lab5

lh223ng och fe222pa

November 2021

1 task1 and task2

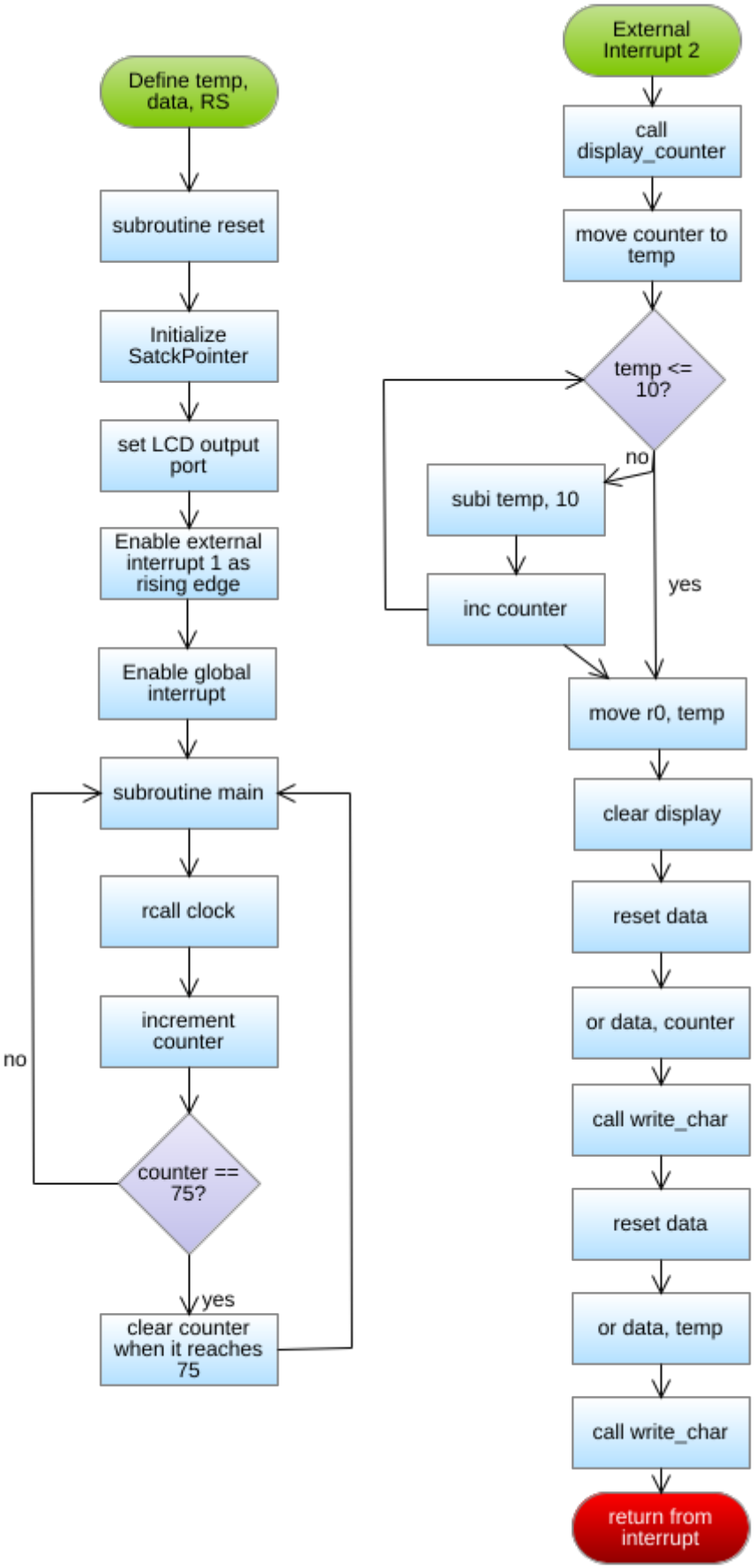
```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2021-09-18
; Author:
; Student name Li Ang Hu
; Student name Fredric Eriksson Sep lveda
;
; Lab number: 5
; Title: task1 and task2
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Create an electronic bingo generator. The generator shall create random number
The numbers shall be shown on the display. Clear the display before a new value is displaye
an interrupt and a push-button for the input.

; Input ports: PORTD2
;
; Output ports: PORTE(LCD_PORT)
;
; Subroutines: If applicable.
; Included files: m2560def.inc
;
; Other information:
;
; Changes in program: (Description and date)
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


; Replace with your application code
.def temp = r16
.def data = r17
.def RS = r18

.equ BITMODE4 = 0b0000_0010
.equ CLEAR = 0b0000_0001
.equ DISPLAY_CTRL = 0b0000_1111          ; Display on, cursor on, blink on.
.equ RS_ON = 0b0010_0000
.equ LCD_PORT = PORTE                    ; Port LCD is connected to
.equ DATA_DIR = DDRE                     ; Data dir. of port LCD is connected to
.equ COUNTER_MAX = 75

.cseg
.org 0x00
jmp reset
```

```
.org INT2addr
jmp interrupt2_handler

.org 0x72

.def COUNTER = r19 ; registers 16 to 18 are already in use in common.inc

reset:
; Initialize stack pointer
ldi temp, low(RAMEND)
out SPL, temp
ldi temp, high(RAMEND)
out SPH, temp

/*; Setup PORTD as input
clr temp
out DDRD, temp*/
ser temp
out DDRB, temp

; turn on two lights
ldi temp, 0b1100
mov r1, temp
mov r2, temp

; set LCD output port
ser temp ; Loads $FF directly to register temp
out DATA_DIR, temp ; PORTE for output


; Enable external interrupt 1 as rising edge
ldi temp, (1<<INT2)
out EIMSK, temp
ldi temp, 0b0010_0000
sts EICRA, temp

sei ; Enable interrupts

call init_display

main:
        rcall clock
        nop
        nop
        nop
rjmp main

;
; This functions as clock
; Purpose: increase counter in a loop.
;
clock:
        inc counter

        ; Clear the counter when it has reached COUNTER_MAX, 75
        cpi counter, COUNTER_MAX
        brlo skip0
                clr counter
        skip0:
        ret
```

```
;
; interrupt1_handler
; Purpose: Calls display_counter
;
interrupt2_handler:
        rcall display_counter
        eor r1, r2;
        out portb, r1
reti

;
; display_counter
; Purpose: Displays the value of 'counter' on the display.
;
display_counter:
        ; Prepare registers
        push temp
        mov temp, counter
        push counter
        clr counter



        ; Seperate tens to counter, ones to temp. Via substraction by 10
        L0:
        cpi temp, 10            ; while temp >= 10
        brlo E0
                subi temp, 10
                inc counter     ; increment for N tens removed from temp
                rjmp L0
        E0:

        mov r0, temp ; store temp in r0
        ; Clear the display
        call clear_display


        ; Display Tenth place
        ldi data, 0b0011_0000
        or data, counter
        call write_char

        ; Display Units place
        ldi data, 0b0011_0000
        ;inc temp
        or data, r0
        call write_char

        ;out PORTB, r0  ; for debugging purposes

        ; Restore registers
        pop counter
        pop temp
ret

; Display subroutines
