SWITCH: Microcontroller Touch-switch Design & Test (Part 2)

2nd Year Electronics Lab

IMPERIAL COLLEGE LONDON

v2.10

Table of Contents

Equipment

- Custom designed PCB
- Soldering Equipment
- Bench PSU, Signal Generator, Oscilloscope and Digital Multi Meter
- 14pin DIL socket, 40pin DIL socket, 8 pin SIL OLED connector, 74HC14 IC, laboratory resistors and capacitors
- **mbed** module and OLED display module

Aims

In this experiment, you will test and debug the switch hardware and software for your touch-switch PBC. Finally, you will write a sample high-level application using the touch-switches as user input to control a software signal generator.

Objectives

1. Build and debug hardware
2. Integrate and test hardware with software
3. Make switch on/off indication robust
4. Design and test high-level User Interface

Recommended Timetable

- Session 1 – Build and debug hardware, test away from microcontroller. It is challenging to understand everything about the operation of your touch-switches, so make sure you start building and debugging hardware as soon as possible. If you spend too long on this part, split up to ensure that you are in parallel working on the required software for sessions 2, 3, and 4.
- Session 2 – Integrate hardware with microcontroller. Test switch frequency characteristics and design a robust switch on/off indication. Your switches should provide a single "switch press" indication when touched.
- Session 3 – Code high-level Graphical User Interface (using your touch-switches and the OLED display) to generate signals of various frequencies.
- Interviews will be held during your last session.

Introduction

This experiment continues from PCB last term; refer to the PCB information as necessary. The experiment contains an interesting mixture of software and hardware work, typical of many embedded applications. Although the basic principles of operation are not complex, the way in which they interact to make errors or limitations in the design is very complex. In order to make your touch-switches work well, which can be challenging, you need to understand both the hardware error mechanisms and the software techniques that can be used to reduce errors, such as false triggering. The CPU you use is very fast, but the **mbed** software framework is particularly slow. This means that it will difficult to obtain very high frequencies for your switches which is otherwise desirable.

This handout is not long. Please read all of it before starting and refer forward as necessary – for example to the PCB section, if you are testing component values using the PCB.

Note on Q* questions. These are questions about the design that will deepen your understanding but are not directly needed. You may discuss them with demonstrators if you wish. You should not spend too much time on these, since time is limited.

Session 1 – PCB Build Up

The PCB you have designed in the ARM experiment last term should now be fabricated. In this first section, you will be calculating the appropriate component values as well as soldering all the components onto the PCB.

Upon collecting your PCB, you will also need to collect the 14pin DIL socket, the 40pin DIL socket, 8 pin OLED module connector, as well as the 74HC14 IC.

Initial PCB Checks

Debugging new hardware efficiently is not complex, but requires careful attention to detail and tasks must be performed in the correct order. The same strategies will work with a wide range of circuits, so understanding how best to debug hardware is a useful skill.

While building and testing, remember to switch off the power supply whenever you are soldering the board. Soldering irons are connected the mains ground and can easily short out live circuits since oscilloscope earth is also connected to mains ground.

1. Before loading any components check the PCB with a multimeter measuring resistance:
    a. Are the two supplies (measured at the PCB header) open circuit? If not there is probably a schematic error leading to a connection between the supplies. Debug this by inspection and correct, since nothing else can be done if supplies are connected together.
    b. Are the 74HC14 IC supplies correctly connected to the mbed socket 3.3v out and GND pins?
2. For initial testing, temporarily solder flying leads to GND and 3.3V nets (you may have pins for this). Use these to connect to a bench PSU. Double check, directly from the IC and mbed socket supply holes, that the supply voltage is correct.
3. Solder the DIL sockets for the 74HC14 and the **mbed**, and the OLED connector, onto the top side of the PCB. Be careful to get sockets the right way round. The advantage of using a DIL socket is that broken ICs can be diagnosed and replaced. (In the case of the mbed and OLED modules it is essential since we will reuse these!). NB - if you have used an SMD 74HC14 you will have to solder this in place - check with lab technicians for help.

Q* Which PCB problems (short circuit or open circuit) are most difficult to locate and therefore debug?

Q* Can you measure the resistance of a component accurately in circuit?

Component Selection and PCB Build

The touch-switch forms a relaxation oscillator with frequency (see 74HC14 datasheet):

$$f = \frac{1}{KRC'}$$

Where K is a constant equal to 1.2 when running at 3.3V. Here, C' is a parallel combination of the touch-pad capacitance C_p, track capacitance (which can be ignored) and a chosen capacitor C.

$$C' \approx C_p + C$$

C_p has two possible values. With the touch-pad not touched, it will have a low value of C_{p0} (typically less than 25pF). With the switch touched, it will have a high value of C_{p1}, which depends to some extent on how hard the touch-pad is pressed. Note that pressing the switch a finger must overlap both parts of the touch-pad. C_{p1} can be measured using an RLC meter.

