Fall 2023

ECE 445L Lab 3

Alarm clock, LCD, edge-triggered input interrupts, and SysTick periodic interrupts

Table of Contents

Fall 2023

Team Size

The team size for this lab is **2**.

Goals

- Develop a graphics driver for the LCD that can plot lines and circles
- Design a hardware/software interface for a keyboard or individual switches
- Design a hardware/software driver for generating a simple tone on a speaker
- Measure supply current necessary to run the embedded system
- Implement a digital alarm clock using periodic interrupts

Review

- Valvano Section 3.4 on developing modular software
- Valvano Chapter 4 on Basic Handshake Mechanisms,
- Valvano Sections 4.5 and 4.9 on edge-triggered interrupts and the LCD
- Valvano Section 5.7 and 6.2 on periodic interrupts

Starter Files

- Starter project:
    - Lab 3 template provided on the Github Classroom repo.

Required Hardware

Parts	Datasheet	Price	Source (**price source)**
EK-TM4C123GXL	EK-TM4C123GXL datasheet	$16.99	**TI**
Sitronix ST7735R Color LCD	ST7735R Driver datasheet	$19.95	**Adafruit**
8Ω or 32Ω speaker	N/A	N/A	EER Checkout Desk
10kΩ resistor	N/A	N/A	EER Checkout Desk
1uF Tantalum Capacitor	N/A	N/A	EER Checkout Desk
Switches	N/A	N/A	EER Checkout Desk
IRLD120 MOSFET or IRLD024 MOSFET	IRLD120 datasheet IRLD024 datasheet	$1.88	EER Checkout Desk Or Mouser, **Digikey**
1N914 Diode	1N914 datasheet	$0.10	EER Checkout Desk, Or **Mouser**

Lab Overview

Labs in EE445L are extremely open-ended. For Labs 3, 4 and 5 you will be given a requirements document. Your TA is your client or customer. A grade of B can be achieved by satisfying these minimum specifications. To achieve higher grades, you are expected to expand sections 2.1 and 2.5 describing what your system will do. You are free to make any changes to this document as long as you achieve the educational goals for the lab. All changes must be approved by your TA. Excellent grades are reserved for systems with extra features and are easy to operate. You will need your LaunchPad and color LCD.

From checkout you can borrow speakers, switches, IRLD024 or IRLD120 MOSFET, 1N914, and some resistors for this lab.

Requirements Document

As always, feel free to adjust the syntax and format of your requirements document as you think appropriate. The goal of the document is to provide a clear and unambiguous description of what the project does. A copy or a template of this document is provided in the Github classroom repository.

1. Overview

1.1. Objectives: Why are we doing this project? What is the purpose?

The objectives of this project are to design, build and test an alarm clock. Educationally, students are learning how to design and test modular software and how to perform switch/keypad input in the background.

1.2. Process: How will the project be developed?

The project will be developed using the TM4C123 board. There will be switches or a keypad. The system will be built on a solderless breadboard and run on the usual USB power. The system may use the on-board switches and/or the on-board LEDs. Alternatively, the system may include external switches. The speaker will be external. There will be at least four hardware/software modules: switch input, time management, LCD numerical graphics, and sound output. The process will be to design and test each module independently from the other modules. After each module is tested, the system will be built and tested.

1.3. Roles and Responsibilities: Who will do what? Who are the clients?

EE445L students are the engineers, and the TA is the client. Students are expected to modify this document to clarify exactly what they plan to build. Students are allowed to divide responsibilities of the project however they wish, but, at the time of demonstration, both students are expected to understand all aspects of the design.

1.4. Interactions with Existing Systems: How will it fit in?

The system will use the TM4C123 board, a ST7735 color LCD, a solderless breadboard, and be powered using the USB cable. 1.5. Terminology: Define terms used in the document. Power budget, device driver, critical section, latency, time jitter, and modular programming. See textbook for definitions

1.5. Terminology: Define terms used in the document.

Power budget, device driver, critical section, latency, time jitter, and modular programming. See textbook for definitions.

1.6. Security: How will intellectual property be managed?

The system may include software from Tivaware and from the book. No software written for this project may be transmitted, viewed, or communicated with any other EE445L student past, present, or future

(other than the lab partner of course). It is the responsibility of the team to keep its EE445L lab solutions secure.