init_display:
    rcall power_up_wait                 ; Wait for display to power up

    ldi data, BITMODE4                  ; Set 4-bit operation
    rcall write_nibble                                  ; (in 8-bit mode)
```

```
    rcall short_wait                                    ; wait min 39 us
    ldi data, DISPLAY_CTRL                              ; diaply on. blink on. cursor on.
    rcall write_cmd                                             ; send command
    rcall short_wait                                    ; wait min 39 us

clear_display:
    ldi data, CLEAR                                             ; clr diplay
    rcall write_cmd                                             ; send command
    rcall long_wait                                             ; wait min 1.53 ms
    ret

; Write subroutines
write_char:
    ldi RS, RS_ON                                               ; RS = high
    rjmp write

write_cmd:
    clr RS                                                              ; RS = low

write:
    mov temp, data                                              ; copy data
    andi data, 0b1111_0000          ; mask out high nibble
    swap data                                                   ; swap low nibble with high
    or data, RS                     ; Add RS to command to write
    rcall write_nibble              ; send high nibble
    mov data, temp                                              ; restore data
    andi data, 0b0000_1111          ; mask out low nibble
    or data, RS

write_nibble:
    rcall switch_output                                 ; modify for diplay JHD202A. port
    nop                                                                 ; wait 542 ns
    sbi LCD_PORT, 5                                             ; enable high JHD202A
    nop
    nop                                                                 ; wait 542 ns
    cbi LCD_PORT, 5                                             ;enable low JHD202A
    nop
    nop                                                                 ; wait 542 ns
    ret

; Wait subroutines
short_wait:
    clr ZH
    ldi ZL, 30                      ; wait 542 ns
    rjmp wait_loop
long_wait:
    ldi ZH, HIGH(1000)
    ldi ZH, LOW(1000)           ; wait 2 ms
    rjmp wait_loop
dbnc_wait:
    ldi ZH, HIGH(4600)
    ldi ZL, LOW(4600)           ; wait 10 ms
    rjmp wait_loop
power_up_wait:
    ldi ZH, HIGH(9000)          ; wait 20 ms
    ldi ZL, LOW(9000)

wait_loop:
    sbiw Z, 1                       ; 2 cycles
    brne wait_loop                  ; 2 cycles
    ret
```

```
; Modify output to fit LCD JHD202C
switch_output:
    push temp
    clr temp

    sbrc data, 0                    ; If D4 == 1?
        ori temp, 0b0000_0100       ;     then set PIN3
    sbrc data, 1                    ; If D5 == 1?
        ori temp, 0b0000_1000       ;     then set PIN4
    sbrc data, 2                    ; If D6 == 1?
        ori temp, 0b0000_0001       ;     then set PIN0
    sbrc data, 3                    ; If D7 == 1?
        ori temp, 0b0000_0010       ;     then set PIN1
    sbrc data, 4                    ; If E == 1?
        ori temp, 0b0010_0000       ;     then set PIN5
    sbrc data, 5                    ; If RS == 1?
        ori temp, 0b1000_0000       ;     then set PIN7

    out LCD_PORT, temp
    pop temp
    ret
```


Define temp, data, RS
subroutine reset
Initialize SatckPointer
set LCD output port
Enable external interrupt 1 as rising edge
Enable global interrupt
subroutine main
rcall clock
increment counter
counter == 75?
no
yes
clear counter when it reaches 75
External Interrupt 2
call display_counter
move counter to temp
temp <= 10?
no
subi temp, 10
inc counter
yes
move r0, temp
clear display
reset data
or data, counter
call write_char
reset data
or data, temp
call write_char
return from interrupt

2 task3 and task4

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2021-09-18
; Author:
; Student name Li Ang Hu
; Student name Fredric Eriksson Sep lveda
;
; Lab number: 5
; Title: task3 and task4
;
; Hardware: STK600, CPU ATmega2560
;
; Function: program modules from Assignment 4 and write a program that receives a characte
port and displays each character on the display.
2 lines of text can be displayed. Each text-line shall be
displayed during 5 seconds, after that the text on line 1 should be moved to line 2 and so
text shall be entered from the terminal program, e.g. PUTTY, via the serial port.
STK600

; Input ports: PD3 and PD2(USART1)
;
; Output ports: PORTE
;
; Subroutines: If applicable.
; Included files: m2560def.inc
;
; Other information:
;
; Changes in program: (Description and date)
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<



; Replace with your application code
;;LCD definations
.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.
.equ    lcd_SetCursor2          = 0b11000000    ; set cursor position  to line 1 address
.equ    lcd_SetCursor1          = 0b10000000    ; set cursor position  to line 2 address

;; serial communcation definations
.equ    FCLOCK  =       1000000
.equ    BAUD    =       4800
.equ    MYUBRR  =       (FCLOCK/8/BAUD)-1


