Birzeit University - Faculty of Engineering and Technology
Electrical & Computer Engineering Department - ENCS4330
Real-Time Applications & Embedded Systems - 2nd semester - 2024/2025

Project #4
16F877A PICMicro programming under MPLAB
Due: June 22, 2025

Instructor: Dr. Hanna Bullata

Complex calculator

We would like to build a complex calculator that is able to perform *only* division mathematical operations on float numbers. We'll assume each float number will not exceed 1 million (10^6) in value and the decimal part will not contain more than 6 digits (e.g. `999999.999999`).

The hardware should be composed of 16F877A $\times$ 2 microcontrollers in addition to a push button P to enter the numbers and a 16 $\times$ 2 character LCD. We'll call the first microcontroller the `master CPU` and the second microcontroller the `auxiliary CPU` or `co-processor`. The push button and the character LCD are connected to the master CPU while the computation will take place on the co-processor.

The system should behave as follows:

1. When the system is powered up, the first line of the LCD should display the message `''Welcome to''` and the second line of the LCD should display the message `''Division''`.

   The above message blinks 3 times with a 0.5 second delay between blinks. After 2 seconds, we move to the next step.

2. The first LCD line displays the message `''Number 1''`. The cursor should then be positioned on the second LCD row. On the second line, we'll proceed to entering the first number digit by digit, starting by the most significant digit first.

   On the second LCD line, we'll have the cursor blinking on the location of the most significant digit: When we click on the push button P, the first digit of the first number should increment by 1. Since originally that digit is 0, the first click should make it 1, the second click should make it 2, the third click should make it 3, etc until it reaches the value 9. On the tenth click, that digit goes back to 0, and so on.

3. If we leave P unclicked for over 1 second, the first digit of the first number becomes fixed. Next, and in order to speed things up, we'll assume all the other digits of the number have the same value as the first digit (e.g. if the first digit is set to 6, we'll assume all the number is composed of 6 (`666666.666666`)).

   The cursor is currently blinking on the location of the next-to-most significant digit. That digit will increment with each click on the push button P until it reaches 9 and then back to 0, and so on.

   If the user is OK with current integer part of the first number, he/she can double-click on the push button P and the cursor will move to the decimal part immediately. Otherwise we repeat the same current step on all remaining digits of the integer part.

4. After entering the integer part of the first number, we proceed to entering the decimal part in the same fashion as described above: Once the first decimal number is set, the remaining 5 decimal digits will get the same initial value as the most significant digit of the decimal part (e.g. if the first digit of the decimal part is set to 4, we'll assume all the decimal part is composed of 4 (`.444444`).

If the user is OK with current decimal part of the first number, he/she can double-click on the push button P and we'll proceed to entering the second number. Otherwise, we'll proceed to setting each digit of the decimal part as described above.

5. The LCD screen should display on the first line the message ``‘‘Number 2’’`` for 1 second.

6. Next, we need to proceed to entering the second number digit by digit as we did for the first number.

7. Once the second integer number has been entered, the system should put on the LCD screen the sign ``‘‘=’’`` and then execute the division mathematical operation as follows:

  - The master CPU will send the first number (both integer and decimal parts) to the co-processor as a sequence of inputs as follows: First send the number of bytes with an interrupt and wait for acknowledgment from the co-processor. Then send the first number as successive bytes to the co-processor with interrupts and after each sending wait for the acknowledgment.
  - Afterwards, the master CPU will send the second number to the co-processor in a similar fashion.
  - The co-processor will divide the first number by the second number.
  - The co-processor will return the result of the division operation as successive bytes while using an interrupt with each sending and waiting for acknowledgment from the master CPU.
  - Once the master CPU gets the complete result from the co-processor, it should output the division result of the 2 numbers on the second row of the LCD screen. At the same time, the first LCD line displays the message ``‘‘Result’’``.

8. While the result is displayed per the last step, clicking on the push button P will display the first number on the first line of the LCD screen. Clicking P again will display on the first line of the LCD screen the second number, and so on.

9. If we double-click on P, a new division operation starts by going to step **2** above.

What you should do

- Use proteus application to build the schematic for the whole system, including the master CPU, the co-processor, the 16×2 character LCD and the push button P.
- Use PortC of the master CPU and the co-processor for data transfer back and forth between both processors.
- Use pin PortB.0 of the master CPU for the push button.
- Use PortD of the master CPU to connect the character LCD in 4-bit mode. Remember to pull up the RS pin of the LCD using a 4.7KΩ resistor.
- Remember to add a 10KΩ pull-up resistor to push button P (or use the internal weak pull-up resistor of PortB), add a 4MHZ oscillator (with $2 \times$ 15pF capacitors) and a 10KΩ pull-up resistor to the MCLR pin.
- Enable interrupts on push button P if you decide to use interrupts.
- Build the PIC assembly code for both the master CPU and the co-processor that implement the behavior described above under MPLAB IDE. You will have thus to build 2 projects, one for each processor.

- Assemble both projects and make sure you get a successful build on both. Use the simulator if you wish to make sure the behavior is correct.
- Send the zipped folder that contains the MPLAB code for both projects and the proteus schematic before the deadline. If the deadline is reached and you are still having problems with your code, just send it as is!