# EECS 388 Laboratory Exercise #03
# Button Input and Serial Communication

September 16, 2018
Gary J. Minden

## 1 Introduction

In this laboratory exercise you will write, compile, execute, and demonstrate a task to sense the push buttons on the TM4C1294 board, generate a tone, and communicate over the serial port.

This laboratory builds on Lab #01 (Blinky) and Lab #02 (SpeakerBuzz). You should make a copy of your Task_Blinky/SpeakerBuzz project and work with the copy in this lab.

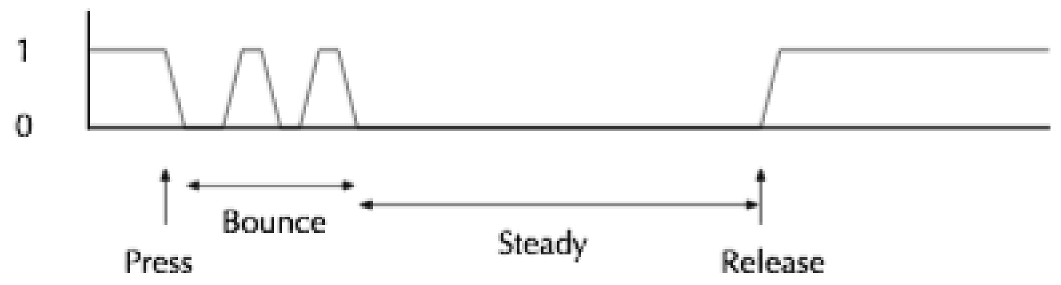
The program will: (1) monitor the two push buttons on the TM4C1294, (2) generate a (distinct) tone when a button is pushed and released, and (3) report the button press via the serial port. You will write a report describing your measurements, analyses, and conclusions based on your GTA's directions.

### 1.1 TM4C1294 Evaluation Board Switches Background

The TM4C1294 evaluation board has two switches that can be used for user input. These are named USR SW1 and USR SW2 and are located opposite the Ethernet connector. When a switch is pressed, the corresponding GPIO input signal is pulled to Ground, i.e. a 0 input to your program.

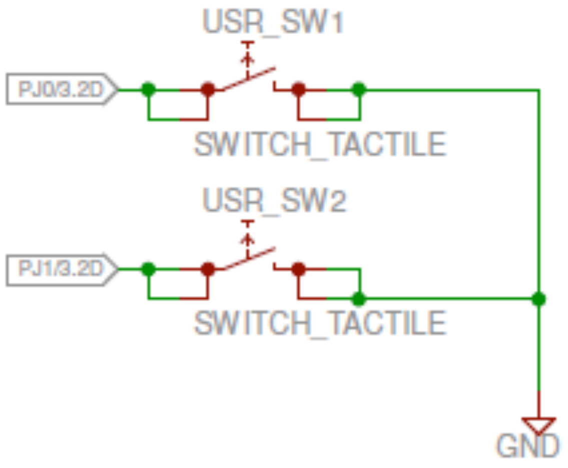
#### 1.1.1 Switch Bounce

When a switch is pressed, it does not generate a clean signal. That is, the signal may change from 1 -> 0 -> 1 -> 0 -> 1 -> 0 before it settles at a 0 state. This is called switch bounce and is a characteristic of the switch mechanics. When the switch is released, it returns to a 1 state. Generally, there is no switch bounce when the switch is released. A typical GPIO signal might look like:



Your program must compensate for switch bounce.

The two switches on TM4C1294 evaluation board are shown in the following figure:



The figure is from: http://www.ittc.ku.edu/~gminden/Embedded_Systems/PDFs/TI_TM4C1294_LaunchPad_Eval_B71003.pdf. The schematic shows the signals, e.g. USR_SWx, are connected to ground when the switch is pressed. The GPIO port must provide a weak pull-up to bring the signal high when the switch is not pressed. The schematic shows that signals USR_SW1 is connected to PortJ<0> and USR_SW2 is connected to PortJ<1>. Therefore, you will have to program GPIO PortJ appropriately to read the buttons.

Review class notes on GPIO from lecture and the TI DriverLib documentation at: http://www.ittc.ku.edu/~gminden/Embedded_Systems/PDFs/TI_TIVA_DriverLib_UG-2.1.0.12573.pdf. [Remember, some browsers may break long URLs when reading PDF documents in the browser.]

## 2 Laboratory Exercises

### 2.1 Before Laboratory Period(s)

Before you come to lab you should do the following:

1. Design your task and write pseudo-code describing your task. Submit your pseudo-code prior to your laboratory period according to your GTA's instructions.
2. Write your task using a text editor. Make sure your task is well organized, well formatted, and documented.
3. Use the computers in the lab to compile and debug your task.
4. Bring a printed version of your task to your lab according to your GTA's instructions.

You may use parts of the task you have run in lab or other programs. However, if you use code from any other program, you must cite the source of that code in your report.

You may ask questions about the lab in lecture.

### 2.2 In Lab

1. Show your lab instructor your printed listing of your task according to your GTA's instructions.
2. Demonstrate your task to the lab instructor.
3. Be prepared to answer questions about your task asked by your lab instructor.

### 2.3 Complete the Laboratory

Collect your measurements, logs, copies of your task in your directory, and clean up your lab bench.

Instructions for writing your report will be provided by your lab instructor.

## 3 Program Description

Your project will implement the following general activities (along with LED Blink and SysTickCount reporting):

1. Your task will monitor the two buttons on the evaluation board.
2. Your task will generate a tone when the switch press is detected and when the switch is released.
3. Your task will report the identity of the switch released via the serial communication port.
4. You do not have to account for more than one switch pressed at the same time nor for rapid, less than 2 Seconds interval, pressing of the switches.

### 3.1 Button Monitoring and Tone Generation

1. Your task will monitor the two buttons on the evaluation board. You shall check the button state once each 1 ms.
2. When a button transition is detected, your task shall wait 10 ms to insure the button is actually pressed. This is the de-bounce period.
3. When a button is pressed and switch bounce is compensated for, your program will emit a 440 Hz tone on the speaker for 0.2 seconds.
4. When a button is released, your program will emit a 440 Hz tone for 0.5 seconds and report the event via the UART. The message shall state: "Button

<x> released.\n." Where <x> is the identity of the button, e.g. "SW1" or "SW2."

## 4 Program Structure

### 4.1 Button Monitor Task Setup

You program must configure GPIO PortJ to read the two buttons. The buttons are connected to PortJ<1..0>. The actions are:

1. Enable PortJ with SysCtlPeripheralEnable.
2. Set PortJ<1..0> pins to input with `GPIOPinTypeGPIOInput()` subroutine `from DriverLib`.
3. Set PortJ<1..0> pins to weak pull-up with `GPIOPadConfigSet`. You do not have to set the drive current on these pins.

### 4.2 Button Monitor Task

Your task must:

1. Read the state of two buttons every 1 ms.
2. If a button is released, go to step 6.
3. If no button is pressed, exit.
4. If a button is pressed, wait 10 ms.
5. If the same button is pressed after 10ms, declare a button press and emit a 0.2 S tone at 440 Hz.
6. On release of a button, emit a 0.5 s tone at 440 Hz and send a message using the UART.

### 4.3 UART Setup

You can access the UART using the follow statements in appropriate locations in your program:

```
#include   "Drivers/UARTStdio_Initialization.h"
#include   "Drivers/uartstdio.h"
```

These statements import the APIs for the UART.

```
UARTStdio_Initialization();
```

This statement initializes the UART.

To use the UART, use a statement similar to:

```
UARTprintf( "FreeRTOS Starting!\n" );
```

### 4.4 COM setup on the Windows PC

See Appendix A for setting up serial communications on Windows PC machines.

### 4.5 Modifications to SpeakerBuzz

Replace your SpeakerBuzz task from Lab #02 with a modified SpeakerBuzz task.

The structure of your new SpeakerBuzz task will be discussed in lecture.

# Appendix A – Using PuTTY

This is a guideline for using PuTTY to communicate over the serial interface between the workstation and the TM4C1294 evaluation board. The basic steps are:

1. Determine the Windows COM interface number.

2. Start and configure PuTTY.

3. Use TI Code Composer to load and execute your program.

To determine the Windows COM interface number, do the following.