A wide range of possible R and C values can, in principle, be used, and exact values are not very critical. The appropriate component values can be chosen by noting the following constraints:

- f < 75kHz - this ensures that the frequency is low enough to be measured by the sample software. For this constraint assume pessimistically that $C_{p0} = 0$.
- $\frac{C_{p1}}{C} > 0.2$, this is desirable to make the frequency change easy to measure.
- R is as small as possible subject to the above constraints.

Mains interference picked up by a person touching the pad causes an unwanted 50Hz current I_n to be injected onto the capacitor. This will disturb the oscillator by a ratio θ where:

$$\theta = \frac{I_n}{I_R}$$
$$I_R \approx \frac{1.5V}{R}$$

From which we see that smaller R will lead to a smaller effect of interference. This justifies the third constraint above.

- Calculate R & C values for the switches
- Solder R & C components (**one oscillator only**). Leave 5mm bare wire on the resistor so that if in later sessions you need to adjust the R value this can be done without unsoldering the resistor, which is dangerous and will usually break the PCB.
- Test oscillator frequencies with the switch pressed and not pressed. (mbed not on board, bench power supply at 3.3v).
- If all is good solder the three other oscillator components. Allow 5mm free wire between resistors and board to allow later change in resistor values by cutting out and soldering to leads, without desoldering from the board.
- Check that the oscillator outputs are present on the **mbed** pins expected from your layout, and that frequencies are roughly as you expect.
- Reflect on your build and test experience. Even if no problems were discovered, you can consider the methodology you would have used to identify possible problems.

NOTE: the Schmitt trigger circuit is only approximately characterised by the datasheet frequency of $1/(1.2RC')$. There are deviations at high frequencies (f > 4MHz) and both large (R > 1MΩ; $C \approx 0$) and small (R < 500Ω) values of R.

By the end of session 1 you should have a working PCB with 4 oscillators whose frequencies, measured by a scope etc, changes expected when you press switches.

Q* Can you characterise these deviations and explain them?

Q* You were told to minimise parasitic capacitance between oscillator inverter inputs and different oscillator outputs. Do you note any problems due to this (you may not). What problems would exist?Experiment with a small capacitor simulating parasitic capacitance to determine its efefct on the oscillator frequencies as switches are pressed. Can you understand this – either theoretically – quantifying the effect – or experimentally.

Session 2 – Software Integration and Development

For this section you do not need the bench PSU, since the hardware will be powered by the 3.3v regulated VOUT from the MBED module which itself is powered from a host PC via its USB connection.

DO NOT CONNECT A BENCH PSU TO YOUR BOARD WHEN YOU HAVE AN MBED MODULE IN PLACE. THIS MAY BREAK THE MBED.

Make sure any flying leads are tidy and unable to short the circuit. Disconnect all test equipment (you do not need it, having previously tested the hardware). Shorts at this stage are capable of breaking the **mbed** module. Remember that the scope ground and **mbed** module ground are likely to be connected via mains earth (because the **mbed** is connected to a grounded PC via its USB cable).

Software preliminaries:

- Go to http://developer.mbed.org
- Sign up for an mbed acount if needed
- Compiler->Add Platform -> LPC1768
- Compiler->Import->Programs Tab->Import from URL-> https://developer.mbed.org/users/estott/code/IC-SWITCH/
- This will set up a copy of the sample project software in your own **mbed** account cloud workspace. You can compile this and download it to an **mbed** module attached via USB using the web interface.

List of actions for this task:

1. Connect MBED module and OLED display to your PCB. Connect MBED module to PC via USB lead.
2. Check that the SPI port used in the sample code for the OLED display uses the same pins as those in your hardware: if not change the sample code to use the correct port.
3. Compile and download the mbed sample project code to your MBED module (see link on handout page). Check that your OLED works.
4. Modify the mbed pin as needed to use one of your oscillators and check that you can correctly print its frequency on the OLED. Use a measurement period a multiple of 20ms to minimise the effect of mains interference.
5. NOTE: Use the OLED to help you debug, print the output frequency onto the screen.
6. Repeat for all oscillators
7. Develop and test software to make one of your touch-switches give a robust and fast on/off indication. This can be quite challenging. Note the questions below.
8. Modify your software to work for all switches simultaneously.

Q Why would having different on and off switching frequency thresholds make the touch switch more robust?

Q Why is a multiple of 20ms measuring period preferable to minimise noise?

Q Why will a longer measurement time give a more accurate switch touch indication?

