1DT301 lab5

Songho Lee Sarpreet Singh Buttar

October 7, 2016

1 Introduction

This report provides the solutions for the fifth laboratory of the course 1DT301, which is focusing on utilising of the LCD display (JHD202) and serial communication of STK600, and pointers for manipulating the SRAM memory.. For all tasks it is configured to run on the CPU oscillation rate of 1.8432MHz. It is due to our choice of UBRR as 47 (UBRR baud rate as 2400bps) and the provided initialisation sequence of display is configured to the same frequency rate.

2 Task 1

We are connecting the display on E port using 4-bit mode since it is easier to connect than 8 bit mode as it only requires to connect RS, E, D7, D6, D5 and D4 from the display to pinE on the STK600.

The programme is performing the initialisation sequence as provided on the section 12 of the JHD202C display which is including the wait sequence; subroutines for the initialising part is provided by Anders on the mymoodle. After initialising the display, we print % mark which is equivalent to 0*b*00100101. The code of the programme is presented below, and coming chart describes the process of initialisation of the display. The flowchart presented in task 1 is repeated on the whole tasks of the current lab; therefore the repeated part of the flowchart is not presented on the other flowcharts. Instead the process will be marked as "initialise display".

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;     1DT301, Computer Technology I
;     Date: 2016-10-03
;     Author:
;     Songho Lee
;     Sarpreet Singh
;
;
;     Lab number: 5
;     Title: Task 1
;
;     Hardware: STK600, CPU ATmega2560, JHD202
;
;     Function: Display text on JHD202
;
;     Input ports: None
;
;     Output ports: JHD2012 on PORTE.
;
;     Subroutines: If applicable.
;     Included files: m2560def.inc
;                           Initialize display
;
;     Other information:
;     Changes in program: (Description and date)
;
```

```
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


;     1DT301
;     Lab 5, Initialize display JHD202A.
;
;     Date: 2016-09-30
;     Author, Anders Haggren
;    Modified:
;
;     Function
;     --------
;     Initialize display JHD202 connected to PORTE
;
;     (run @ 1.8432 MHz clk frequency)
;

.include  "m2560def.inc"
.def Temp = r16
.def Data = r17
.def RS   = r18

.equ BITMODE4  = 0b00000010        ; 4-bit operation
.equ CLEAR     = 0b00000001             ; Clear display
.equ DISPCTRL  = 0b00001111        ; Display on, cursor on, blink on.

.cseg
.org 0x0000                   ; Reset vector
     jmp reset

.org 0x0072

reset:

     ldi Temp, HIGH(RAMEND)   ; Temp = high byte of ramend address
     out SPH, Temp            ; sph = Temp
     ldi Temp, LOW(RAMEND)    ; Temp = low byte of ramend address
     out SPL, Temp            ; spl = Temp

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

; **
; ** 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, 0b00100101
```

```
    rcall write_char

loop:     nop
90        rjmp loop           ; loop forever


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

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

write_char:
          ldi RS, 0b00100000        ; RS = high
105       rjmp write
write_cmd:
          clr RS                        ; RS = low
write:
          mov Temp, Data            ; copy Data
110       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
115       andi Data, 0b00001111     ; mask out low nibble
          or Data, RS                   ; add register select