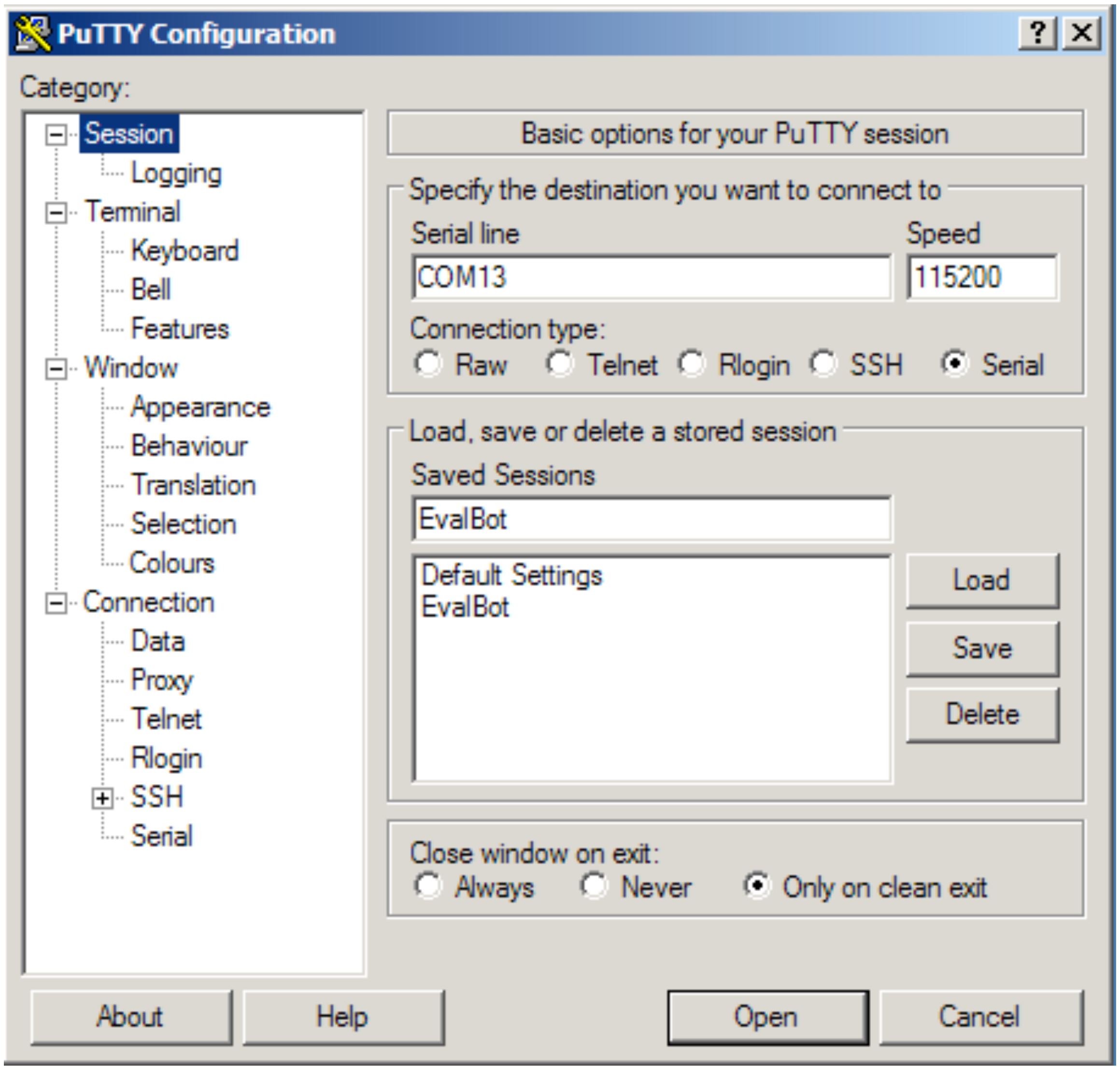
1. Attach the TM4C1294 evaluation board to the USB cable. The board must be powered up.
2. Open the Control Panel from the Windows Start menu.
3. Select "Hardware and Sound"
4. Under "Devices and Printers" select "Device Manager"
5. Expand the "Ports (COM & LPT) item
6. You should see an entry similar to: "Stellaris Virtual COM Port (COM11)
7. Remember the specific COM port on your workstation.

Configuring PuTTY

1. Start PuTTY from the Windows Start menu. You should see a dialog similar to Figure 1.
2. Set the COM port to the port you found in the previous process, set the Connection Type to "Serial," and set the Speed to 115200.
3. You can set other serial parameters by selecting "Serial" in the left hand menu. In the Serial dialog you can set the COM port, speed, number of data bits, parity, stop bits, and flow control. An example is shown in Figure 2.
4. Select "Session" in the left hand menu. You can name and save your configuration for use in the future. However, there is no guarantee that the COM port number will be the same in the future or that your settings will be maintained from workstation to workstation. I've just not tried that from an EECS student account.
5. Select "Open" and a session window should appear. Typing in the window sends characters to your program on the TM4C1294 evaluation board. Characters your program transmits are shown in the window. Note, there is no automatic "Echo" of characters. The only characters that show in the window are those transmitted by your program.

That's all for configuring PuTTY to use with your program.

Figure 1 shows a PuTTY Session Dialog.


PuTTY Configuration

Category:

Basic options for your PuTTY session

Specify the destination you want to connect to

Serial line: COM13

Speed: 115200

Connection type: Raw Telnet Rlogin SSH [✓ Serial]

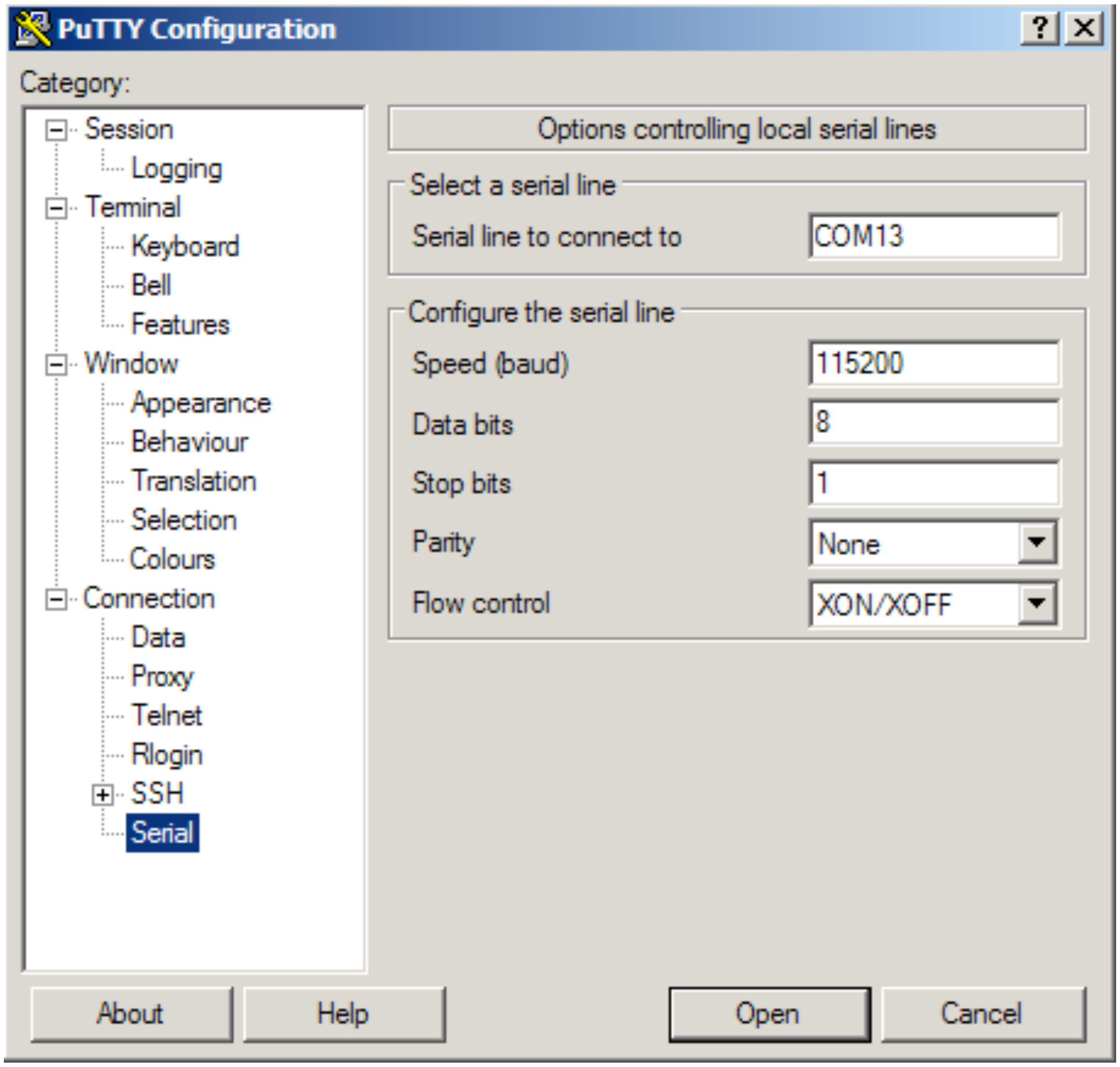
Load, save or delete a stored session

Saved Sessions: EvalBot

Default Settings
EvalBot

Close window on exit: Always Never [✓ Only on clean exit]

Figure 2 shows a PuTTY Serial Dialog.



Options controlling local serial lines

Select a serial line

Serial line to connect to: COM13

Configure the serial line

Speed (baud): 115200

Data bits: 8

Stop bits: 1

Parity: None

Flow control: XON/XOFF