Department of Electrical and Computing Engineering

UNIVERSITY OF CONNECTICUT

ECE 3411 Microprocessor Application Lab: Fall 2015

Lab Test III

There are 2 longer programming problems in this test. There are 6 pages in this booklet. Answer each question according to the instructions given.

You have **100 minutes** to answer the questions. Once you are done, you need to show the output to the Instructor or TA and e-mail the code to the TA.

Some questions are harder than others and some questions earn more points than others—you may want to skim all questions before starting.

If you find a question ambiguous, be sure to write down any assumptions you make.
Be neat and legible. If we can't understand your answer, we can't give you credit!

Write your name in the space below. Write your initials at the bottom of each page.

THIS IS AN OPEN BOOK, OPEN NOTES TEST.
YOU CAN USE YOUR LAPTOP BUT PLEASE TURN YOUR NETWORK DEVICES OFF.

Any form of communication with other students is considered cheating and will merit an F as final grade in the course.

Do not write in the boxes below

1 (x/60)	2 (x/40)	3 (x/60)	Total (xx/160)

Name:

Student ID:

1. [60 points]:

a. Write a C program for your AVR that performs the following functions:

- It *always* blinks an LED at a constant frequency of 1Hz with 50% duty cycle.
- It is always ready to receive characters over UART and as soon as a character is received, it echoes/prints it back to the Terminal immediately without waiting for a return key (`'\r'`) or a next line character (`'\n'`).
- Once a return key (`'\r'`) or a next line character (`'\n'`) is received, it converts all the characters received so far (since last `'\r'` or `'\n'`) to lower caps and prints them on the Terminal. E.g. `'A'` becomes `'a'`, whereas `'b'` remains `'b'` and so on.

b. Extend Task1(a) such that your code now prints the received characters on the Terminal over UART as well as on the LCD.

- The first row of the LCD should be used to immediately print the received characters without waiting for a return key (`'\r'`) or a next line character (`'\n'`).
- The second row of the LCD should be used to print all the characters received so far (since last `'\r'` or `'\n'`) converted to lower caps.
- It should still *always* blink an LED at a constant frequency of 1Hz with 50% duty cycle.

c. Extend Task1(b) such that now you can choose to print the received characters either on UART or on LCD based on a push switch.

- Upon startup, the characters are printed on Terminal over UART.
- A button push toggles the printing between LCD and UART. Notice that a single button push must only toggle the printing mode once.
- It should still *always* blink an LED at a constant frequency of 1Hz with 50% duty cycle.

Initials:

2. [40 points]: In this task, you'll use the LCD as a small billboard for displaying (printing) different advertisements. The advertisement strings that need to be displayed are stored in the program memory and are as follows:

```
const uint8_t Google[]    PROGMEM = "I am Google";
const uint8_t Intel[]     PROGMEM = "I am Intel";
const uint8_t Microsoft[] PROGMEM = "I am Microsoft";
const uint8_t IBM[]       PROGMEM = "I am IBM";
const uint8_t NVIDIA[]    PROGMEM = "I am NVIDIA";
```

Using `Timer0`, implement a system that:

- Prints a new advertisement string on the first row of the LCD **every second**.
- The order in which the strings are printed should be as follows:
  (a) Intel
  (b) IBM
  (c) NVIDIA
  (d) Google
  (e) Microsoft

Note: You are not allowed to use `_delay_ms()`/`_delay_us()` function calls.

Hint: You can use blocking LCD functions from the LCD library for this task.

Initials:

3. [60 points]: In this task, we are going to measure the wasted time (in milliseconds) due to blocking function calls, and then demonstrate how non-blocking functions avoid such wastage. A layout of the code that measures and prints the wasted time is provided on the next page.
Notice that you are only allowed to use `Timer0` for this task, and `_delay_ms()`/`_delay_us()` function calls are not allowed (except for the ones already present in `lcd_lib.c`).

a. (15 points) Extend the provided C code layout to implement the following functionality using `Timer0`:

- Toggle a LED after every 1 second.
- Call the 'print_difference()' function every 10 seconds. This function prints over UART the milliseconds wasted (if any) in toggling the LED.

b. (15 points) Extend Task(a) to implement the following functionality using `Timer0`:

- Issue a ***blocking*** LCD refresh request after every 1 second. The LCD refresh request simply moves the cursor to position (0,0) and prints a character different than previous one.
- Call the 'print_difference()' function every 10 seconds. This function now prints over UART the milliseconds wasted (if any) in refreshing the LCD using blocking functions, and toggling the LED.
- **Hint:** The LCD library functions are blocking.

c. (30 points) Extend Task(a) to implement the following functionality using `Timer0`:

- Issue a ***truly non-blocking*** LCD refresh request after every 1 second. The LCD refresh request simply moves the cursor to position (0,0) and prints a character different than previous one.
- Call the 'print_difference()' function every 10 seconds. This function now prints over UART the milliseconds wasted (if any) in refreshing the LCD using non-blocking functions, and toggling the LED.
- **Hint:** An extended state machine for non-blocking LCD writes could help.

Note: If you are unable to solve part (a), you may ask for its solution from the TA in order to make progress with part (b) and (c). However, if you do so, you will get **zero** credit for part(a) and only **80%** credit at max for both part (b) and (c).

Initials:

Provided code layout to measure timing:

```
volatile int ctr1 = 0;
volatile int ctr2 = 0;
volatile uint8_t ctr1_flag = 0;
//-------------------------------------------------------------------------
// Timer 0 Compare Match A ISR
ISR (TIMER0_COMPA_vect)
{
    ctr2++;           // Increment counter 2
    ctr1_flag = 1;    // Set the flag for counter 1

    /* Your code for ISR goes here */
}
//-------------------------------------------------------------------------
// Prints the difference of two counters over UART
void print_difference()
{
    stop_timer0();       // Stop Timer 0 by clearing prescaler bits to zero
    printf("%i \n", ctr2 - ctr1);    // Printing the difference
    ctr1 = 0;    ctr2 = 0;           // Reseting the counters
    start_timer0();      // Start Timer 0 by setting selected prescaler.
}
//-------------------------------------------------------------------------
int main(void)
{
    initialize_all();    // Initialize everything
    sei();               // Enable global interrupts

    while(1)
    {
        /* Here issue a LED Toggling/LCD Refresh request every second */


        /* Incrementing counter 1 */
        if(ctr1_flag == 1)
        {
            ctr1++;
            ctr1_flag = 0;
        }

        /* Here call the 'print_difference()' function every 10 seconds */

    }
}
//-------------------------------------------------------------------------
```

Initials:

End of Quiz

Please double check that you wrote your name on the front of the quiz.

Initials: