CS 431-1 Embedded Systems

Lab 2

In this lab we're going to play around with SFRs and some standard peripherals of 8051 along with an external component that is the ADC. The overall goal is to implement a device that can periodically read the ADC and dump the read data as a formatted string to serial port. 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, but timers should be used. So the lab has 3 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. 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. Once we set-up the UART we will be able to use printf! 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. 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 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 sends a new line every time you press send.

2. Now that our system clock is set, we can start thinking about the "periodicity" of this system. We've already seen that by having an empty function you use NOPs to create dumb delay loops. However, we've also mentioned that their precision isn't amazing because we never know what exactly the compiler will give us. 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 (or microseconds, that's up to you to figure out). To achieve this you will set up timer0 to count from/to a certain value and you'll constant 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.

3. Final software component is to read an external adc. Simple 8051 variants dont come with an adc, but edsim51 includes an external one. To our luck the workings of this external adc is also a good surrogate for internal ADC components. Study reference 3 and you should have no problem being able to read data from adc. By the end of this exercise you should be able to demonstrate you can read values from ADC (printf optional).

4. Integration: By the end of this lab you should implement a main function with an everlasting while loop (we'll talk about it this week) in which you should 1. wait for a predefined period, 2. make an ADC measurement, 3. print that ADC measurement. You should also be able to reflect on your experience of this loop. Think about for example how much period you wanted to set, and how much you actually got (i.e. what's the overhead of ADC and printf together and separately).

I know this is practically a massive step from 1st lab, but all the information you need is out there, and if not use the discussion forum here: https://moodle.bilkent.edu.tr/2019-2020-summer/mod/forum/view.php?id=3373

Grading: each item is 25 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

2. https://www.edsim51.com/8051Notes/8051/timers.html

3. https://www.edsim51.com/8051Notes/interfacing.html#adc

edsim51di.tar.gz 7 July 2020, 11:36 AM

Grading summary

Hidden from students	No
Participants	11
Submitted	6
Needs grading	6
Due date	Thursday, 16 July 2020, 2:30 AM
Time remaining	Assignment is due
Late submissions	Only allowed for participants who have been granted an extension

View all submissions | Grade

◄ Lecture 3 Recording

Jump to...

Lab 2 Discussion ►