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

; **
130 ; ** busy_wait loop
; **
short_wait:
          clr zh                        ; approx 50 us
          ldi zl, 30
135       rjmp wait_loop
long_wait:
          ldi zh, HIGH(1000)        ; approx 2 ms
          ldi zl, LOW(1000)
          rjmp wait_loop
140 dbnc_wait:
          ldi zh, HIGH(4600)        ; approx 10 ms
          ldi zl, LOW(4600)
          rjmp wait_loop
power_up_wait:
145       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
```

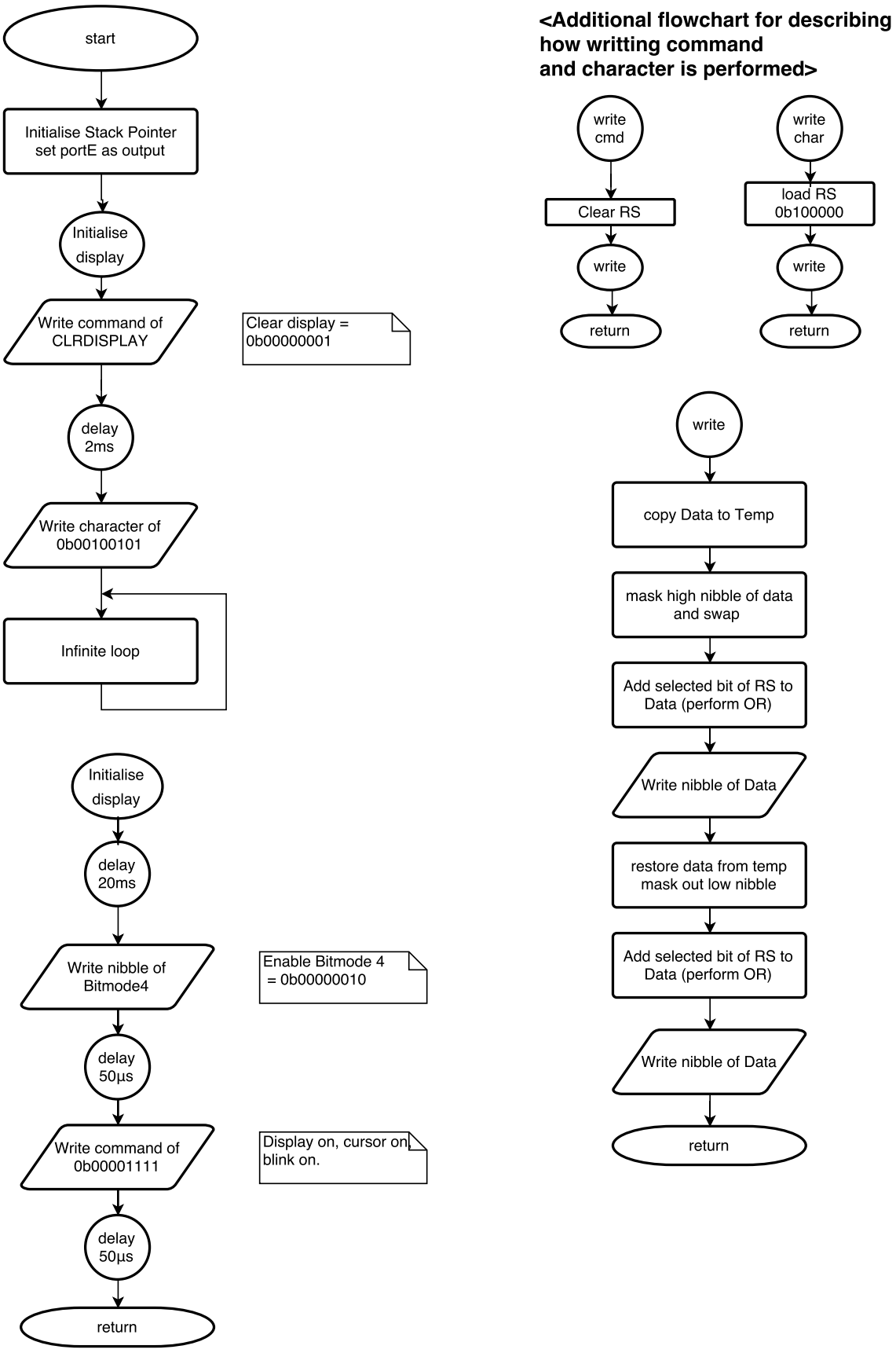

Figure 1: Flowchart of task 1

3 Task 2

An Electric bingo generator is created in this task. Hereby bingo generator refers to a programme which creates a random integer which ranges between 1 and 75. However, we do not have pure random function on the commands of ATMega CPU and therefore it is going to use similar mechanism on the previous second lab which was to generate in accordance with the time differences of durations when switches are pressed. However, unlike the second lab, responses of switch inputs are now handled by external interrupt. Furthermore, number which need to be generated exceeds 9, which means that it is displayed in two digits. The display takes only one char per input and therefore it is required to divide two digits integer into two chars. For example integer 10 needs to be presented 1 and 0 respectively. Thus it is decided to hold two separate integers for those under 10 and 10's decimal holder. Code for the programme is presented below, and flowchart of the programme follows.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;     1DT301, Computer Technology I
;     Date: 2016-10-03
;     Author:
;     Songho Lee
;     Sarpreet Singh
;
;
;     Lab number: 5
;     Title: Task 2
;
;     Hardware: STK600, CPU ATmega2560, JHD202
;
;     Function: Display text on JHD202
;
;     Input ports: Switches on portD
;
;     Output ports: JHD2012 on PORTE.
;
;     Subroutines: If applicable.
;     Included files: m2560def.inc
;                          Initialize display by Anders Haggren
;
;     Other information:
;     Changes in program: (Description and date)
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include  "m2560def.inc"
.def Temp = r16
.def Data = r17
.def RS   = r18
.def      BINGO_lit = r19
.def BINGO_big = r20

.equ BITMODE4  = 0b00000010        ; 4-bit operation
.equ CLEAR     = 0b00000001             ; Clear display
.equ DISPCTRL  = 0b00001111        ; Display on, cursor on, blink on.

.cseg
.org 0x0000                    ; Reset vector
     jmp reset

.org INT1addr
rjmp generate_random

.org 0x0072
```

```
reset:

     ldi Temp, HIGH(RAMEND)    ; Temp = high byte of ramend address
     out SPH, Temp             ; sph = Temp
     ldi Temp, LOW(RAMEND)     ; Temp = low byte of ramend address
     out SPL, Temp             ; spl = Temp

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

; **
; ** 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

;;   Enable interrupt for SW1
ldi temp,0b10
out EIMSK, temp
ldi temp, 0b1000
sts EICRA, temp

sei  ;Enable global interrupt

rcall clr_disp
loop:

     rjmp loop           ; loop forever


;***************SUB ROUTINES*********************
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, 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

generate_random:
push temp

reset_Bingo:
ldi BINGO_big, 0
ldi BINGO_lit, 1

increase:
;    Listen if switch has released.
in temp, pinD
cpi temp, 0xFF
breq quit

;    Increase Bingo
inc BINGO_lit

;    replace 10 in little to big
;    to update decimal holder.

cpi BINGO_lit, 10
brne compare_bing
inc Bingo_big
ldi Bingo_lit, 0

compare_bing:
cpi BINGO_big, 7
brne increase
cpi BINGO_lit, 6
breq reset_Bingo

rjmp increase
quit:

rcall clr_disp
rcall short_wait

;    Write bingo on LCD
mov Data, bingo_big
ldi temp, 0b00110000
add Data, temp

rcall write_char
mov Data, bingo_lit
ldi temp, 0b00110000
add Data, temp
rcall short_wait
rcall write_char

pop temp
reti
```