2. Function Description

2.1. Functionality: What will the system do precisely?

The clock must be able to perform five functions. 1) It will display hours, minutes, and seconds in just numeric forms on the LCD. The numerical output will be easy to read. 2) It will allow the operator to set the current time using switches. 3) It will allow the operator to set the alarm time including enabling/disabling alarms. 4) It will make a sound at the alarm time. 5) It will allow the operator to stop the sound. An LED heartbeat will show when the system is running.

2.2. Scope: List the phases and what will be delivered in each phase.

Phase 1 is the preparation; phase 2 is the demonstration; and phase 3 is the lab report. Details can be found in the lab manual.

2.3. Prototypes: How will intermediate progress be demonstrated?

A prototype system running on the TM4C123 board and solderless breadboard will be demonstrated. Progress will be judged by the preparation, demonstration and lab report.

2.4. Performance: Define the measures and describe how they will be determined.

The system will be judged by three qualitative measures. First, the software modules must be easy to understand and well-organized. Second, the clock display should be beautiful and effective in telling time. Third, the operation of setting the time and alarm should be simple and intuitive. The system should not have critical sections. All shared global variables must be identified with documentation that a critical section does not exist. Backward jumps in the ISR should be avoided if possible. The interrupt service routine used to maintain time must be completed in as short a time as possible. This means all LCD I/O occurs in the main program. The average current on the +3.3V power will be measured with and without the alarm sounding

2.5. Usability: Describe the interfaces. Be quantitative if possible.

Minimum requirements: There will be two switch inputs from PF0, PF4 (you can add additional external switches as well). In the main menu, the switches can be used to activate 1) set time; 2) set alarm; 3) turn on/off alarm; and 4) display mode. The user should be able to set the time (hours, minutes, seconds) and be able to set the alarm (hour, minute). Exactly how the user interface works is up to you. After some amount of inactivity, the system reverts to the main menu. The user should be able to control some aspects of the display configuring the look and feel of the device. The switches MUST be debounced, so only one action occurs when the operator touches a switch once.

Minimum requirements: The LCD display shows the time using 12 numbers, the minute hand, and the hour hand are easy to see. The clock must display the time in numeric mode using numbers.

Minimum requirements: The alarm sound can be a simple square wave. The sound amplitude will be just loud enough for the TA to hear within 3 feet. 90% will be the maximum score for meeting all minimum requirements.

2.6. Safety: Explain any safety requirements and how they will be measured.

The alarm sound will be VERY quiet in order to respect other people in the room during testing. Connecting or disconnecting wires on the protoboard while power is applied may damage the board.

3. Deliverables

3.1. Reports: How will the system be described?

The lab report described below is due by the due date listed in the syllabus. This report includes the final requirements document.

3.2. Audits: How will the clients evaluate progress?

The preparation is due at the beginning of the lab period on the date listed in the syllabus.

3.3. Outcomes: What are the deliverables? How do we know when it is done?

There are three deliverables: preparation, demonstration, and report. (Note to students: you should remove all notes to students in your final requirements document).

Preparation

1. Edit the requirements document to reflect your design. The requirements document is fluid, and we expect it to change as you develop your solution and discover what works and what doesn't. You are allowed to modify the requirements document.
2. Draw a detailed circuit diagram showing all external hardware connections. We expect you to use KiCad (because this is the program with which we will be designing PCBs in Labs 6 and 7). More specifically, we want you to use the same CAD program you plan to use in Labs 6 and 7. Label all hardware chips, pin numbers, and resistor values. You do have to show connections to the LaunchPad, but not circuits within the LaunchPad itself. You must have in your possession all external hardware parts, but you do not have to construct the circuit. To limit the surge current into the MOSFET, we recommend a 10k resistor
3. For each module you must have a separate header and code file. As stated earlier we expect at least four modules. As part of the preparation, you need to have the software designed, written and compiled. For the preparation, you do not need to have run or debugged any code. For the modules you have written include a main program that can be used to test it. The SysTick or timer module used to maintain time must be written at a low level, like the book, without calling TivaWare driver code.
4. Write one main program that implements the digital alarm clock. Figure 3.1 shows the data flow graph

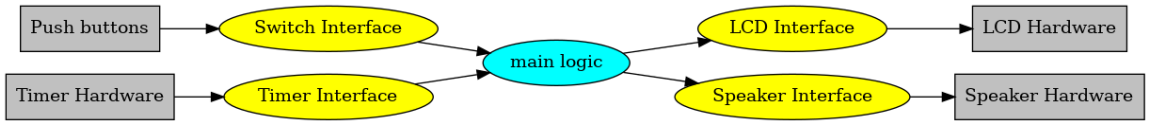


Figure 3.1. Data flows from the timer and the switches to the LCD and speaker.

Figure 3.2 shows a possible call graph of the system. Dividing the system into modules allows for

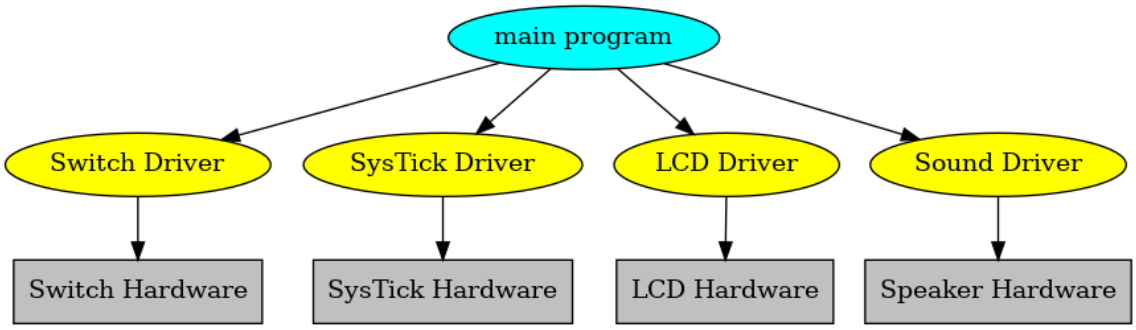


Figure 3.2. A call graph of the four modules used by the alarm clock.

5. Choose a frequency for the alarm sound, f. Let R (e.g., 32Ω) be the resistance of your speaker. Let C (e.g., 1µF) be the capacitance of your ceramic capacitor. Calculate, the time constant of an RC circuit is $\tau = R * C$, in seconds. Calculate the corresponding cutoff frequency is $f_c = 1/(2\pi RC)$The system should have a cutoff frequency $2 \leq f_c \ / \ f \leq 10$, so it passes the alarm sound, but rejects some of the harmonics of the square wave.

Procedure

1. Build and test any external hardware needed. Debug each module separately. Debug the overall alarm clock. Measure how long it takes to update the graphical time on the LCD. Identify all shared I/O ports and global variables. I.e., document in your software all the permanently allocated variables that have read or write access by more than one thread. Next, consider what would happen if the interrupt occurred between any two instructions of the main program. Remember high priority interrupts can suspend lower priority ISRs. Look for critical sections, and if you find any remove them. Document in your software that each shared object is not critical. During checkout, the TA may ask you to prove that your system has no critical sections.)
2. Record the +3.3V power line rms noise level. The easiest way to measure rms is to use the voltmeter in AC voltage mode. Theoretically one would expect this rms level to be zero, but the presence of noise will cause the rms value to be somewhere in the range of 0.5 to 5 mV.
3. We expect you to use a real oscilloscope. Use it to measure the speaker voltage when the alarm is sounding with and without the capacitor. I.e., measure the drain pin of the MOSFET (bottom side of the speaker). Figure 3.3a was measured with no resistor between digital output and the gate of the MOSFET, and with no diode or capacitor. Look at the **back EMF**, which is what happens without the 10k resistor, diode or capacitor. The **dI/dt** when the current is removed is so large the inductance of the speaker produces a 25V spike. Recall the voltage across an inductor is **V=L*dI/dt**. When we interface a device with inductance, we MUST remove the large voltage spikes caused by back EMF. ***Do not try to collect data without the 10k resistor, it will destroy your circuits.***

