp interrupt\_0 ;interrupt subrotine

.org INT1addr

jmp interrupt\_1

.org 0x50

start:

again:

ldi r20, HIGH(RAMEND) ; st-pointer

out SPH,R20

ldi R20, low(RAMEND)

out SPL,R20

ldi r17, 0xFF

out DDRB, r17

out PORTB, r17 ; output pourt

ldi r16, 0b11111100 ; input pins

out DDRD, r16

;Global register

ldi r16, 0b00000011 ; interrupt enbled

out EIMSK, r16

ldi r16, 0b00001010

sts EICRA, r16

sei ; global interrupt enabled

ldi r29, 0x00

ldi r30,0b00000001

lednormal:

ldi r21, 0x00 ; goes normal

ldi r21, 0b00111100 ; LED-OFF

out PORTB, r21

cpi r29, 0b00000001 ; interrupt jomp to RingCounter2

breq RingCounter2

cpi r30, 0b00000011

breq RingCounter3 ; interrupt jomp to RingCounter3

jmp lednormal

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

RingCounter2:

com r20 ; complement r20

ldi r20, 0b00010011 ; goes to left

com r20 ; com r20

call Delay\_Ett ; delay

in r29, PIND ; input pin

cpi r29, 0b00000001 ; r29 increment when it interruptd, compare to a constant

breq hopp ; jomp hopp

out PORTB, r20 ; put the value of r20 to output port

ldi r20, 0b00100011

com r20

call Delay\_Ett

in r29, PIND

cpi r29, 0b00000001

breq hopp

out PORTB, r20

ldi r20, 0b01000011

com r20

call Delay\_Ett

in r29, PIND

cpi r29, 0b00000001

breq hopp

out PORTB, r20

ldi r20, 0b10000011

com r20

call Delay\_Ett

in r29, PIND

cpi r29, 0b00000001

breq hopp

out PORTB, r20

ldi r20, 0b00000011

com r20

call Delay\_Ett

in r29, PIND

cpi r29, 0b00000001

breq hopp

out PORTB, r20

call Delay\_Ett

in r29, PIND

cpi r29, 0b00000100

breq hopp

in r29, PIND

cpi r29, 0b00000001

brne RingCounter2

hopp:

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

RingCounter3: ; same process but to left

com r20 ; it goes to right

ldi r20, 0b11001000

com r20

call Delay\_Ett

in r30, PIND

cpi r30, 0b00000010

breq hoppB

out PORTB, r20

ldi r20, 0b11000100

com r20

call Delay\_Ett

in r30, PIND

cpi r30, 0b00000010

breq hoppB

out PORTB, r20

ldi r20, 0b11000010

com r20

call Delay\_Ett

in r30, PIND

cpi r30, 0b00000010

breq hoppB

out PORTB, r20

ldi r20, 0b11000001

com r20

call Delay\_Ett

in r30, PIND

cpi r30, 0b00000010

breq hoppB

out PORTB, r20

ldi r20, 0b11000000

com r20

call Delay\_Ett

in r30, PIND

cpi r30, 0b00000010

breq hoppB

out PORTB, r20

call Delay\_Ett

in r30, PIND

cpi r30, 0b00000010

brne RingCounter3

hoppB:

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

interrupt\_0:

inc r29

reti

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

interrupt\_1:

inc r30

reti

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Delay\_Ett:

ldi r18, 255 ; r18 = 255

ldi r19, 0 ; r19 = loop counter

ldi r23, 255 ; r23 = 255

ldi r24, 0

del\_1:

del\_2:

inc r24 ; r19 = r19+ 1

cp r24, r23 ; compare r18 - 19

brne del\_2 ; if greater or equal, go back

ldi r24, 0

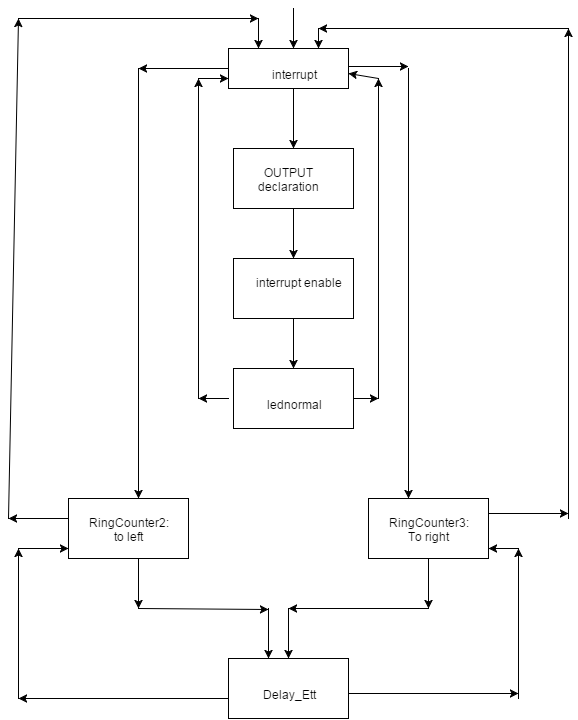
dec r18

cp r19, r18

brne del\_1

ret

reti



;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

; ED2022, Computer Technology I

; Lab 3, task 4

; Hardware: STK600, CPU ATmega2560

; Date: 2015-11-26

; Authors: Ivan Hussein

; Function:

; --------

; This program use interrupt to simulates the rear lights on a car.

; Stop: all lights ON.

; Turning right and brake: LED 4 – 7 on, LED 0 – 3 blinking as RING counter.

; Turning left and brake: LED 0 – 3 on, LED 4 – 7 blinking as RING counter.

; Used subroutine: Delay as a subroutine

; Global subroutines (that can be used from other programs): Included

; Other information: Stock Pointer

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.INCLUDE"m2560def.inc"

.CSEG

.org 0

jmp start

.org INT0addr

jmp interrupt\_0 ;interrput

.org INT1addr

jmp interrupt\_1

.org INT2addr

jmp interrupt\_2

.org 0x50

start:

again:

ldi r20, HIGH(RAMEND)

out SPH,R20 ; st-pointer

ldi R20, low(RAMEND)

out SPL,R20

ldi r17, 0xFF

out DDRB, r17 ; output port

out PORTB, r17

ldi r16, 0b11111000

out DDRD, r16

ldi r16, 0b00000111 ; all interrupt enabled

out EIMSK, r16

ldi r16, 0b11101010

sts EICRA, r16

sei ; global interrupt enabled

ldi r18, 0x00

ldi r22, 0x00

ldi r29, 0x00

ldi r30,0b00000000

ldi r28, 0x00

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

lednormal:

ldi r21, 0x00

ldi r21, 0b00111100 ; led normal

out PORTB, r21

cpi r29, 0b00000001

breq RingCounter2 ; interrupt occurre jmp to RingCounter2

cpi r30, 0b00000001 ; interrupt occurre jmp to RingCounter2

breq hoppToRingCounter3

brne shorthopp

hoppToRingCounter3:

jmp RingCounter3

shorthopp:

cpi r18, 0x01

brne lednormal

bromsar: ; iand this interrupt stop , all ends on

out PORTB, r22

cpi r18, 0x02

breq hoppbrom

brne bromsar

hoppbrom:

jmp start

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

RingCounter2:

com r20 ; comelemnt becaouse of interrupt call

ldi r20, 0b00010011 ; leds hoing and then turn to left

com r20 ; com the register

call Delay\_Ett ; delay

cpi r18, 0x01 ; compare it with a constant

breq JohnsonCounterLeft ; if same jomp to JohnsonCounterLeft

cpi r29, 0x02

breq hopp

out PORTB, r20

ldi r20, 0b00100011

com r20

call Delay\_Ett

cpi r18, 0x01

breq JohnsonCounterLeft

cpi r29, 0x02

breq hopp

out PORTB, r20

ldi r20, 0b01000011

com r20

call Delay\_Ett

cpi r18, 0x01

breq JohnsonCounterLeft

cpi r29, 0x02

breq hopp

out PORTB, r20

ldi r20, 0b10000011

com r20

call Delay\_Ett

cpi r18, 0x01

breq JohnsonCounterLeft

cpi r29, 0x02

breq hopp

out PORTB, r20

ldi r20, 0b00000011

com r20

call Delay\_Ett

cpi r18, 0x01

breq JohnsonCounterLeft

cpi r29, 0x02

breq hopp

out PORTB, r20

call Delay\_Ett

cpi r18, 0x01

breq JohnsonCounterLeft

cpi r29, 0x02

breq hopp

cpi r18, 0x01

breq JohnsonCounterLeft

cpi r29, 0x02

brne RingCounter2

JohnsonCounterLeft:

jmp JohnsonCounterLeftA

hopp:

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

RingCounter3:

com r20 ; comelemnt becaouse of interrupt call

ldi r20, 0b11001000 ; leds hoing and then turn to left

com r20 ; com the register

call Delay\_Ett ;delay

cpi r18, 0x01 ; compare it with a constant

breq JohnsonCounterRight

cpi r30, 0x02 ; if same jomp to JohnsonCounterRight

breq hoppB

out PORTB, r20

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ldi r20, 0b11000100

com r20

call Delay\_Ett

cpi r18, 0x01

breq JohnsonCounterRight

cpi r30, 0x02

breq hoppB

out PORTB, r20

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ldi r20, 0b11000010

com r20

call Delay\_Ett

cpi r18, 0x01

breq JohnsonCounterRight

cpi r30, 0x02

breq hoppB

out PORTB, r20

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ldi r20, 0b11000001

com r20

call Delay\_Ett

cpi r18, 0x01

breq JohnsonCounterRight

cpi r30, 0x02

breq hoppB

out PORTB, r20

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ldi r20, 0b11000000

com r20

call Delay\_Ett

cpi r18, 0x01

breq JohnsonCounterRight

cpi r30, 0x02

breq hoppB

out PORTB, r20

call Delay\_Ett

cpi r18, 0x01

breq JohnsonCounterRight

cpi r30, 0x02

breq hoppB

cpi r30, 0x02

brne RingCounter3

hoppB:

cpi r30, 0x02

breq hoppagain

JohnsonCounterRight:

jmp JohnsonCounterRightA

hoppagain:

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

interrupt\_0:

inc r29

reti

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

interrupt\_1:

inc r30

reti

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

interrupt\_2:

inc r28

inc r18

reti

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Delay\_Ett:

ldi r18, 255 ; r18 = 255

ldi r19, 0 ; r19 = loop counter

ldi r23, 255 ; r23 = 255

ldi r24, 0

del\_1:

del\_2:

inc r24 ; r19 = r19+ 1

cp r24, r23 ; compare r18 - 19

brne del\_2 ; if greater or equal, go back

ldi r24, 0

dec r18

cp r19, r18

brne del\_1

ret

reti

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

JohnsonCounterLeftA:

com r20 ; comelemnt becaouse of interrupt call

ldi r20, 0b00011111 ; leds hoing and then turn to left

com r20 ; com the register

call Delay\_Ett ;delay

cpi r18, 0x02 ; compare it with a constant

breq hoppLeft ; if same jomp to hoppLeft

out PORTB, r20

ldi r20, 0b00101111

com r20

call Delay\_Ett

cpi r18, 0x02

breq hoppLeft

out PORTB, r20

ldi r20, 0b01001111

com r20

call Delay\_Ett

cpi r18, 0x02

breq hoppLeft

out PORTB, r20

ldi r20, 0b10001111

com r20

call Delay\_Ett

cpi r18, 0x02

breq hoppLeft

out PORTB, r20

ldi r20, 0b00001111

com r20

call Delay\_Ett

cpi r18, 0x02

breq hoppLeft

out PORTB, r20

call Delay\_Ett

cpi r18, 0x02

breq hoppLeft

cpi r28, 0x02

brne JohnsonCounterLeftA

hoppleft:

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

JohnsonCounterRightA:

com r20

ldi r20, 0b11111000

com r20

call Delay\_Ett

cpi r18, 0x02

breq hoppNow

out PORTB, r20

ldi r20, 0b11110100

com r20

call Delay\_Ett

cpi r18, 0x02

breq hoppNow

out PORTB, r20

ldi r20, 0b11110010

com r20

call Delay\_Ett

cpi r18, 0x02

breq hoppNow

out PORTB, r20

ldi r20, 0b11110001

com r20

call Delay\_Ett

cpi r18, 0x02

breq hoppNow

out PORTB, r20

ldi r20, 0b11110000

com r20

call Delay\_Ett

cpi r18, 0x02

breq hoppNow

out PORTB, r20

call Delay\_Ett

cpi r18, 0x02

breq hoppNow

cpi r28, 0x02

brne JohnsonCounterRightA

hoppNow:

ret

