



Computer Technology I

Lab. 5 :Display JHD202



Author: LOIC GALLAND,
LEONARDO PEDRO

Supervisor:

Semester: Autumn 2019

Area: Computer Science

Course code: 1DT301

Contents

1	Task 1 - Write a program that displays a character on the display	1
2	Task 2 - Electronic bingo machine	5
3	Task 3 -Serial communication and display	13
4	Task 4 - Modify the program in task 3	17
5	Task 5 - Serial communication using Interrupt	19

1 Task 1 - Write a program that displays a character on the display

Write a program in Assembly that displays the character "percent". Look in the data sheet how to initiate the display. The data sheet you'll find on <https://www.student.vxu.se/>. The display will be connected as in the figure above. 4-bit-mode should be used, since only RS, E, D7, D6, D5 and D4 are connected to I/O-pins on the STK600.

[illegible]

```

; **
; ** init_display
; **
init_disp:
    rcall power_up_wait           ; wait for display to power up
    ldi Data, BITMODE4           ; 4-bit operation
    rcall write_nibble           ; (in 8-bit mode)
    rcall short_wait             ; wait min. 39 us
    ldi Data, DISPCTRL           ; disp. on, blink on, curs. On
    rcall write_cmd              ; send command
    rcall short_wait             ; wait min. 39 us

rcall clr_disp
ldi Data, 0x25
rcall write_char
loop:    nop
        rjmp loop                ; loop forever

clr_disp:
    ldi Data, CLEAR              ; clr display
    rcall write_cmd              ; send command
    rcall long_wait              ; wait min. 1.53 ms
    ret

; **
; ** write char/command
; **

write_char:
    ldi RS, 0b00100000           ; RS = high
    rjmp write

write_cmd:
    clr RS                       ; RS = low

write:
    mov Temp, Data                ; copy Data
    andi Data, 0b11110000        ; mask out high nibble
    swap Data                     ; swap nibbles
    or Data, RS                   ; add register select
    rcall write_nibble           ; send high nibble
    mov Data, Temp                ; restore Data
    andi Data, 0b00001111        ; mask out low nibble
    or Data, RS                   ; add register select

write_nibble:
    rcall switch_output           ; Modify for display JHD202A,
    port E
    nop                           ; wait 542nS
    sbi PORTE, 5                  ; enable high, JHD202A
    nop
    nop                           ; wait 542nS
    cbi PORTE, 5                  ; enable low, JHD202A
    nop
    nop                           ; wait 542nS
    ret

; **
; ** busy_wait loop
; **