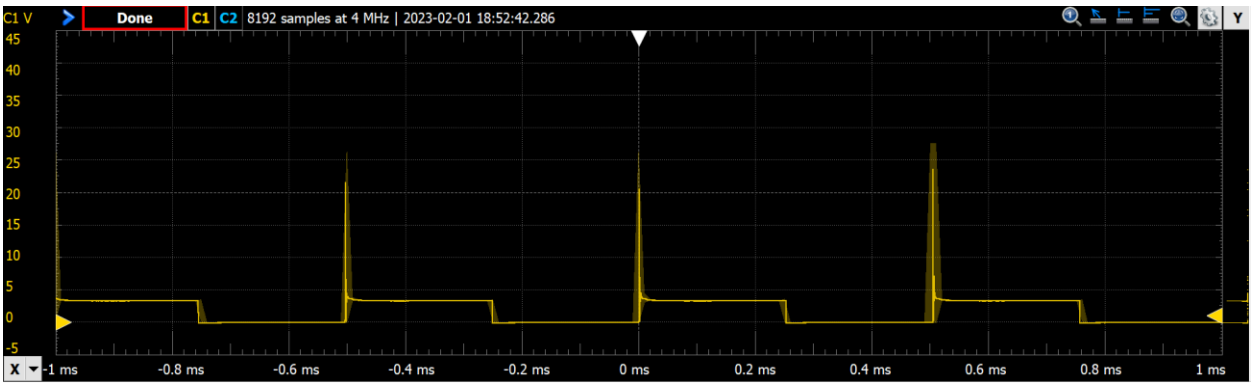


Figure 3.3a. Scope trace of the voltage on the speaker (the other speaker pin is +3.3V). Current flows when the drain pin is low. 32-ohm speaker 0-Ω resistor between pin and gate, and IRLD120 MOSFET.

Figure 3.3b was measured with a 10k resistor between digital output and the gate of the MOSFET, and with no diode or capacitor. The 10k resistor slows the MOSFET down, reducing the dI/dt. Notice the small voltage spikes (4.6V peaks) that occur when current is turned off. Use the scope to verify the sound frequency is 2 kHz

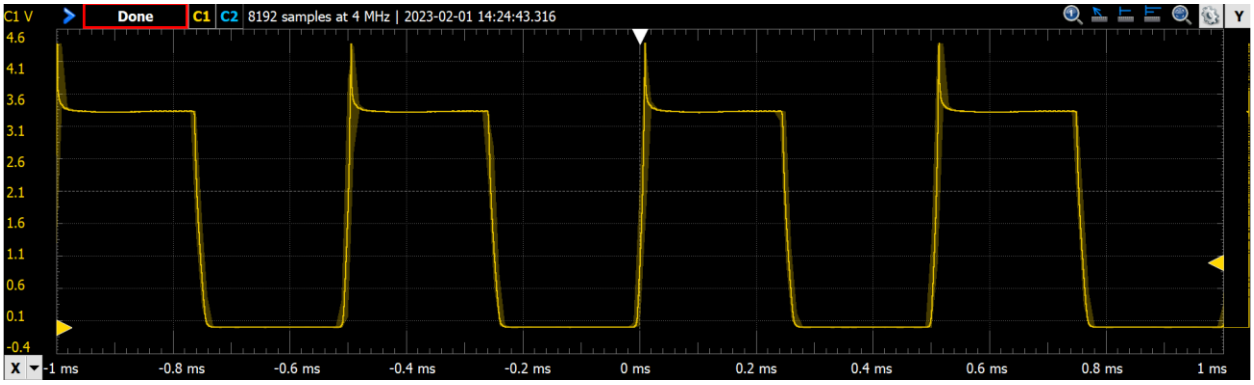


Figure 3.3b. Scope trace of the voltage on the speaker (the other speaker pin is +3.3V). Current flows when the

Decide whether you wish to use either a diode or a capacitor to remove the back-EMF spikes. Figure 3.3c was measured with a 1uF capacitor. Notice the small voltage spikes are gone and the edges are round.

