CS 431-1 Embedded Systems

HW 2

Opened: Wednesday, 29 September 2021, 12:00 AM
Due: Friday, 29 October 2021, 5:59 PM
To do: Make a submission

In this lab we're going to play around with SFRs and some standard peripherals of 8051. The overall behaviour is going to be a device that wait for a serial data (1 byte), waits 0-255 milliseconds with respect to the received byte and then, reads 3 digits from the keypad and prints the both read data to LCD as a HEX string (1 line each).

We will **NOT use any interrupt system** for this lab and will resort to polling. For implementing the periodicity, we will not use dumb-looping*(explained later)*, but timers should be used. So the lab has components, plus an integration.

1. We will first start with setting up **serial port.** The UART module is a device that can receive and transmit serial data with predefined "baud rates" (raw data bits per second). You can do a quick internet search for how UART works. **At its simplest UART is a parallel2serial shift-register (Tx) and a serial2parallel shift-register (Rx) module with some extra options and bits.** 8051 user manual also gives a fair explanation as to how it should be used. Even better is edsim51's relaxed-language explanation of how serial ports can be set up and can be used. You can also read up about how to use UART on c51 guides. Once we set-up the UART we will be able to use printf (and some others like _getkey)! Please read reference 1 and understand that the consequences of not matching the baud rate has severe consequences (you'll get garbled characters). If matching is awful this'll happen quick, if matching is close but not perfect you'll not be able to print long strings. **Note here that edsim51 is not perfect so it still garbles characters every once in a while.** Normally in a system the clock frequency is fixed (either via internal or external oscillators) and we usually set the baud rate as close as possible. Here, however since we're using a simulator we can set the clock frequency as we wish. So we will aim for 19200 baud rate, and you should set your system clock frequency for matching that baud rate perfectly. At the end of this exercise you should be able to tell how you calculated the necessary system clock frequency for 19200 baud rate, and you should demonstrate that you can send and receive characters. You should also test getc, getchar or scanf utils and ready to explain your experience.

For this lab it's utmost important that you use the edsim51_nonewline.jar variant, the standard one doesn't play nice with getc, getchar or scanf variants. The standard one appends a new-line character (0x10) every time you press send.

2. Now that our system clock is set, we can start thinking about the delay after reception. By having an empty function you can use a certain number of NOPs (no operation instructions) to create dumb delays. Taking it 1 step further, you can loop NOPs to achieve parametrised delay. However, their precision isn't amazing because we can't predict how many assembly instructions the compiler will give us, plus this delay will change if the clock frequency changes. There is always trial-and-error of course but that's not engineering. Instead we will implement busy-waiting delay loops by using timers *(timer0 specifically, because you should've used timer1 for serial port)*. Your goal here is to write a function with a single parameter *delay_time*, that waits just about *delay_time* milliseconds. To achieve this you will set up timer0 to count from/to a certain value and you'll constantly poll if timer0 overflow event has happened (read reference 2). The overhead from the function call, condition checking and other things are negligible, so don't worry about optimising it that much. By the end of this exercise you should demonstrate that you can create delays of arbitrary numbers. You should also be able to test its limits and be ready to explain minimum and maximum delays you could create. You should also be ready to think how you could implement delays longer than the maximum with timer0 (hint: for/while loops).

3. Another component is to read from the keypad. Some uControllers come come with internal keypad decoders but this 8051 does not. You should keep the Key bounce disabled for this exercise. Other settings are up to you *(I'd suggest Pulse to keep it simple)*. Study reference 3 and you should have no problem being able to read data from keypad. By the end of this exercise you should be able to demonstrate you can read values from keypad (printing is optional but you might as well since you got it ready in 2 right?).

4. Writing to LCD. The practice that I want you to gain here is completely different. Sometimes some components are extremely popular and common that I personally believe there is no need to re-invent the wheel (which is also one the basis of software engineering). What I want you to do is to find a library for the LCD, modify it if necessary to make it work and demonstrate that you can print arbitrary strings on the LCD. Please give reference to the library that you've found and used as a comment inside your code with links. And finally be ready to explain the shortcomings (if any) of the library that you found).

5. Integration: By the end of this lab you should implement a main function with an everlasting while loop (we talked about it in lectures) in which you should 1. receive a character and echo it back. 2. wait for a period, 3. print the char in first line, 4. make an keypad reading, 5. print keypad value in second line. You should also be able to reflect on your experience of elapsed time between 1-3 (inclusive). Think about for example how much period you wanted to set, and how much you actually got (i.e. what's the overhead of printf together and separately).

I know this is practically a massive step from 1st HW, but all the information you need is out there, and if not use the discussion forum here: HW 2 Discussion

Grading: each item is 20 points, where 5 points for each will be rewarded for in-demo questions.

Good luck,

References

1. https://www.edsim51.com/8051Notes/8051/serial.html (website is dead, find it in wayback machine)

2. https://www.edsim51.com/8051Notes/8051/timers.html (website is dead, find it in wayback machine)

3. https://www.edsim51.com/simInstructions.html#keypad

Submission status

Submission status	No attempt
Grading status	Not graded
Time remaining	3 days 15 hours
Last modified	-
Submission comments	▸ Comments (0)

Add submission

You have not made a submission yet.

◄ inttypes.h

Jump to...

HW 2 Discussion ►

AIRS
Webmail
Bilkent Home
Academic Calendar
Bilkent News
English (en)
Deutsch (de)
English (en)
Español - Internacional (es)
Français (fr)
Italiano (it)
Türkçe (tr)
Русский (ru)
العربية (ar)
한국어 (ko)
日本語 (ja)
简体中文 (zh_cn)

Data retention summary
Get the mobile app

Previous Semesters
General Purpose (gen3Moodle)
Moodle@Bilkent Tutorials
Zoom@Bilkent Tutorials
Guidelines by BETS
Ed. Tech. Support Unit (BETS)
Bilkent Computer Center (BCC)
STARS --- SRS --- AIRS
Webmail
Bilkent Home
Academic Calendar
Bilkent News