.cseg
.org    0x0000                          ; Reset vector
        jmp reset


.org    0x0048                  ; USART0, Rx Complete
```

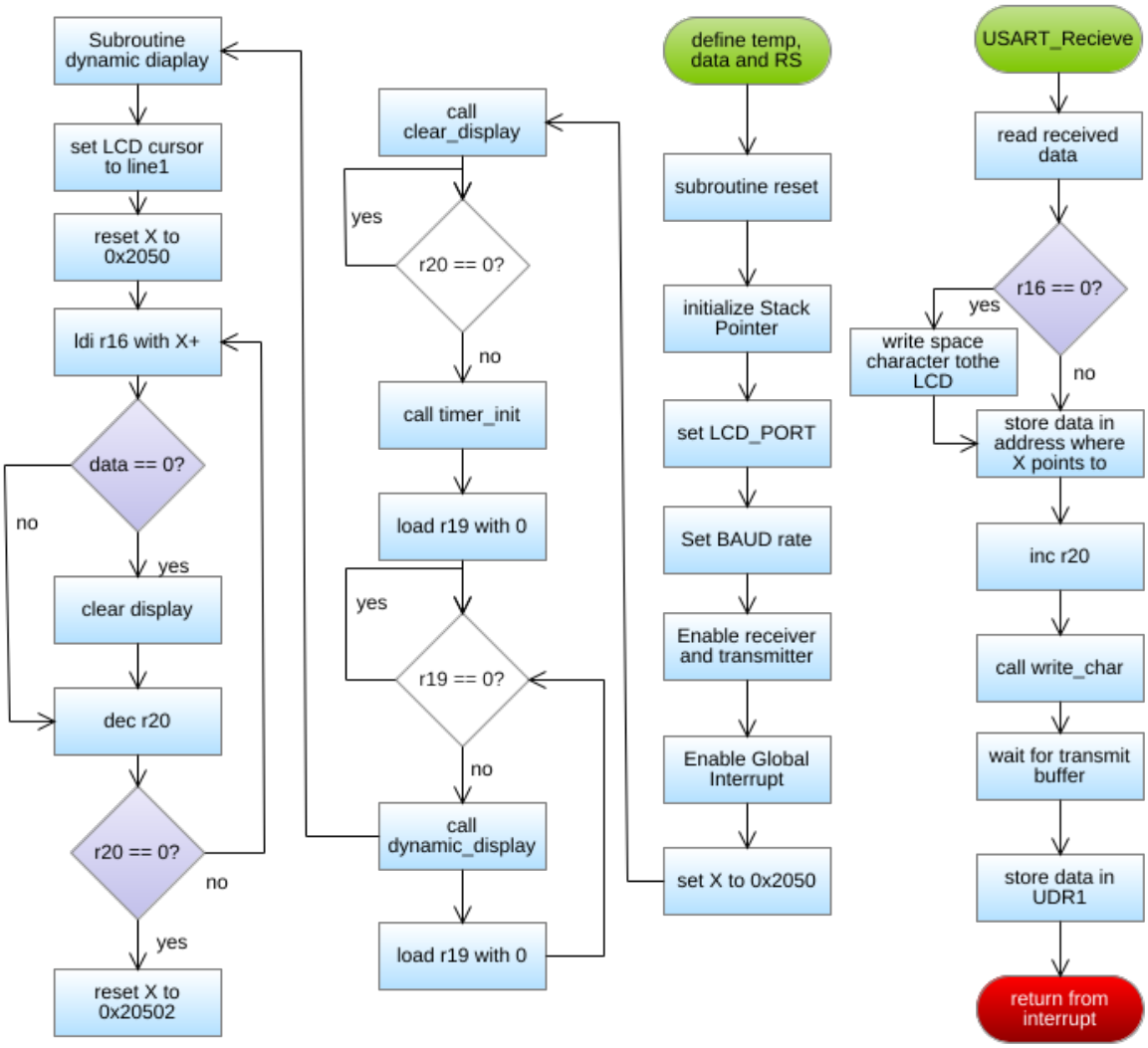
```
                jmp isr_URXC1

.org     0x004A                 ; USART0 data register Empty
                jmp isr_UDRE1


.org     OVF1addr               ; Timer/Counter1 Overflow
                 jmp Timer1_isr
.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 ddrb, temp          ; PORTB as OUTPUT
        out DDRE, temp                  ; port E = outputs ( Display JHD202A)
        clr temp                                ; r16 = 0
        out PORTE, temp
        call init_display                       ; initialize the lcd in 4 bit mode
        call init_uart

       call clr_display
       ldi r20,0 ; count
       LDI XL, 0x50                     ;the low byte of address of RAM
       LDI XH, 0x20                     ;the high byte of address of RAM

    loop1:call clr_display
      cpi r20,0
      breq loop1
      call timer_init
      ldi r19,0

  loop:
      cpi r19,0
      breq loop
      call dynamic_display
      ldi r19,0

    rjmp loop                   ; loop forever


;-----------------------------------------------------------
; **
; ** init_display
; **
init_display:
        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 usinit_display

        ldi data, 0x28                  ; 4-bit operation
        rcall write_cmd                 ; send command
        rcall short_wait

        ldi data, DISPCTRL              ; disp. on, blink on, curs. On
```

```
        rcall write_cmd                 ; send command
        rcall short_wait                ; wait min. 39 us

        ldi data, 0x06                  ; clear display
        rcall write_cmd                 ; send command
        rcall short_wait

clr_display:
        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


        ;configure usart1
init_uart:
        ldi     r17,HIGH(MYUBRR)        ; UBRR0H
        ldi     r16,LOW(MYUBRR)         ; UBRR0L
        sts     UBRR1H,r17                      ; UBRR0H
        sts     UBRR1L,r16                      ; UBRR0L
        ldi r16,(1<<U2X1)               ; U2X0 = 1
        sts     UCSR1A,r16                                                      ;
        ldi r16,(1<<RXEN1)|(1<<TXEN1)|(1<<RXCIE1)|(1<<UDRIE1)           ; enable receiver,
        sts     UCSR1B,r16                                                      ;
        ldi r16, (1<<UCSZ10)|(1<<UCSZ11)                                        ; Set frame format
        sts UCSR1C,r16                                          ;
        clr             r8                                      ; databuffer
        clr             r9                                      ; databuffer flag, 0=nodata, 255=h
        sei                                                     ; enable interruptions
        ret



isr_URXC1:
        ; Wait for data to be received
        lds r17, UCSR1A
        sbrs r17, RXC1 ; Skip if Bit in Register Set
        rjmp isr_URXC1
        ; Get and return received data from buffer
        lds             r16,UDR1                                                        ; read the
        out portb, r16                                                          ; debug purpose
        cpi r16, 0
        breq writeSpace