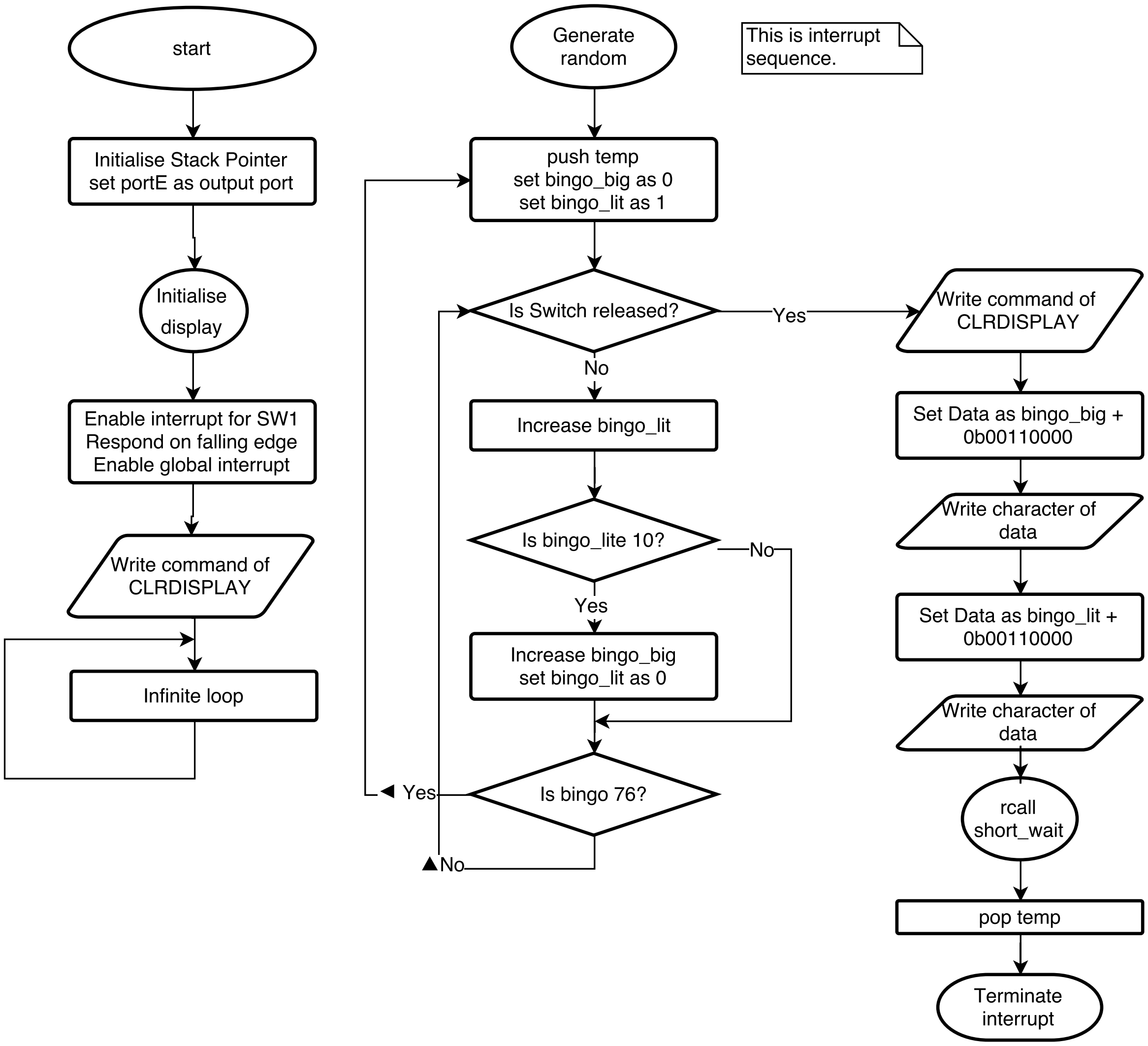
start
Initialise Stack Pointer
set portE as output port
Initialise
display
Enable interrupt for SW1
Respond on falling edge
Enable global interrupt
Write command of
CLRDISPLAY
Infinite loop
Generate
random
This is interrupt
sequence.
push temp
set bingo_big as 0
set bingo_lit as 1
Is Switch released?
Yes
No
Increase bingo_lit
Is bingo_lite 10?
No
Yes
Increase bingo_big
set bingo_lit as 0
Is bingo 76?
Yes
No
Write command of
CLRDISPLAY
Set Data as bingo_big +
0b00110000
Write character of
data
Set Data as bingo_lit +
0b00110000
Write character of
data
rcall
short_wait
pop temp
Terminate
interrupt