Q* Can you exactly characterise the unwanted variation in measured oscillator frequency when a switch is being touched? Think of this variation as being because the instantaneous oscillator frequency is modulated by a potential ideal 50Hz error signal as above.

Q* If the input frequency is higher than the maximum for reliable detection what would you expect the measured frequency to be relative to real? How would it change when the touch-pads are touched?

Q* Suppose the C values are only specified to within +/-30%. How could you write code to make the switches work with no manual setup of thresholds?

Sessions 3 – Interface Design

Design a user interface which uses touch-switches to input a **4 or more digit decimal code**. Use this to control the frequency (in Hz) of a square wave on an output pin of the mbed module, making a simple square wave signal generator (see mark scheme). The mbed output pins can be programmed using `DigitalOut` or `PwmOut`. You may find **mbed** `Ticker` useful.

If you have time make a more sophisticated signal generator with square/triangle/sine outputs using a digital to analog convertor.

Q* What are the limits to square wave generation using Ticker and PWM? Why might Ticker in some circumstances disturb the frequency measuring? Why might Ticker cause jitter?

Q** Assuming that C++ is no more than two times slower than assembler, can you estimate what is the maximum square wave frequency that could be generated using interrupts? Assume the LPC1768 clock frequency is 96Mhz and that this CPU executes ARM7 code - as used in the Introduction to Computer Architecture. Determine interrupt entry time from the LPC1768 datasheet. (In fact the LPC1768 runs the sophisticated Thumb-2 instruction set but this is comparable in speed to, and compatible with, the code you have studied.)

Q** For interest only. How does the Thumb-2 ISA differ from the ARM ISA?

Further Information

Using edge GPIO interrupts under the mbed framework as in the sample code is easy but, it turns out, **very slow** because of the high interrupt overhead within the mbed framework.

https://developer.mbed.org/forum/mbed/topic/2326/

There are a number of more complex possible ways to speed up this type of embedded system pulse counting. These all require some work understanding the hardware and low-level software of the ARM microcontroller.

1. Using raw interrupts: replacing the mbed interrupt vector for the NVIC vector used by the GPIO edge interrupt by your own purpose written code, possibly in optimised assembler.

2. Using a **polling loop**, as in Table 1, where all 4 input pins are loaded in a single hand-crafted load of the memory-mapped GPIO pin register or (slower) a `PortIn` mbed operation. The pseudocode can with effort be made fast if good use is made of ARM bitwise logical instructions: these can be compiled from C bitwise logical operators.

   Each iteration the current and previous values of the inputs are compared. For example '1' current and '0' previous means a '0' -> '1' transition happened. All such detected transitions can be counted.

   The polling loop can be stopped after a specific amount of time as determined by an **mbed** `TimeOut` object which can change the value of the global variable `running` to determine whether the loop continues. The `volatile` qualifier is essential here to tell the compiler this variable may be changed by the TimeOut attached function which executes as an interrupt.

3. Use a microcontroller hardware timer as a counter with an input pin as a clock source. This method is obviously fastest, since the counting is entirely in hardware, but can have limitations based on the microcontroller hardware available.

```
volatile int running = 0; // volatile is needed here to make access
                          //from normal and interrupt code safe
void polling_loop()
{
    running = 1;
    while (running)
  {
    // pins_previous = pins_current
    // pins_current = GPIO input pin values
    // compare current with previous and incremented counter on edges
  }
}
```

Table 1 - Pseudocode for polling loop

Appendix 1: Marking Scheme

Logbooks are not marked, but can be used by students at interviews. Yellow marks require higher levels of analysis, design, or understanding.

Mark	Component	Notes
10%	Own PCB used	If we agree (in advance) it is not your fault you could not use your own PCB you will get this mark, otherwise only those using own PCBs will have it.
20%	4 touch switches work	This will be reduced if only some touch-switches work
10%	Touch switches work robustly	Multiple triggering is not possible; touch-switches cannot affect other switches; markers can easily get switches to work
5%	Touch switches work quickly	Response to touch on/off appears instantaneous (or nearly so)
10%	Application works as specified	You must be able to output (at least) a square wave frequency in Hz corresponding to any non-zero 4 digit decimal number. The frequency must be accurate to within 1% measured over a 1s period. Jitter on individual edges is allowed.
5%	Touch switches are not disturbed by output.	At least one touch switch must work while frequency is output. This is sometimes tough to implement, but desirable so that new numbers can be entered without resetting the board
20%	Deep understanding of experiment	This will be individually determined during interview/demo
20%	Outstanding work	This mark, or part of it, will only be awarded for work which is unusually excellent and innovative. Anything adopted by multiple pairs, or pairs last year, will not count. Something different but not outstanding will not count.