```

```

short_wait:
    clr zh                                ; approx 50 us
    ldi zl, 30
    rjmp wait_loop
long_wait:
    ldi zh, HIGH(1000)                    ; approx 2 ms
    ldi zl, LOW(1000)
    rjmp wait_loop
dbnc_wait:
    ldi zh, HIGH(4600)                    ; approx 10 ms
    ldi zl, LOW(4600)
    rjmp wait_loop
power_up_wait:
    ldi zh, HIGH(9000)                    ; approx 20 ms
    ldi zl, LOW(9000)

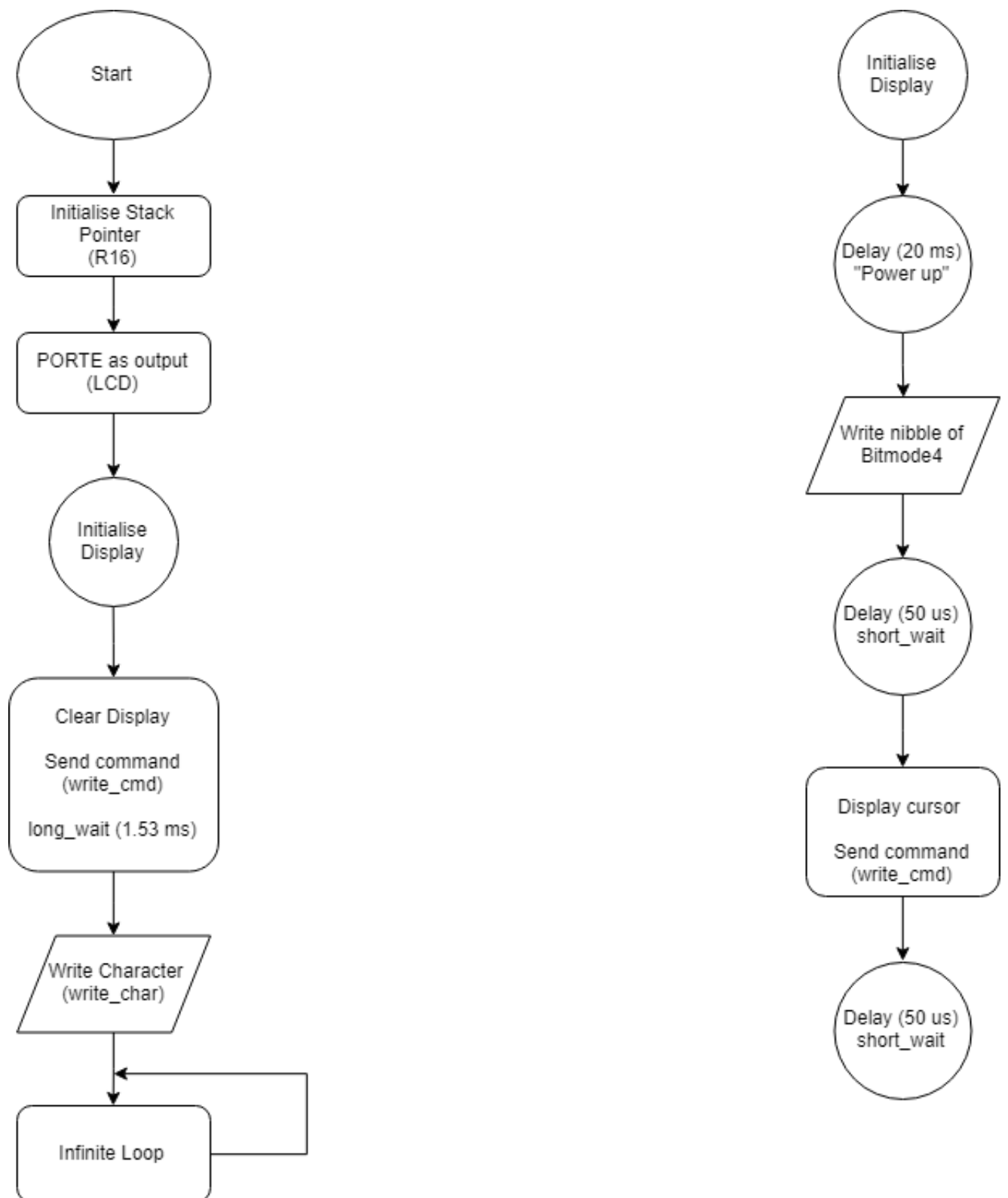
wait_loop:
    sbiw z, 1                             ; 2 cycles
    brne wait_loop                        ; 2 cycles
    ret

; **
; ** modify output signal to fit LCD JHD202A, connected to port E
; **

switch_output:
    push Temp
    clr Temp
    sbrc Data, 0                          ; D4 = 1?
    ori Temp, 0b000000100                ; Set pin 2
    sbrc Data, 1                          ; D5 = 1?
    ori Temp, 0b000001000                ; Set pin 3
    sbrc Data, 2                          ; D6 = 1?
    ori Temp, 0b000000001                ; Set pin 0
    sbrc Data, 3                          ; D7 = 1?
    ori Temp, 0b000000010                ; Set pin 1
    sbrc Data, 4                          ; E = 1?
    ori Temp, 0b001000000                ; Set pin 5
    sbrc Data, 5                          ; RS = 1?
    ori Temp, 0b100000000                ; Set pin 7 (wrong in previous
    version)
    out porte, Temp
    pop Temp
    ret

```

This is the flowchart of the task 1:



2 Task 2 - Electronic bingo machine

You should create an electronic bingo generator. The generator should create random numbers between 1 and 75. The numbers should be displayed on the display. Clear the display before a new value is displayed. Use interrupt and a pushbutton for the input.

[illegible]

```

ser Temp                                ; r16 = 0b11111111
out DDRE, Temp                          ; port E = outputs ( Display
JHD202A)
clr Temp                                ; r16 = 0
out PORTE, Temp

ldi Temp, 0x00
out DDRD, Temp

ldi Temp, 0b00000010 ;Setting INTO into falling edge
sts EICRA, Temp
ldi Temp, 0b00000001 ;Enable INTO
out EIMSK, Temp
ldi RandomNumber, 0x31
ldi RandomNumber2, 0x30
ldi NumberBetween05, 0x30

sei

; **
; ** init_display
; **
init_disp:
    rcall power_up_wait                ; wait for display to power up

    ldi Data, BITMODE4                 ; 4-bit operation
    rcall write_nibble                 ; (in 8-bit mode)
    rcall short_wait                   ; wait min. 39 us
    ldi Data, DISPCTRL                 ; disp. on, blink on, curs. On
    rcall write_cmd                    ; send command
    rcall short_wait                   ; wait min. 39 us

loop:
    rcall increaseNumber
    rcall increaseNumber2
    rcall increaseNumber3
rjmp loop                               ; loop forever

clr_disp:
    ldi Data, CLEAR                    ; clr display
    rcall write_cmd                    ; send command
    rcall long_wait                    ; wait min. 1.53 ms
    ret

; **
; ** write char/command
; **

increaseNumber: ;To get the first digit

    cpi RandomNumber, 0x37 ;
    breq ResetDisplay