Figure 2: Flowchart of task 2

4 Task 3

Task 3 asks to rewrite a programme to get a character input via a serial communication port (RS232) using Universal Synchronous and Asynchronous serial Receiver and Transmitter(USART). The code and chart is presented below.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;    1DT301, Computer Technology I
;    Date: 2016-10-03
;    Author:
;    Songho Lee
;    Sarpreet Singh
;
;
;    Lab number: 5
;    Title: Task 3
;
;    Hardware: STK600, CPU ATmega2560, JHD202
;
;    Function: Display text on JHD202
```

```
;
;     Input ports: TX,RX on PIND2,PIND3
;
;     Output ports: JHD2012 on PORTE.
;
;     Subroutines: If applicable.
;     Included files: m2560def.inc
;                          Initialize display by Anders Haggren
;
;     Other information:
;     (run @ 1.8432 MHz clk frequency)
;     Changes in program: (Description and date)
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include  "m2560def.inc"
.def Temp = r16
.def Data = r17
.def RS   = r18
.def      temp_usart = r19

.equ BITMODE4  = 0b00000010        ; 4-bit operation
.equ CLEAR     = 0b00000001             ; Clear display
.equ DISPCTRL  = 0b00001111        ; Display on, cursor on, blink on.
.equ      UBRR_choice = 47

.cseg
.org 0x0000                   ; Reset vector
     jmp reset

.org URXC1addr
     rjmp readChar

.org 0x0072

reset:

     ldi Temp, HIGH(RAMEND)   ; Temp = high byte of ramend address
     out SPH, Temp            ; sph = Temp
     ldi Temp, LOW(RAMEND)    ; Temp = low byte of ramend address
     out SPL, Temp            ; spl = Temp