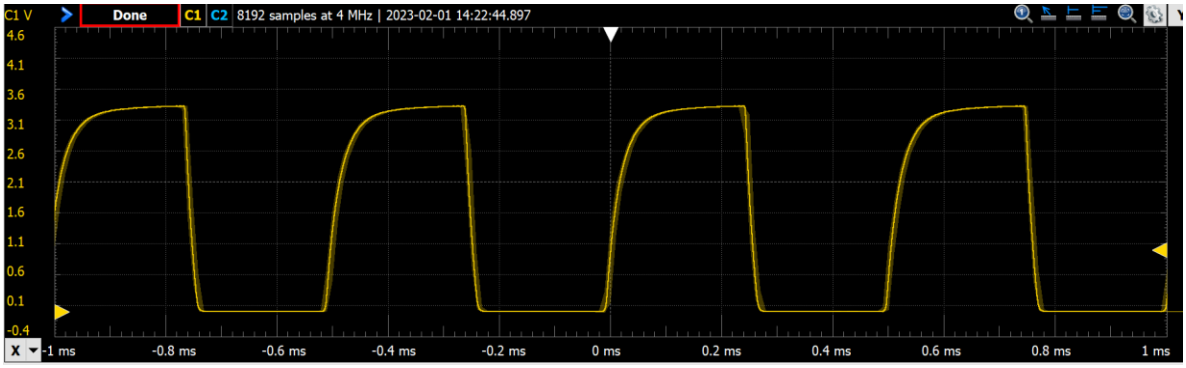
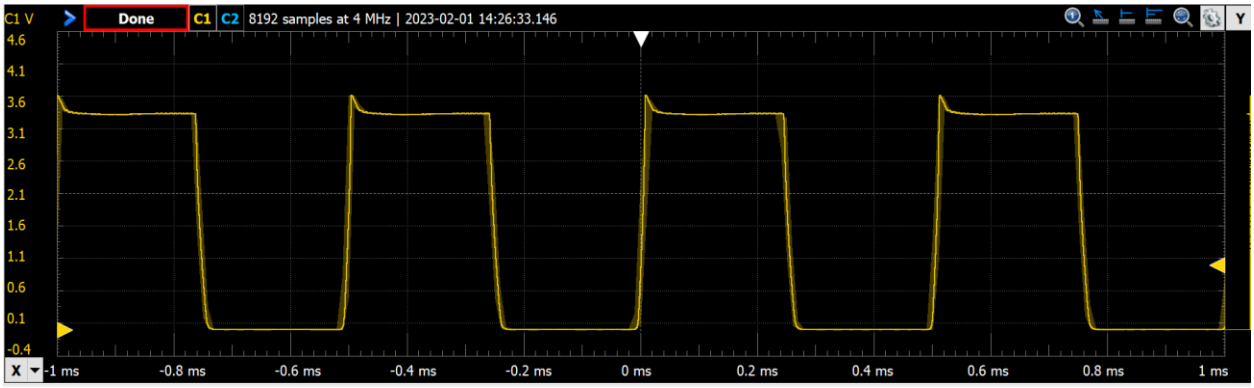


Figure 3.3c. Scope trace of the voltage on the speaker (the other speaker pin is +3.3V). Current flows when the drain pin is low. 32-ohm speaker 10k-Ω resistor between pin and gate, and a capacitor across the speaker.

Figure 3.3d was measured with a 1N914 snubber diode. Notice the voltage spikes are smaller but not totally gone.



4. Measure the 3.3V supply current. Remove the jumper on the LaunchPad (see Figure 3.4) and connect a DC current meter across the pins. Double check the connections before turning it on. If you are at all unsure about this measurement, ask your TA for help. Measure the required current to run the alarm clock. Take a measurement with and without the alarm sounding. *Remember these numbers; they will be valuable when you design your project in Labs 6 and 7.*

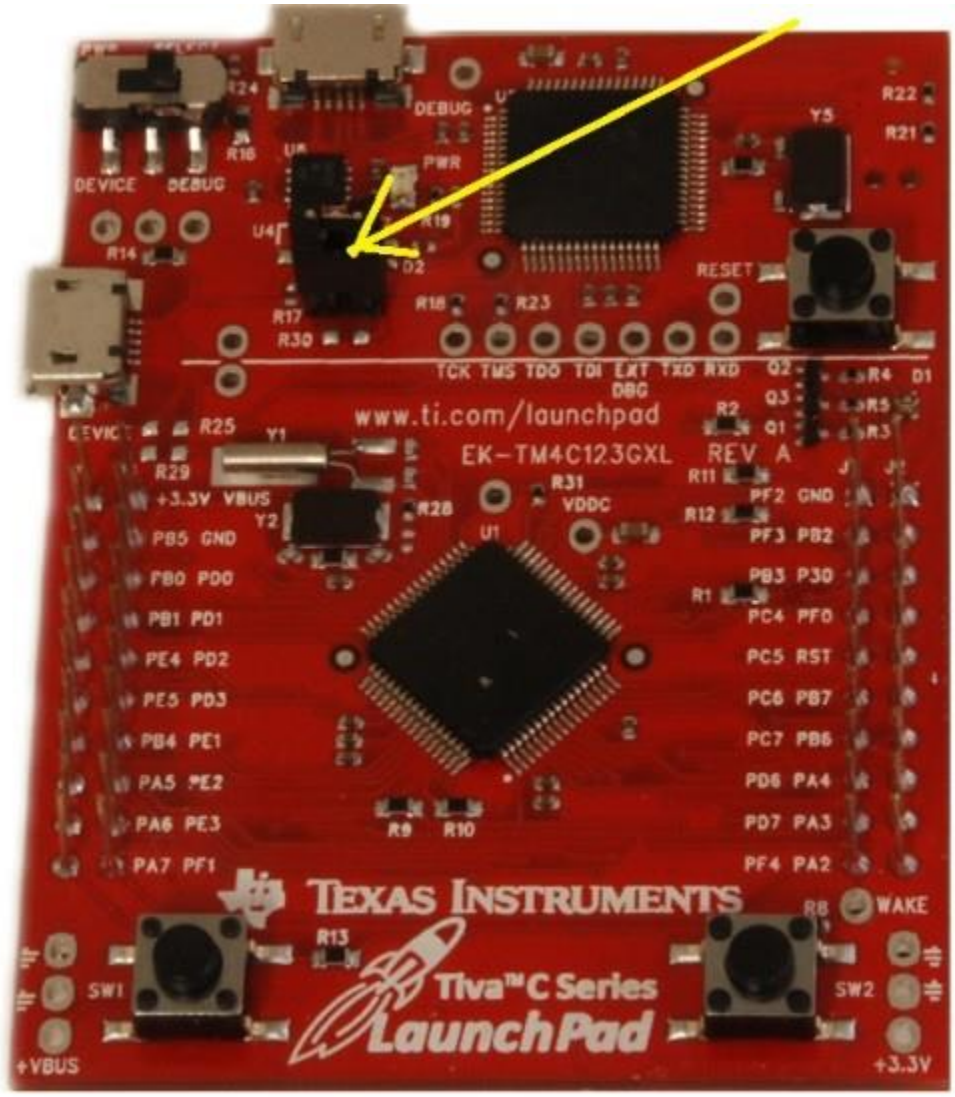


Figure 3.4. Use a current meter to measure required current to run the clock.

Deliverable 1

Draw the electrical circuit you used to create the alarm clock. Put either a diode or a capacitor in parallel with the speaker. If you use a polarized tantalum capacitor, orient the + and - pins with the direction of the current flow. The diode on the other hand must be oriented with the + and - pins opposite with the direction of the current flow.

Deliverable 2

Measure how long it takes the LCD graphics to update.

Deliverable 3

Show the RMS noise level on the 3.3V with and without the alarm sounding.

Deliverable 4

Measure the voltage versus time of the drain pin of the MOSFET, without capacitor, and use it to determine the current through the speaker. Measure the frequency of the sound. Place a picture of the scope trace like **Figure 3.3b** into your lab report, either a photo or digital downloaded image

Deliverable 5

Measure the voltage versus time of the drain pin of the MOSFET, with the capacitor or the diode. Place a picture of the scope trace like **Figure 3.3c** or **Figure 3.3d** into your lab report, either a photo or digital downloaded image

Deliverable 6

Show the system current with and without the alarm sounding.

Lab Checkout

Demonstrate that your digital alarm clock is stand-alone by turning the power off, then on. The digital alarm clock should run (the time will naturally have to be reprogrammed) without downloading the software each time.

Lab Report

Deliverables

1. Objectives (final requirements document)
2. Hardware Design (Deliverable 1)
    a. Create a schematic or figure showing all external components connected to the TM4C123 board. You do not need to show hardware components on the TM4C123 LaunchPad board
3. Software Design
    a. If you organized the system different than Figure 3.1 and 3.2, then draw its data flow and call graphs
4. Measurement Data
    a. Deliverable 2: LCD graphics update latency
    b. Deliverable 3: 3V3 RMS noise
    c. Deliverable 4: Speaker measurements without dampening
    d. Deliverable 5: Speaker measurements with dampening
    e. Deliverable 6: System current measurement

Analysis and Discussion Questions

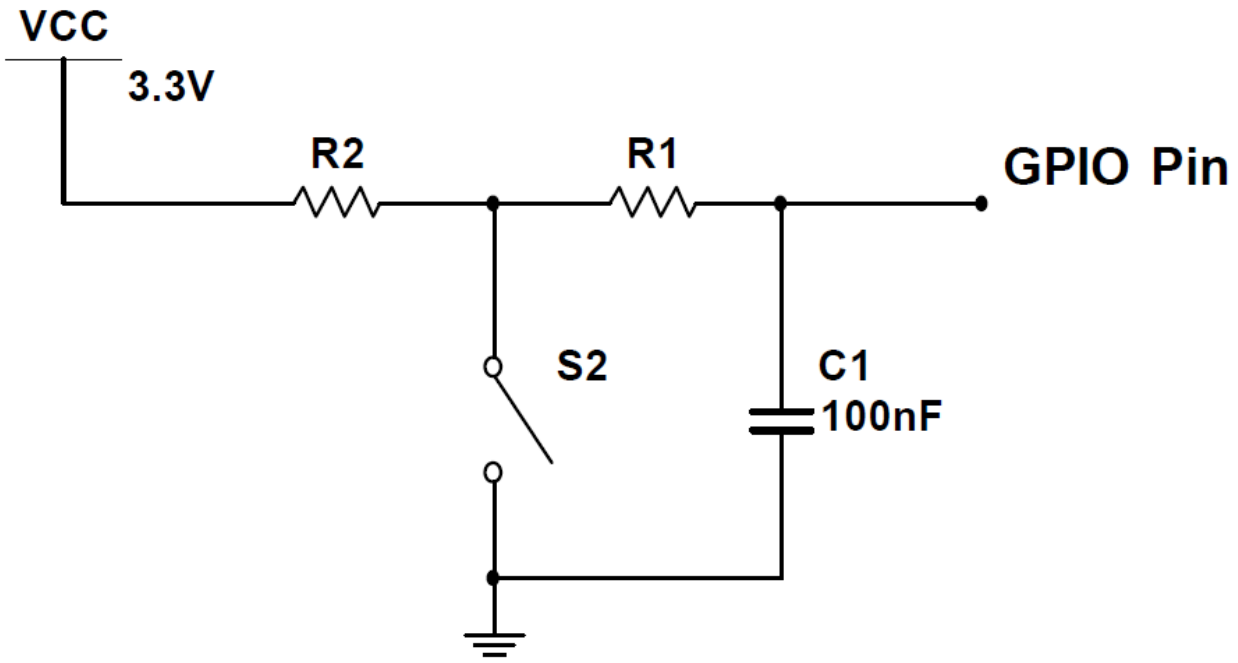
Give short one or two sentence answers to these questions.

