24.05.2021 16:15

EMISY LAB 3

Krzysztof Piotrowski
Index. No.: 300175

Task 1.

The main goal of this task was to display my name on LCD display when chosen switch (in that case I used switch SW7) is pressed. The code is following:

```
; Save to RAM
        mov 30H, #'K'
        mov 31H, #'R'
        mov 32H, #'Z'
        mov 33H, #'Y'
        mov 34H, #'S'
        mov 35H, #'Z'
        mov 36H, #'T'
        mov 37H, #'O'
        mov 38H, #'F'
        mov 39H, #0

; creating labels for better management of LCD pins

        LCD_BUS equ P1
        LCD_RS equ P3.1
        LCD_E equ P3.2
        SWITCH equ P2.7

; main routine to call other routines in one place
main:
        mov R0, #30
        call ms_delay     ; waiting for 30 ms

        call initialization ; initialize display

display_off:
        jb SWITCH, $ ; wait here until switch change it state
display_on:
        mov R1, #30H
        call send_data ; display index number

        mov R0, #39 ; wait for more than 39 us
        call us_delay

        jnb SWITCH, $ ; wait here until switch change it state
        call clear_display ; routine for clearing display
        jmp display_off ; back to the label display_off
```

```
; routine for whole proces of initiatlization
initialization:

            clr LCD_RS ;instruction are sent to the module

; function set
            mov LCD_BUS, #00111000B ; set proper values for 1 line, default font

            call send_command ; negative edge on E to perform action

            mov A, #39 ; wait for 39 us according to algorithm
            call us_delay

; display on/off control
            mov LCD_BUS, #00001110B ; turn on display and cursor

            call send_command ; negative edge on E to perform action

            mov A, #39 ; wait for 39 microseconsd according to algorithm
            call us_delay

; entry mode set
            mov LCD_BUS, #00000110B ; mode set to inc. address by one and shift cursor to the right

            call send_command ; negative edge on E to perform action

            mov R0, #39 ; wait for more than 39 us
            call us_delay

            ret
; routine for uppering and lowering LCD_E, making it to be on negatiev edge
send_command:
            setb LCD_E
            clr LCD_E

            ret
; routine for sending a letters to display
send_data:
            setb LCD_RS ; change mode to sending data
loop:
            mov A, @R1
            jz finish
            mov LCD_BUS, A

            call send_command ; negative edge on E to perform action

            mov R0, #39 ; wait for more than 39 us
            call us_delay

            inc R1
            jmp loop
finish:
            ret
```

```
; routine for clearing display
clear_display:
        clr LCD_RS ;instruction are sent to the module
        mov LCD_BUS, #00000001B ; clear display

        call send_command ; negative edge of pin E
        mov R0, #2 ; wait approx. 2 ms for clearing display
        call ms_delay

        ret
; routine to create multiple delay of 1us
us_delay:
        mov A, R0
        rr A ; divide delay by two as djnz instruction take 2 us to execute
        subb A, #5 ; correct delay as each instruction (despite djnz) takes time to execute
        mov R0, A
        djnz R0, $

        ret
; routine to create multiple delay of 1ms
ms_delay:
        ms_one:
                mov R1, #10
        ms_two:
                mov R2, #48 ; as other instruction takes time to execute, 48 instead of 50 is used
                nop ; and NOP instruction for some correction, overall 0.03% of error for ms_delay is
applied
        djnz R2, $
        djnz R1, ms_two
        djnz R0, ms_one

        ret
```

Code Description:

Following code is remake of the code from first laboratory. Program consists of initialization routine, long and short delay routines and sending data routine. It starts with saving my name to the RAM and assigning labels to proper pins: RS, E, whole data bus and pin responsible for switch. After 30 ms of delay, which is needed to turn on display, the initialization process starts. During that process all instructions are sent to make display work properly. When initialization is finished we check switch state. By default in EDSIM51 switch logical value is set to ‘1’. If this value is maintained we wait in that condition until user will press the switch which will change its value. Then, due to change it will go to next instructions as condition will be no longer fullfilled. Now LCD Display will show my name as long as switch logical value will remain ‘0’. If the change will occur, we go to ‘clear display’ routine. In that routine display goes into instruction mode and clears the display. Next it waits for 2 ms to apply changes. After that we again check the logical value of switch SW7 and wait for next user move. All instructions will be again executed as long as user will press the switch button.

Task 2.

The main goal of this task was to turn on led when any key was pressed on keypad. The code is following:

```
LED_BANK equ P1 ; label for LEDs
KEYBOARD equ P0 ; label for keyboard

main:

mov KEYBOARD, #11110000B ; set all rows to '0'
jmp init ; initialize interrupt INT1

org 013H ; interrupt address

xrl LED_BANK, #10100101B ; switch on/off 0 2nd 5th and 7th LED
reti ; back to infinite loop

init:
setb EA ; enable global interrupt
setb EX1 ; enable INT1 interrupt
setb IT1 ; make INT1 work with falling edge

jmp $ ; infinite loop until interrupt occurs
```

Code Description:

At the beginning labels were assigned for both Bank of LED and keyboard. In EDSIM51 all keypad's rows and columns are by default set to logical value '1' so the AND gate which is connected to keypad has also value '1'. The AND gate mentioned before is connected to pin 3.3 which is responsible for interrupt 1. When program starts, all rows are cleared. Thanks to that when user will press any key, the result of AND gate will change to '0'. After clearing rows we jump to initialization process of INT1. Here, both INT1 and global interrupts are enabled by setting up proper bits and also we make INT1 to work with falling edge of input signal. After that process we are in infinite loop until interrupt occurs. So when user will press any key, we leave the loop and go to interrupt address. I decided to turn on/off zero, second, fifth and seventh LEDS of LED bank alternately with each key press. At the end we go back to infinite loop and wait for user move. Depending on keypad mode: if keypad works in standard mode, only „pressed in" key will cause changes so in practise we will make changes every second press. In pulse mode each press of key will provide changes.

I declare that this piece of work which is the basis for recognition of achieving learning outcomes in the EMISY course was completed on my own.

Krzysztof Piotrowski
300175
24.05.2021