        ST X+, R16                                                              ;store in
        inc r20                                                                 ; incremen
```

```
        mov data,r16                                                    ;display l
                mov r6, data
        call write_char                                                 ;call data

        USART_Transmit:
                ; Wait for empty transmit buffer
                lds temp, UCSR1A
                sbrs temp, UDRE1
                rjmp USART_Transmit
                ; Put data (r16) into buffer, sends the data

                sts UDR1, r6

        lds             r17,UCSR1A                                      ; read UCS
        bst             r17,UDRE1                                       ; UDR0 emp
        brtc    send01                                          ; non empty, then
        tst             r9                                                      ;
        brne    send01                                          ; if has data send

send01:
        reti                                                            ; return fr

writeSpace:
        ldi r16, 0b0010_0101 ; # character ASCII number 35
        ;call write_char
        ret

isr_UDRE1:
        sbrs    r9,0                                                    ;
        reti                                                            ; return fr
        sts             UDR1, r8                                        ; send dat
        clr             r9                                                      ;
        reti                                                            ; return fr

dynamic_display:
        cli
        call clr_display
        ldi data, lcd_SetCursor2        ; set lcd cursor to line1 to display received byte
        rcall write_cmd                 ; send command
        rcall short_wait                ; wait min. 39 us

        LDI XL, 0x50                    ;the low byte of address of RAM
        LDI XH, 0x20                    ;the high byte of address of RAM

rxloop2:
        LD  R16,X+                      ;load r16 from the byte stored from 0x2050 address
        mov data,r16                    ;

        cpi data, 0 ; check whether data is empty
        breq clear_display
        jmp else

        clear_display:
                call    clr_display         ; clear LCD Screen
                ldi data, 0b0010_0000   ; Write Character Space to the LCD screen
                call write_char                 ;call data write function
                jmp restCode
        else:
                call    write_char              ;call data write function

        restCode:
```

```
        DEC R20                         ; read all the characters of line1
        BRNE rxloop2                    ; display till all characters over

/*      ; test purppose
    ldi data, 0b0010_0101       ; Write Character % to the LCD screen
        call write_char                 ;call data write function*/

        ldi data, lcd_SetCursor1        ; set lcd cursor to line1 to display received byte
        rcall write_cmd                 ; send command
        rcall short_wait                ; wait min. 39 us
        LDI XL, 0x50                    ;the low byte of address of RAM
        LDI XH, 0x20                    ;the high byte of address of RAM
        ldi r20,0
        sei
        ret


timer_init:

  ldi r19,(1<<CS12);              U2X0 = 1
  sts   TCCR1B,r19 ;   The prescaler value is 256 therefore timer frequency is 1 MHz/ 256 =

  ldi r19,(1<<TOV1);
  sts    TIMSK1,r19 ;   TIMSK1|=       //Enable timer overflow interrupt

 ldi r19,(1<<ICF1)|(1<<OCF1C)|(1<<OCF1B)|(1<<OCF1A)|(1<<TOV1);  //Clear all interrupt flags
 sts TIFR1,r19


ldi    r17,0xb3
ldi    r16,0xb5
STS    TCNT1H,r17
STS    TCNT1L,r16

 sei
 ret


 Timer1_isr:
         cli
         ldi r19,1
  ldi    r17,0xb3
ldi    r16,0xb5
STS    TCNT1H,r17
STS    TCNT1L,r16

        sei

         reti
```


Subroutine dynamic diaplay
set LCD cursor to line1
reset X to 0x2050
ldi r16 with X+
data == 0?
no
yes
clear display
dec r20
r20 == 0?
no
yes
reset X to 0x20502
call clear_display
yes
r20 == 0?
no
call timer_init
load r19 with 0
yes
r19 == 0?
no
call dynamic_display
load r19 with 0
define temp, data and RS
subroutine reset
initialize Stack Pointer
set LCD_PORT
Set BAUD rate
Enable receiver and transmitter
Enable Global Interrupt
set X to 0x2050
USART_Recieve
read received data
r16 == 0?
yes
write space character tothe LCD
no
store data in address where X points to
inc r20
call write_char
wait for transmit buffer
store data in UDR1
return from interrupt