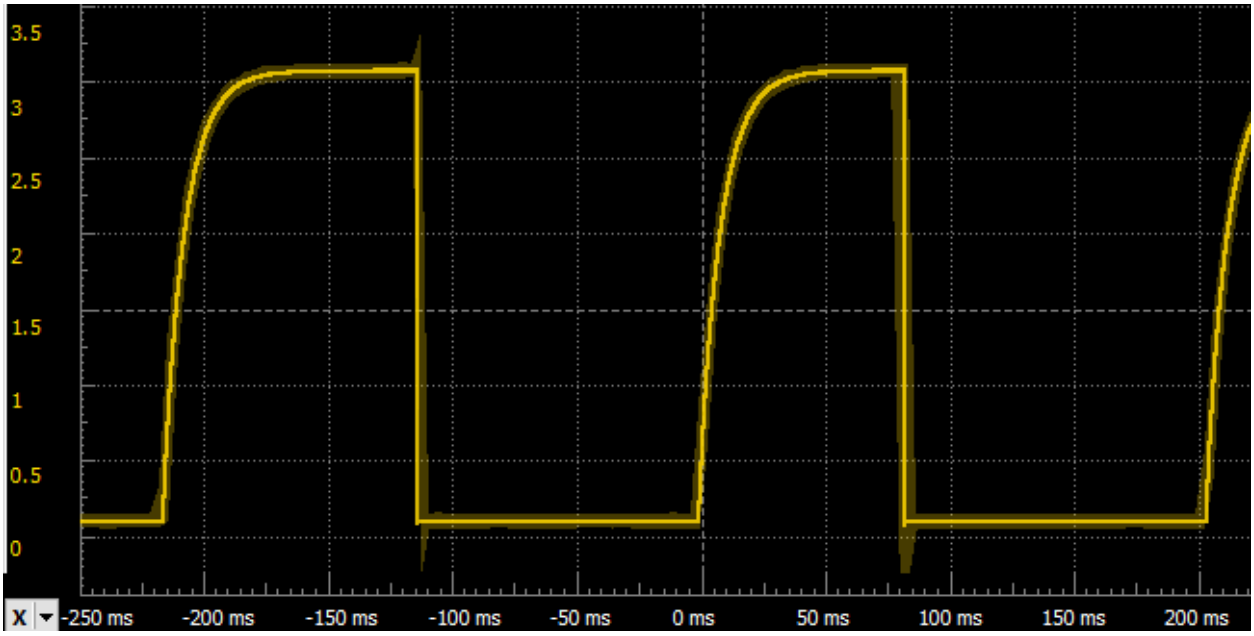
1. Give two ways to remove a critical section.
2. What would be the disadvantage of updating the LCD in the background ISR?
3. Did you redraw the entire clock for each output? If so, how could you have redesigned the LCD update to run much faster, and create a lot less flicker? If not, how did you decide which parts to redraw?

4. Assuming the system was battery powered, list three ways you could have saved power.

Hints

1. The requirements document should change a couple of times during the lab as you determine features
2. You can interface an 8Ω or 32Ω - speaker to an output port using an NPN MOSFET like the IRLD120. A 10kΩ resistor between digital output pin and gate reduces current surges but does not affect loudness. Loudness is determined by the voltage drop across the speaker. Connect the source to ground, and the drain to one side of the speaker. Connect the other side to +3.3V. The maximum I_{DS} of the transistor must be larger than 3.3V/8Ω or (3.3V/32Ω). The speaker has inductance, but the MOSFET includes an internal diode to remove back EMF when the transistor switches off. If you toggle the output pin in the background ISR, then sound will be generated. Loudness is determined by the voltage drop across the speaker. From Figure 3.3a, we see the MOSFET drain voltage is about 0.5V when active. So, the voltage drop will be 3.3V-0.5V = 2.7V
3. You must be careful not to let the LCD show an intermediate time of 1:00 as the time rolls over from 1:59 to 2:00. You must also be careful not to disable interrupts too long (more than one interrupt period), because a time error will result if any interrupts are skipped.
4. You may use 32-bit or 64-bit timer modes on the TM4C microcontrollers. However, it is good practice to refer to the errata for the microcontroller you are using. The errata describe bugs and flaws not listed in the data sheet.
5. If you use the on-board switches, then you must activate the internal pull-up resistors. You will set the corresponding bits in the GPIO_PORTF_PUR_R register. These on-board switches are simply SPST switches to ground. When the switch is pressed, the signal goes to 0V (ground). When the switch is not pressed, the internal pull-up makes the signal go high (3.3V.) Furthermore, coming up out of a reset PF0 is locked, and thus if you use PF0 you will need to unlock it.
6. Learn to use KiCad. You will need to be proficient with this application during Labs 6 and 7. Using it now for simple circuits will be an efficient use of your time.
7. If you use edge-triggered interrupts, build an analog filter to debounce each switch. Set R1=0, and R2=100k to create the negative logic switch. Choose R2 and C1 so the time constant (τ=R2*C1) is around 10 ms. Test the circuit with a scope before connecting to the microcontroller.





8. A better method to debounce edge-triggered interrupts is to use a second timer (see **EdgeInterruptDebounce_4C123**)