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


     ldi temp, UBRR_choice
     sts UBRR1L, temp
     ldi temp, (1<<TXEN1) | (1<<RXEN1) | (1<<RXCIE1)
     sts UCSR1B, temp

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

     ldi Data, BITMODE4       ; 4-bit operation
```

```
75      rcall write_nibble      ; (in 8-bit mode)
        rcall short_wait        ; wait min. 39 us


        ldi Data, DISPCTRL      ; disp. on, blink on, curs. On
80      rcall write_cmd             ; send command
        rcall short_wait        ; wait min. 39 us

sei  ;Enable global interrupt

85  rcall clr_disp
    loop:

        rjmp loop           ; loop forever


90
    ;***************SUB ROUTINES*********************
    clr_disp:
        ldi Data, CLEAR             ; clr display
        rcall write_cmd             ; send command
95      rcall long_wait             ; wait min. 1.53 ms
        ret

    ; **
    ; ** write char/command
100 ; **

    write_char:
        ldi RS, 0b00100000      ; RS = high
        rjmp write
105 write_cmd:
        clr RS                      ; RS = low
    write:
        mov Temp, Data          ; copy Data
        andi Data, 0b11110000   ; mask out high nibble
110     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
115     or Data, RS                 ; add register select

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


    ; **
    ; ** busy_wait loop
130 ; **
    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

readChar:
     lds temp_usart, UCSR1A
     lds Data, UDR1
     rcall outLCD
     reti

outLCD:
     rcall write_char
     ret
```

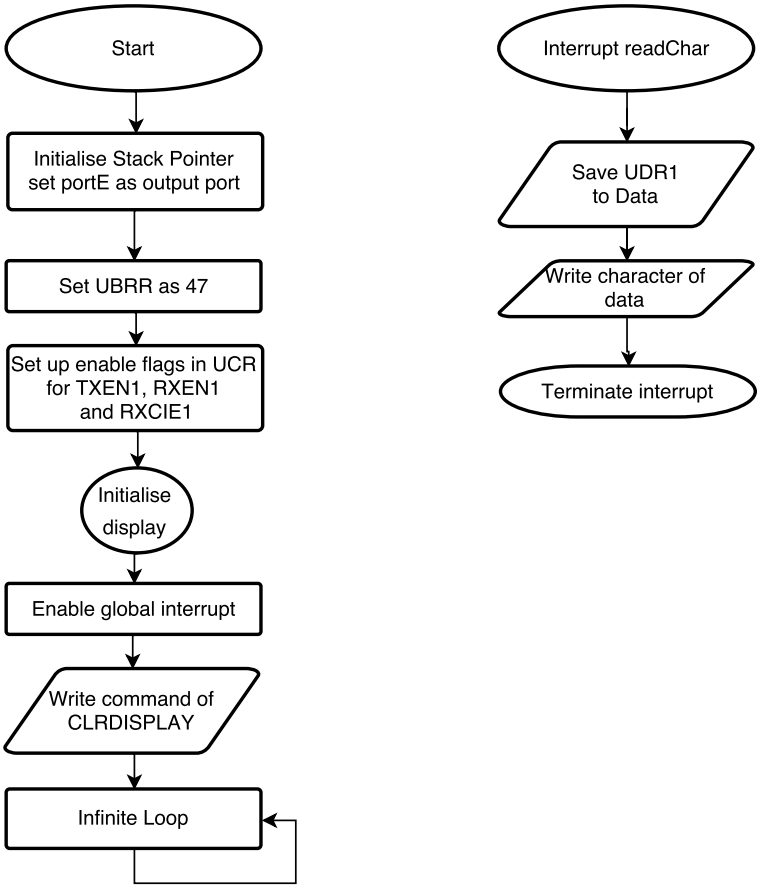

Figure 3: Flowchart of task 3

5 Task 4

Task 4 asks to modify the programme from task 3 to store character inputs into four strings, then print out line by line on LCD but print current string as line 1 of LCD and previous string on line 2 of the LCD. For this, pointer Y is introduce to the point the address in memory where new char to be stored, and pointer X will be iterating over the memory untill the Y pointer so that it saves the series of chars into memory and reads it.

In order to print out two-lines on the display, it is necessary to print out current string and previous string, which means it needs to manipulate the iteration point backwards and point which it was called again. Therefore memory PL1, and PL2 is introduced. PL1 stores previous String's position for X and PL2 stores current String's position for X. PL1 and PL2 enables X-pointer to move back and forth position where string begins. Strings are separated by enter sequence (ASCII kód 13). Source code and diagram is presented below.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;     1DT301, Computer Technology I
;     Date: 2016-10-03
;     Author:
;     Songho Lee
;     Sarpreet Singh
;
;
;     Lab number: 5
;     Title: Task 4
;
```

```
;     Hardware: STK600, CPU ATmega2560, JHD202
;
;     Function: Display text on JHD202
;
;     Input ports: TX,RX on PIND2,PIND3
;                  SW0 on PIN0 (External Interrupt)s
;
;     Output ports: JHD2012 on PORTE.
;
;     Subroutines: If applicable.
;     Included files: m2560def.inc
;                          Initialize display by Anders Haggren
;
;     Other information:
;     (run @ 1.8432 MHz clk frequency)
;     Changes in program: (Description and date)

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

.include  "m2560def.inc"
.def Temp = r16
.def Data = r17
.def RS   = r18
.def      temp_usart = r19
.def PL1H = R21
.def PL1L = R20
.def PL2H = R23
.def PL2L = R22

.equ BITMODE4  = 0b00000010        ; 4-bit operation
.equ CLEAR     = 0b00000001             ; Clear display
.equ DISPCTRL  = 0b00001111        ; Display on, cursor on, blink on.
.equ      UBRR_choice = 47
.equ OURMEMORY = 0x200

.cseg
.org 0x0000                   ; Reset vector
     jmp reset

.org URXC1addr
     rjmp readChar
.org INT0addr
     rjmp outLCD

.org 0x0072

reset:

     ldi Temp, HIGH(RAMEND)   ; Temp = high byte of ramend address
     out SPH, Temp            ; sph = Temp
     ldi Temp, LOW(RAMEND)    ; Temp = low byte of ramend address
     out SPL, Temp            ; spl = Temp

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

```
     ;; Enable interrupt for USART
     ldi temp, UBRR_choice
     sts UBRR1L, temp
     ldi temp, (1<<TXEN1) | (1<<RXEN1) | (1<<RXCIE1)
     sts UCSR1B, temp

     ;;   Enable interrupt for SW0
     ldi temp,0b1
     out EIMSK, temp
     ldi temp, 0b10
     sts EICRA, temp


; **
; ** 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

sei  ;Enable global interrupt

ldi YH, HIGH(OURMEMORY)
ldi YL, LOW(OURMEMORY)

ldi XH, HIGH(OURMEMORY)
ldi XL, LOW(OURMEMORY)

mov PL1H, XH
mov PL1L, XL
mov PL2H, XH
mov PL2L, XL

rcall clr_disp


loop:

     rjmp loop           ; loop forever


;***************SUB ROUTINES*********************
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, 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

readChar:
    lds temp_usart, UCSR1A
    lds Data, UDR1

    st Y+, data
    reti

customDelay:

    push temp
    ldi temp, 1

    customloop1:
    inc temp
    rcall power_up_wait
    cpi temp, 100
    brne customloop1

    ldi temp, 1
    customloop2:

    inc temp
    rcall power_up_wait
    cpi temp, 100
    brne customloop2

    ldi temp, 1
    customloop3:

    inc temp
    rcall dbnc_wait
    cpi temp, 100
    brne customloop3

    pop temp

    ret


printoneline:
    ;COMPARARE BEFORE IT GETS MEMORY OUT OF BOUNDARY
    cp YH, XH
    brne doprint
    cp YL, XL
    breq stopprinting
```

```
    ld data, X+
    cpi Data, 13    ;New line sequence
    breq stopprinting

    doprint:
    rcall write_char
    rjmp printoneline
    stopprinting: nop
    ret

outLCD:

    rcall printoneline  ;PRINT LINE ONE
    rcall customdelay

    ldi R24, 0
    lcdloop:
    inc R24
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    rcall clr_disp
    rcall printoneline  ;PRINT current line

