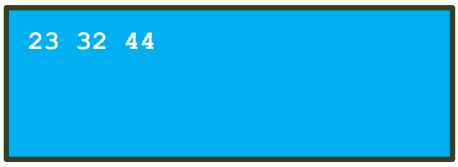
You are going to work with timers, interrupts, USART communication, EEPROM, LCD and the IO board. Please hand in only the **.c** program implementing the following tasks.

Tasks

a. You must use **Timer1** with interrupts such that you can track time with a resolution of 1 second. It is up to you how do you configure the prescaler and how many interrupts occur within 1 second, there must be a variable that gets updated exactly every 1 second.
Expected functionality: implement an LED toggling such as PB5 when 1 second passes. Or show the current seconds on the LCD, just for testing purposes.

b. Configure an USART Communication with 19200bps, 8 bit data, 1 stop bit, no parity. Use the serial port to send a byte from your computer in decimal format on 2 digits (values in decimal will be in range 10 to 49). Each time the microcontroller receives a message, it should be added to the LCD, without erasing the previous message.
Expected functionality: opening the serial communication at 19200 bps and sending the following numbers (in decimal) 23, 32 and 44 in RealTerm/Serial Terminal/CoolTerm will result in the following being shown on the LCD:



c. Consider the message received as having the form XY, where X corresponds to the LED being turned ON (1 - 4), while the Y represents how many seconds will the LED be ON (from 0 to 9). Every time you receive a message, turn the X led ON for Y number of seconds. Timer1 and interrupts from **task a** must be used to count the seconds. When the number of seconds is reached, the corresponding X LED should turn OFF and the program will wait for a new message.
Expected functionality:
- send number 23, LED 2 should turn ON for 3 seconds, then OFF, then wait
- send number 32, LED 3 should turn ON for 2 seconds, then OFF, then wait
- send number 44, LED 4 should turn ON for 4 seconds, then OFF, then wait
- and so on…

d. Add the functionality that you will store in the EEPROM the messages received. If you press Button 3 of the IO board, you should "playback" the received messages in such a way that you will turn the X LED on for exactly Y number of seconds according to the first message, then move on to the next message and reflect the pattern on the corresponding LED and so on, until all the received messages have been played back.

Continues on next page

Expected functionality:
EEPROM Contents: 23 32 44 255 255 255 255 255 …

Pressing button 3 will cause:
LED 2 should turn ON for 3 seconds, then OFF, then
LED 3 should turn ON for 2 seconds, then OFF, then
LED 4 should turn ON for 4 seconds, then OFF, then stop.

e. Add the functionality that pressing button 4 will clear the EEPROM content and clear the LCD. The button press must be configured with a **P**in **C**hange **Int**errupt (PCINT).
Expected functionality: After pressing button 4:
EEPROM Contents: 255 255 255 255 255 255 …

Hints:
- use functions if possible
- if you send messages outside the 10 - 49 range, the program should ignore the values and not show them on the LCD and not store them in the EEPROM.
- a message such as 10, 20, 30 or 40 is a valid message, but the LED does not really turn ON.
- use simple math operations to extract the LED and the number of seconds from the message (such as modulo or division)
- when you display the contents on the LCD, it is up to you if you want to add a space between the messages, or change to a new line when the current line is full
- if you cannot send messages in decimal (due to using CoolTerm or similar), convert messages into HEX and send them as HEX instead. For example, wanting to send 11, 22, 33 and 44 in decimal means you will send B, 16, 21 and 2C. The LCD will still show messages as: 11, 22, 33 and 44.
- Each byte of unwritten EEPROM Memory should have the value 255.