```



```

    inc RandomNumber
    jmp END

ResetDisplay:
    ldi RandomNumber,0x31
END:
ret

increaseNumber2:      ;Second digit if the first is not 7
    cpi RandomNumber2,0x39 ;
    breq ResetDisplay2

    inc RandomNumber2
    jmp END2

ResetDisplay2:
    ldi RandomNumber2,0x30
END2:
ret

increaseNumber3: ;Second digit if the first is 7
    cpi NumberBetween05,0x35 ;
    breq ResetDisplay3

    inc NumberBetween05
    jmp END3

ResetDisplay3:
    ldi NumberBetween05,0x30
END3:
ret

write_char:
    ldi RS, 0b00100000          ; RS = high
    rjmp write
write_cmd:
    clr RS                      ; RS = low
write:
    mov Temp, Data              ; copy Data
    andi Data, 0b11110000      ; mask out high nibble
    swap Data                   ; swap nibbles
    or Data, RS                 ; add register select
    rcall write_nibble         ; send high nibble
    mov Data, Temp              ; restore Data
    andi Data, 0b00001111      ; mask out low nibble
    or Data, RS                 ; add register select

write_nibble:
    rcall switch_output        ; Modify for display JHD202A,
    port E
    nop                        ; wait 542nS
    sbi PORTE, 5               ; enable high, JHD202A
    nop
    nop                        ; wait 542nS
    cbi PORTE, 5               ; enable low, JHD202A
    nop
    nop                        ; wait 542nS
    ret

```

```

; **
; ** busy_wait loop
; **
short_wait:
    clr zh                                ; approx 50 us
    ldi zl, 30
    rjmp wait_loop
long_wait:
    ldi zh, HIGH(1000)                    ; approx 2 ms
    ldi zl, LOW(1000)
    rjmp wait_loop
dbnc_wait:
    ldi zh, HIGH(4600)                    ; approx 10 ms
    ldi zl, LOW(4600)
    rjmp wait_loop
power_up_wait:
    ldi zh, HIGH(9000)                    ; approx 20 ms
    ldi zl, LOW(9000)

wait_loop:
    sbiw z, 1                             ; 2 cycles
    brne wait_loop                        ; 2 cycles
    ret

; **
; ** modify output signal to fit LCD JHD202A, connected to port E
; **

switch_output:
    push Temp
    clr Temp
    sbrc Data, 0                          ; D4 = 1?
    ori Temp, 0b00000100                  ; Set pin 2
    sbrc Data, 1                          ; D5 = 1?
    ori Temp, 0b00001000                  ; Set pin 3
    sbrc Data, 2                          ; D6 = 1?
    ori Temp, 0b00000001                  ; Set pin 0
    sbrc Data, 3                          ; D7 = 1?
    ori Temp, 0b00000010                  ; Set pin 1
    sbrc Data, 4                          ; E = 1?
    ori Temp, 0b00100000                  ; Set pin 5
    sbrc Data, 5                          ; RS = 1?
    ori Temp, 0b10000000                  ; Set pin 7 (wrong in previous
    version)
    out porte, Temp
    pop Temp
    ret

interrupt:                                ;USART Interrupt
    rcall clr_disp
    mov Data, RandomNumber
    rcall write_char
    call power_up_wait
    cpi RandomNumber, 0x37
    brlt IfLessSeven

    call power_up_wait
    mov Data, NumberBetween05
    rcall write_char

