Using LCD display and LEDs with keyboard inputs in EdSim simulator – an EMISY laboratory

Maciej Marcinkiewicz (300371)

31st May 2021

1 Introduction

1.1 Brief description

Laboratory's main purpose was to learn how to use switches and keyboard matrices with microcontrollers and its peripherals. Once again tasks had to be done in the EdSim5 simulator.

1.2 Hardware description

Microcontroller (Atmel's AT89C4051) is connected to 5 V source and is equipped with 12 MHz clock and simple resetting circuit. LCD screen's data bus is connected to P1 port. Display's RS and E pins are connected to P3.0 and P3.1 pins, respectively. Switch button SW0 from switch bank is connected to P2.0 pin.

2 Task 1

2.1 Assembly code

```
; define constants:
; LCD_BUS is LCD's data bus
; LCD_RS is LCD's R/S pin
; LCD_E is LCD's E pin
; SW is pin connected to switch from the switch bank

LCD_BUS equ P1
LCD_RS equ P3.0
LCD_E equ P3.1
SW equ P2.0

mov R0, #20 ; initialization - wait for approx. 30 ms (or even slightly more)

initial_wait:
lcall long_delay ; call function which waits for more than 1.53 ms
djnz R0, initial_wait ; loop that will perform actions 10 times

mov LCD_BUS, #00111000B ; function set
lcall send_command
lcall short_delay ; wait for more than 39 us
```

```
mov LCD_BUS, #00001110B ; turn display on
lcall send_command
lcall short_delay ; wait for more than 39 us

mov LCD_BUS, #00000110B ; entry set mode
lcall send_command
lcall short_delay ; wait for more than 39 us

not_pressed:
mov LCD_BUS, #00000001B
lcall send_command
lcall short_delay ; wait for more than 39 us
jb SW, $ ; if button is not pressed stay here with blank display else go further to
display name

mov LCD_BUS, #'M' ; write letter "M"
lcall send_data
lcall long_delay ; wait for more than 1.53 ms

mov LCD_BUS, #'a' ; write letter "a"
lcall send_data
lcall long_delay ; wait for more than 1.53 ms

mov LCD_BUS, #'c' ; write letter "c"
lcall send_data
lcall long_delay ; wait for more than 1.53 ms

mov LCD_BUS, #'i' ; write letter "i"
lcall send_data
lcall long_delay ; wait for more than 1.53 ms

mov LCD_BUS, #'e' ; write letter "e"
lcall send_data
lcall long_delay ; wait for more than 1.53 ms

mov LCD_BUS, #'j' ; write letter "j"
lcall send_data
lcall short_delay ; wait for more than 39 us

jnb SW, $ ; stay here if button is pressed, else go to clear it
jmp not_pressed

send_command: ; send command from LCD's data bus
clr LCD_RS ; set mode to command mode
setb LCD_E ; get data from bus
clr LCD_E
ret

send_data: ; send character data from LCD's data bus
setb LCD_RS ; set mode to write mode
setb LCD_E ; get data from bus
clr LCD_E
```

```
ret

long_delay: ; wait for approx. 1.53 ms: (256 us * 2 + 1 us) * 3 + 2 us + 3 us + 1 us
mov R1, #3

inner_loop_long_delay:
mov R2, #256
djnz R2, $ ; loop that will perform actions 256 times

djnz R1, inner_loop_long_delay ; loop that will perform actions 3 times
ret

short_delay: ; wait for approx. 39 us: 1 us + 18 us * 2 + 2 us
mov R0, #18
djnz R0, $ ; loop that will perform actions 18 times
ret
```

2.2 Code description

Part of description is from the lab 1
Program starts with constants definition. In order to make the code more readable, pins has been assigned to mnemonic names. Then the main routine starts with display initialization. Assigning immiediate value to the register and decrementing it makes a simple loop which is required for delay. In the beginning microcontroller has to wait for display at least 30 miliseconds (in order to get proper voltage on display). Program calls `long_delay` subroutine 20 times in order to wait this amount of time.

`long_delay` decrements number 256 to zero three times. It gives appoximately 1.53 ms (even more than this). This number is not coincidence – such delay will be used later when several characters will be printed on display.

After 30 ms of delay data bus is filled with value responsible for function set. Display has to be set to two line mode. Then command is being sent by `send_command` subroutine. It clears RS pin as command is being sent, not data about character. In the end it sets and then clears E pin. It is a signal for display to get data from data bus. The last step is a short delay before the next command. `short_delay` subroutine is even simpler than `long_delay`. It only decrements to zero value of 18. It results in delay of more than 39 μs which is necessary after executing each display's command.

The next command is responsible for turning on the display and showing cursor. It is executed in the same manner as function set. The last step in initialization process is entering set mode. It is set to increment mode, so characters are appearing one after the other.

When initialization is finished, program clears the screen and waits in the loop until the push button is being pressed. When button is pressed, pin is grounded, so program has to exit the loop when SW bit is cleared.

If program exits the loop, it is writing letters into display's DDRAM (staring from the first postiton of the first line). Letter's code is written to the data bus and `send_data` is called. It works almost in the same way as `send_command`, the only difference is that it sets R/S pin to turn on writing mode.

After printing all letters, program stays in the another loop. This time it waits for setting the SW pin which is of course being set after releasing the button. When program exits the loop it goes back to not_pressed label when screen is cleared and once again waits for clearing the SW pin.

3 Task 2

3.1 Assembly code

```
; define constants:
; MATRIX_BUS is a port connected to the keyboard matrix's pins
; LED is the first LED from the LED bank

MATRIX_BUS equ P0
LED equ P1.0

jmp init ; jump to the initialization

; interrupt handler
org 013h ; write instruction to 0xB address (interrupt handler)
xrl LED, #1 ; use xor on LED pin to change the bit's value to the opposite one
reti ; return to the infinite loop

init:
; global interrupt initialization
setb EA ; enable global interrupts
setb EX1 ; enable INT1 interrupt
setb IT1 ; set INT1 to work with falling edge of the input signal

mov MATRIX_BUS, #01110000B ; clear all row bits in order to make keyboard working

jmp $ ; infinite loop waiting for interrupt
```

3.2 Code description

In the beginning macros are being set up and program immiediately jumps to initialization subroutine. It enables global interrupts and enables INT1 external interrupt, whose pin is connected to the output of AND gate. Gates inputs are matrix's column pins. Finally, INT1 is set to detect falling edge and matrix's port is set in a way to clear all row pins. Without the last step keyboard matrix would not work properly.

Then program waits for an interrupt in the infinite loop. Interrupt happens when any key is pressed. In the interrupt handler subroutine XOR operation with 1 is operated on LED pin. With this instruction every handler execution will alter the pin's value.

Declaration of authorship

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.