    ;
    mov PL2H, XH
    mov PL2L, XL

    ;NEW LINE
    ldi data, 0xA8
    rcall write_cmd
    rcall long_wait
    ;END NEW LINE

    mov xh, PL1H
    mov XL, PL1L
    rcall printoneline  ;PRINT previous line
    mov PL1H, XH
    mov PL1L, XL
    mov XH, PL2H
    mov XL, PL2L
    rcall customdelay

    cpi R24 ,3
    brne lcdloop

    quitlcd:
    rcall clr_disp

    ldi XH, HIGH(OURMEMORY)
    ldi XL, LOW(OURMEMORY)

    mov PL1H, XH
    mov PL1L, XL
    mov PL2H, XH
    mov PL2L, XL

    rjmp outLCD
    reti
```

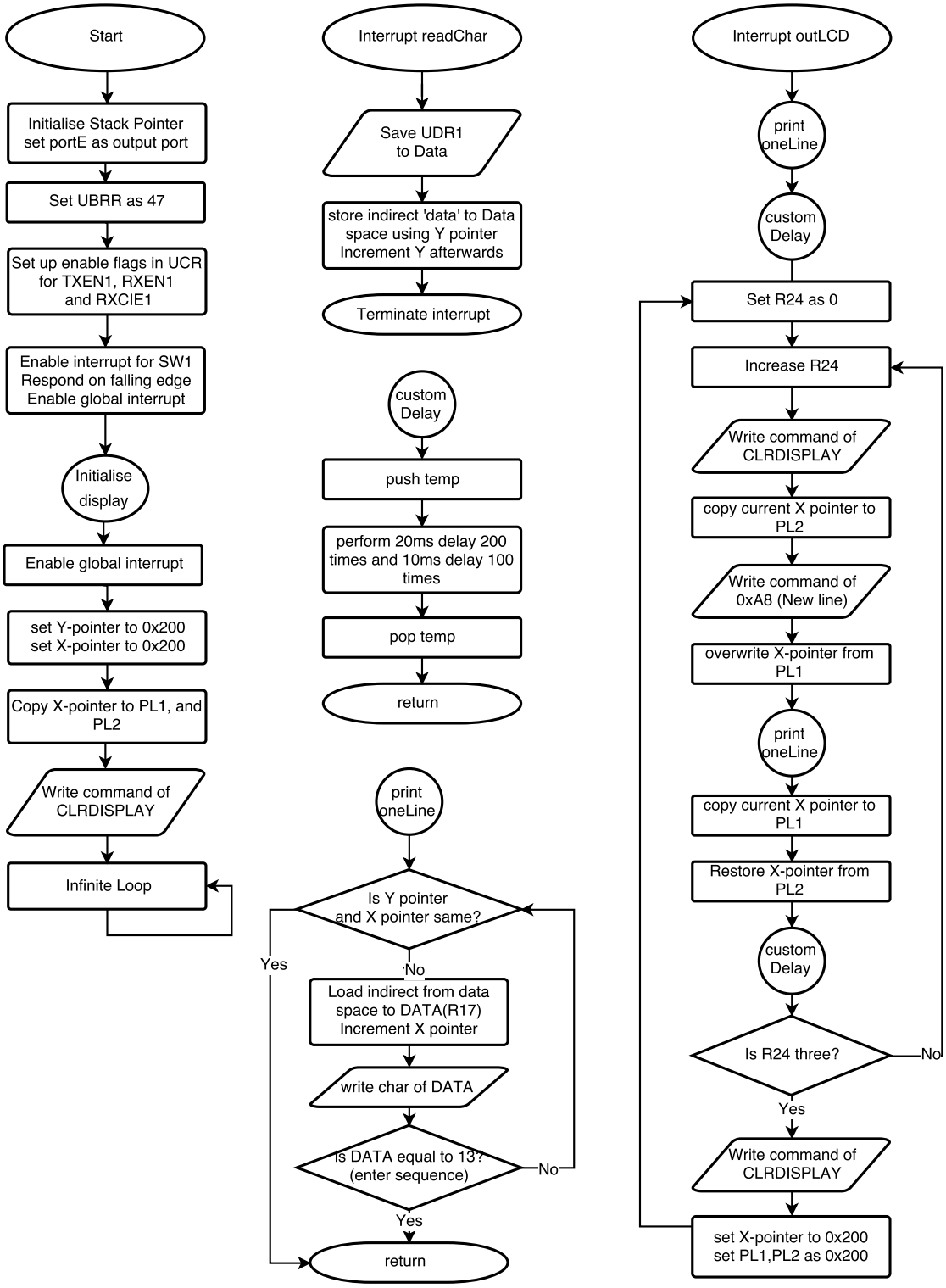

Figure 4: Flowchart of task 4