```

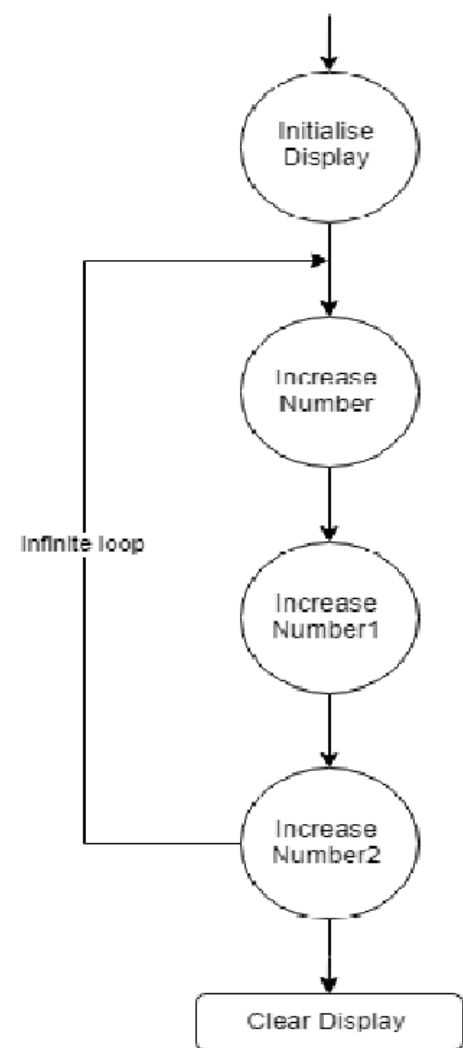
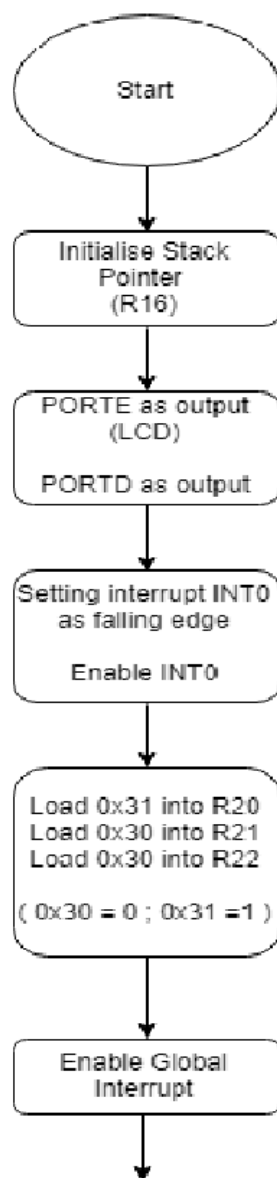
```
    rjmp endInt

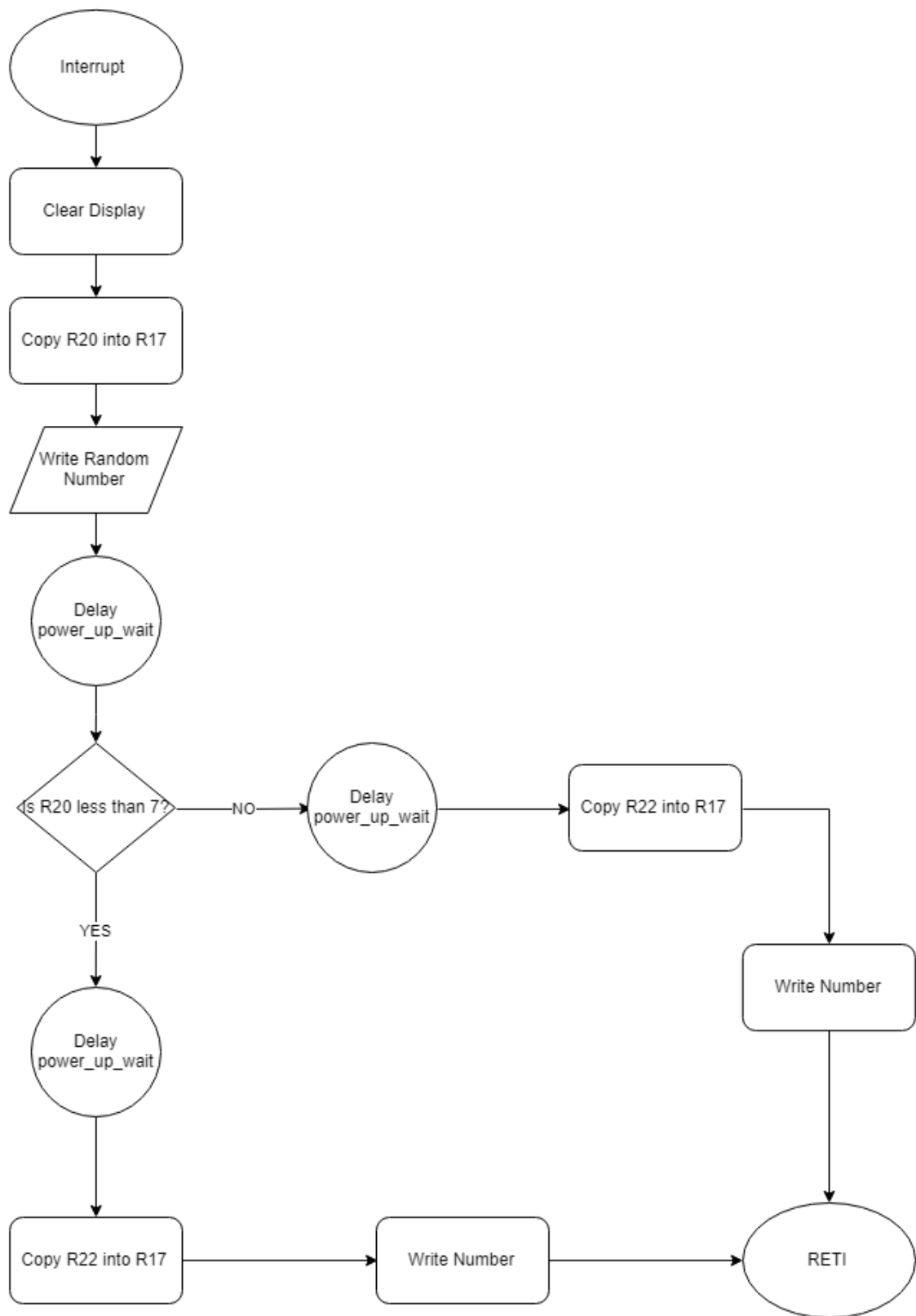
    IfLessSeven:
        call power_up_wait
        mov Data, RandomNumber2
        rcall write_char

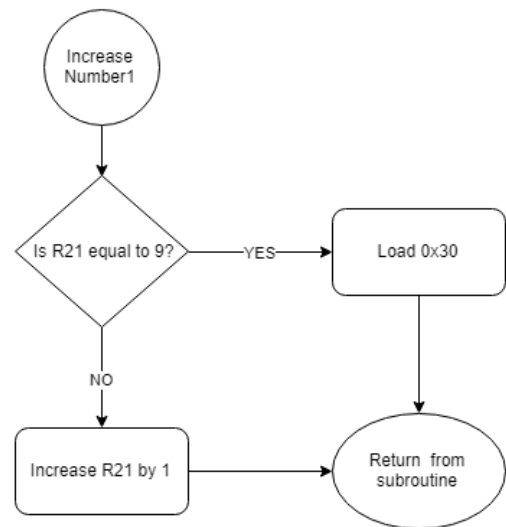
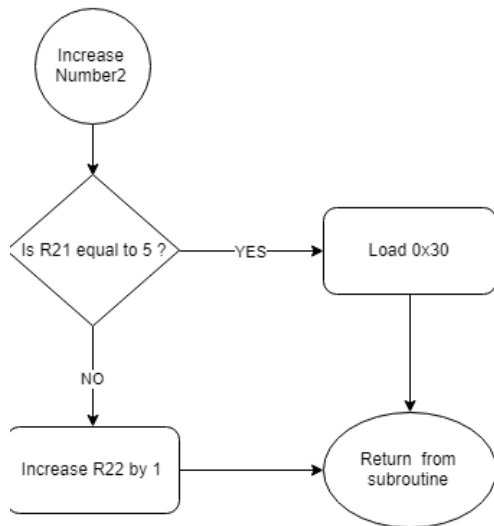
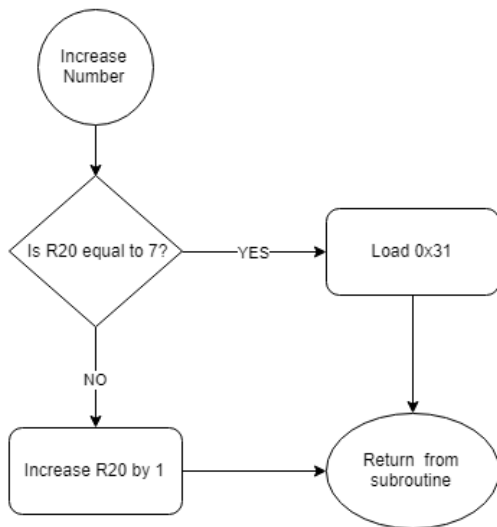
endInt:

RETI
```

This is the flowcharts of the task 2:







3 Task 3 -Serial communication and display

Use program modules from lab 4 and write a program that receives a character on the serial port and displays each character on the display.

[illegible]

```

    out PORTE, Temp

    ldi r16, 23                ;osc = 1.843MHz, 4800 bps => UBBRR = 23
    sts UBRR1L, r16           ;Store Prescaler value in UBRR1L

    ldi r16, (1<<RXEN1 | 1<<TXEN1);Set RX, TX enable flags and
    RXCIE = 1
    sts UCSR1B, r16
    sei                      ;Set global interrupt flag

; **
; ** init_display
; **
init_disp:
    rcall power_up_wait        ; wait for display to power up
    ldi Data, BITMODE4        ; 4-bit operation
    rcall write_nibble        ; (in 8-bit mode)
    rcall short_wait          ; wait min. 39 us
    ldi Data, DISPCTRL        ; disp. on, blink on, curs. On
    rcall write_cmd           ; send command
    rcall short_wait          ; wait min. 39 us

call clr_disp

GetChar:                      ;Receive data
    lds r20, UCSR1A           ;read UCSR1A I/O register to r20
    sbrc r20, RXC1            ;RXC1=1 -> new Character
    rjmp GetChar             ;RXC1=0 -> no character received
    lds r23, UDR1             ;Read character in UDR

    Port_output:
        mov Data, r23
        call write_char

    PutChar:
    lds r20, UCSR1A           ;Read UCSR1A i/O register to r20
    sbrc r20, UDRE1           ;UDRE1 =1 => buffer is empty
    rjmp PutChar             ;UDRE1 = 0 => buffer is not empty
    sts UDR1, r23             ;write character to UDR1
    rjmp GetChar             ;Return to loop

clr_disp:
    ldi Data, CLEAR           ; clr display
    rcall write_cmd           ; send command
    rcall long_wait           ; wait min. 1.53 ms
    ret

; **
; ** write char/command
; **
write_char:
    ldi RS, 0b00100000        ; RS = high
    rjmp write

write_cmd:
    clr RS                    ; RS = low

write:
    mov Temp, Data            ; copy Data
    andi Data, 0b11110000     ; mask out high nibble
    swap Data                 ; swap nibbles
    or Data, RS               ; add register select

```



```

        rcall write_nibble                ; send high nibble
        mov Data, Temp                   ; restore Data
        andi Data, 0b00001111           ; mask out low nibble
        or Data, RS                      ; add register select

write_nibble:
        rcall switch_output              ; Modify for display JHD202A,
        port E
        nop                               ; wait 542nS
        sbi PORTE, 5                     ; enable high, JHD202A
        nop
        nop                               ; wait 542nS
        cbi PORTE, 5                     ; enable low, JHD202A
        nop
        nop                               ; wait 542nS
        ret

; **
; ** busy_wait loop
; **
short_wait:
        clr zh                           ; approx 50 us
        ldi zl, 30
        rjmp wait_loop
long_wait:
        ldi zh, HIGH(1000)               ; approx 2 ms
        ldi zl, LOW(1000)
        rjmp wait_loop
dbnc_wait:
        ldi zh, HIGH(4600)               ; approx 10 ms
        ldi zl, LOW(4600)
        rjmp wait_loop
power_up_wait:
        ldi zh, HIGH(9000)               ; approx 20 ms
        ldi zl, LOW(9000)

wait_loop:
        sbiw z, 1                        ; 2 cycles
        brne wait_loop                  ; 2 cycles
        ret

; **
; ** modify output signal to fit LCD JHD202A, connected to port E
; **
switch_output:
        push Temp
        clr Temp
        sbrc Data, 0                     ; D4 = 1?
        ori Temp, 0b000000100           ; Set pin 2
        sbrc Data, 1                     ; D5 = 1?
        ori Temp, 0b000001000           ; Set pin 3
        sbrc Data, 2                     ; D6 = 1?
        ori Temp, 0b000000001           ; Set pin 0
        sbrc Data, 3                     ; D7 = 1?
        ori Temp, 0b000000010           ; Set pin 1
        sbrc Data, 4                     ; E = 1?
        ori Temp, 0b001000000           ; Set pin 5
        sbrc Data, 5                     ; RS = 1?
        ori Temp, 0b100000000           ; Set pin 7 (wrong in previous

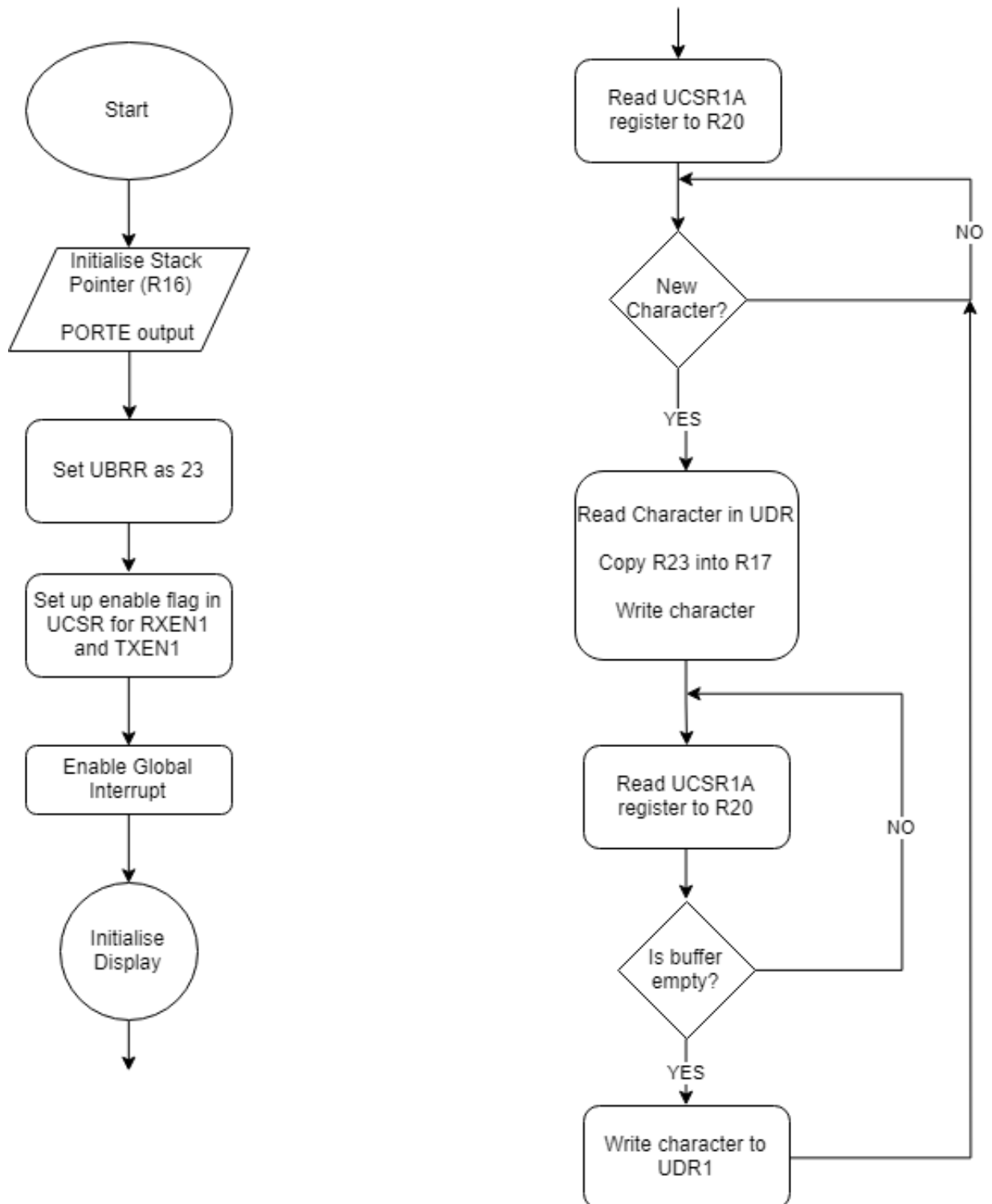
```

```

    version)
    out porte, Temp
    pop Temp
    ret

```

This is the flowchart of the task 3:



4 Task 4 - Modify the program in task 3

Modify the program in task 3 so that 4 lines of text can be displayed. Each textline should be displayed during 5 seconds, after that the text on line 1 should be moved to line 2 and so on. The text should be entered from the terminal program, PUTTY, via the serial port.

[illegible]

```

ser Temp                                ; r16 = 0b11111111
out DDRE, Temp                          ; port E = outputs ( Display
JHD202A)
clr Temp                                ; r16 = 0
out PORTE, Temp

ldi r16, 23                             ;osc = 1.843MHz, 4800 bps => UBBRR = 23
sts UBRR1L , r16                        ;Store Prescaler value in UBRR1L

ldi r16, (1<<RXEN1 | 1<<TXEN1);Set RX, TX enable flags and
RXCIE = 1
sts UCSR1B, r16
sei ;Set global interrupt flag
;Inialiaze Y and X pointer to memory address 0x200
ldi YH , HIGH (0x200)
ldi YL , LOW(0x200)
ldi XH , HIGH (0x200)
ldi XL , LOW(0x200)

; **
; ** init_display
; **
init_disp:
    rcall power_up_wait                 ; wait for display to power up
    ldi Data, BITMODE4                 ; 4-bit operation
    rcall write_nibble                 ; (in 8-bit mode)
    rcall short_wait                   ; wait min. 39 us
    ldi Data, DISPCTRL                 ; disp. on, blink on, curs. On
    rcall write_cmd                     ; send command
    rcall short_wait                   ; wait min. 39 us

call clr_disp

GetChar: ;Receive data
    lds r21, UCSR1A ;read UCSR1A I/O register to r20
    sbrc r21,RXC1 ;RXC1=1 -> new Character
    rjmp GetChar ;RXC1=0 -> no character received
    lds r23,UDR1 ;Read character in UDR
    cpi r23,0x0D
    breq next_line
    st Y+,r23

Port_output:
    ;call clr_disp
    mov Data,r23
    call write_char

PutChar:
    lds r21, UCSR1A ;Read UCSR1A i/O register to r20
    sbrc r21, UDRE1 ;UDRE1 =1 => buffer is empty
    rjmp PutChar ;UDRE1 = 0 => buffer is not empty
    sts UDR1,r23 ;write character to UDR1
    rjmp END

next_line:
    rcall FiveSec_delay
    ldi Data, CLEAR
    rcall write_cmd
    rcall long_wait
    ldi Data, 0x40

```

```

        rcall write_cmd
        ;clr Data

        loop:
        ;COMPARE BEFORE IT GETS MEMORY OUT OF
        BOUNDARY
        cp YH , XH
        brne continue_printing
        cp YL , XL
        breq Stop

        continue_printing:
        ld Data,X+      ;load from X
                        ;pointer to Data
        rcall write_char
        rcall long_wait
        rjmp loop

        Stop:
        ;Reinialize the pointers
        ldi YH , HIGH (0x200)
        ldi YL , LOW(0x200)
        ldi XH , HIGH (0x200)
        ldi XL , LOW(0x200)
        ldi Data, 0b00000010
        rcall write_cmd

        rjmp End

END:nop
rjmp GetChar      ;Return to loop

clr_disp:
        ldi Data, CLEAR      ; clr display
        rcall write_cmd      ; send command
        rcall long_wait      ; wait min. 1.53 ms
        ret

; **
; ** write char/command
; **

write_char:
        ldi RS, 0b00100000      ; RS = high
        rjmp write

write_cmd:
        clr RS      ; RS = low

write:
        mov Temp, Data      ; copy Data
        andi Data, 0b11110000 ; mask out high nibble
        swap Data      ; swap nibbles
        or Data, RS      ; add register select
        rcall write_nibble ; send high nibble
        mov Data, Temp      ; restore Data
        andi Data, 0b00001111 ; mask out low nibble
        or Data, RS      ; add register select

write_nibble:
        rcall switch_output      ; Modify for display JHD202A,
        port E

```

```

        nop                                ; wait 542nS
        sbi PORTE, 5                       ; enable high, JHD202A
        nop
        nop                                ; wait 542nS
        cbi PORTE, 5                       ; enable low, JHD202A
        nop
        nop                                ; wait 542nS
        ret

; **
; ** busy_wait loop
; **
short_wait:
        clr zh                             ; approx 50 us
        ldi zl, 30
        rjmp wait_loop
long_wait:
        ldi zh, HIGH(1000)                ; approx 2 ms
        ldi zl, LOW(1000)
        rjmp wait_loop
dbnc_wait:
        ldi zh, HIGH(4600)                ; approx 10 ms
        ldi zl, LOW(4600)
        rjmp wait_loop
power_up_wait:
        ldi zh, HIGH(9000)                ; approx 20 ms
        ldi zl, LOW(9000)

wait_loop:
        sbiw z, 1                         ; 2 cycles
        brne wait_loop                   ; 2 cycles
        ret

FiveSec_delay:
; Generated by delay loop calculator
; at http://www.bretmulvey.com/avrdelay.html
; Delay 9 215 000 cycles
; 5s at 1.843 MHz
        ldi r18, 47
        ldi r19, 192
        ldi r20, 104
L1: dec r20
        brne L1
        dec r19
        brne L1
        dec r18
        brne L1
RET

; **
; ** modify output signal to fit LCD JHD202A, connected to port E
; **
switch_output:
        push Temp
        clr Temp
        sbrc Data, 0                      ; D4 = 1?
        ori Temp, 0b00000100             ; Set pin 2
        sbrc Data, 1                      ; D5 = 1?
        ori Temp, 0b00001000             ; Set pin 3

```

sbrcl Data, 2	; D6 = 1?
ori Temp, 0b00000001	; Set pin 0
sbrcl Data, 3	; D7 = 1?
ori Temp, 0b00000010	; Set pin 1
sbrcl Data, 4	; E = 1?
ori Temp, 0b00100000	; Set pin 5
sbrcl Data, 5	; RS = 1?
ori Temp, 0b10000000	; Set pin 7 (wrong in previous
version)	
out porte, Temp	
pop Temp	
ret	

This is the flowchart of the task 4: