UNIVERSITY OF CALIFORNIA, DAVIS
Department of Electrical and Computer Engineering

EEC 172 **Spring 2019**

LAB 4: DTMF Texting Over an Asynchronous Serial (UART) Link

Verification Due: May 22nd MW or May 23rd TR at the end of your lab section
Report Due: May 29th MW or May 30th TR in your lab section

Objective: Increasingly embedded systems include audio input. Consider for example Amazon's Alexa or Google Home. In this lab we will create a simple audio interface to our embedded systems, specifically a touch-tone phone input. You will design a system to decode dual-tone multi-frequency (DTMF) audio signals coming from your phone, and use the input to compose text messages, similar to the way you used the IR Remote in Lab 3. You will use your phone to produce DTMF signals and use the same multi-tap text entry system to generate text messages. You can install a DTMF tone generation app on your phone or use a web-based tone generator. You will use SPI to interface to an external analog-to-digital converter (ADC) as well as the OLED. You will use the https://en.wikipedia.org/wiki/Goertzel_algorithm to implement DTMF detection and decoding. The final program will be the same board-to-board texting application used in Lab 3, except you will use DTMF audio signals instead of IR Remote signals to enter the characters.

New Equipment Needed Per Group

2 Adafruit Electret Microphone Amplifier - MAX9814 with Auto Gain Control (product id: 1713)
2 LM1086CT-3.3 Low Dropout +3.3V Voltage Regulator (3-pin TO-220)
2 Microchip MCP3001 10-bit A/D Converter with SPI Serial Interface
Jumper wires
Decoupling capacitors

Resources (Technical Documents)

Refer to the following technical documents, which are available on the course website:

- Adafruit Microphone User Guide (adafruit-agc-electret-microphone-amplifier-max9814.pdf)
- Maxim MAX9814 Microphone Amplifier with AGC and Low-Noise Microphone Bias DataSheet
- LM1086 DataSheet (lm1086.pdf)
- Microchip MCP3001 DataSheet (mcp3001.pdf)

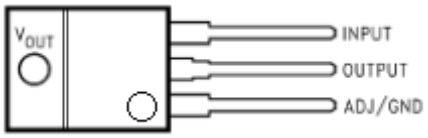
Hardware Interface

The hardware interface consists of three components: the electret microphone, SPI A/D converter and a low dropout voltage regulator.

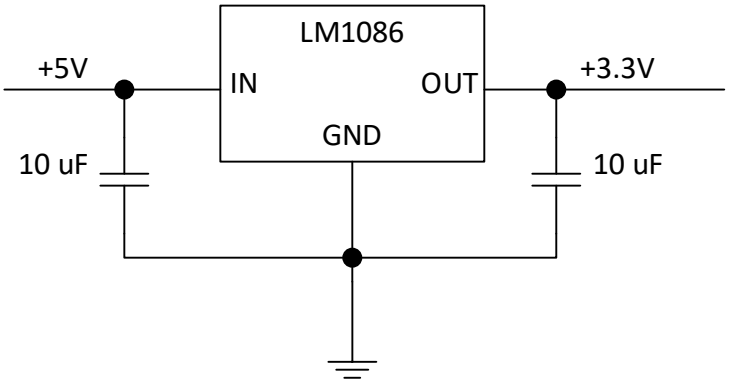
1. LM1086CT-3.3 Low Dropout Voltage Regulator

   For best performance, the microphone must be connected to a "cleanest" +3.3V power supply available because noise on the power supply line will be amplified and will corrupt the microphone's output signal. Because the OLED generates significant noise on the power supply line, we will use a separate power source for the OLED and for the microphone. Thus, the first step in the interface design is to put the OLED on its own power source so that it does not corrupt the audio signal from the microphone.

   Use the LM1086CT-3.3 low dropout voltage regulator to supply the +3.3V power to the OLED. The voltage regulator is a simple 3 terminal device with pins for input, output and ground, as shown below.



Use the +5V output signal from the CC3200 LaunchPad as the input voltage. The Adj/Gnd should be tied directly to ground and the output will be a fixed +3.3V used to power the OLED. You should use 10 uF bypass capacitors at the input and output pins, as shown in the figure below. (Electrolytic capacitors can be used instead of tantalum). Note that the regulator and OLED ground should be connected to a LaunchPad ground pin. The Microphone and ADC will also have their ground pins tied to a LaunchPad ground pin. Aside from having a common ground on the CC3200 LaunchPad, the [OLED and LDO ground] should be isolated from the [microphone and ADC ground] in order to reduce noise.

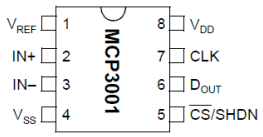


2. Electret Microphone with Amplifier

The electret microphone with amplifier has five interface pins: AR (Attack/Release Ratio Select), Out, Gain, VDD, and GND. You should study the MAX9814 datasheet and the Adafruit electret microphone user's guide to understand how to configure the Gain and AR pins. We recommend setting the gain to 50 dB by tying the Gain pin to GND. The gain at 50 dB produces an Out signal of reasonable amplitude without much noise. When the gain is 60 dB, the noise is much more apparent in the signal. You can verify your output signal at different gains on the oscilloscope. The Attack and Release Ratio (A/R) also has three options. From the AGC Electret Microphone User Guide, you can see that the CT capacitor has been chosen to be 100 nF. Based on Table 2 of the MAX9814 data sheet, this will give an attack time of 0.24 ms and a release time of 120 ms for A/R tied to GND and 480 ms for A/R tied to VDD. For this lab, we recommend tying A/R to GND for a release time of 120 ms. Applications detecting tones, such as DTMF detection, typically requires a shorter release time than applications detecting speech. For a speech recognition application, it may be useful to tie A/R to VDD. VDD should be connected to the +3.3V signal from the CC3200 LaunchPad and GND should be connected to a ground pin on the CC3200 LaunchPad.

3. MCP3001 10-bit A/D Converter

The MCP3001 ADC will be interfaced to the CC3200 LaunchPad using SPI. The interface pins for the MCP3001 are shown in the figure below.



You should use a GPIO signal for the /CS pin so that you can interface both the MCP3001 and the OLED to the same SPI port. Your CC3200 hardware interface will use one GPIO signal to control the MCP3001 /CS pin and another GPIO signal to control the OLED OC pin. Thus, you can access these two SPI devices individually.

Software

1. DTMF detection and decoding

   The Goertzel algorithm can detect DTMF tones more efficiently than a more general Discrete Fourier Transform (DFT). Therefore, you should base your tone detection scheme on the Goertzel algorithm. You should do a web search to learn how the Goertzel algorithm works and to locate example code that implements the Goertzel algorithm. For example, this project on github implements the Goertzel algorithm:

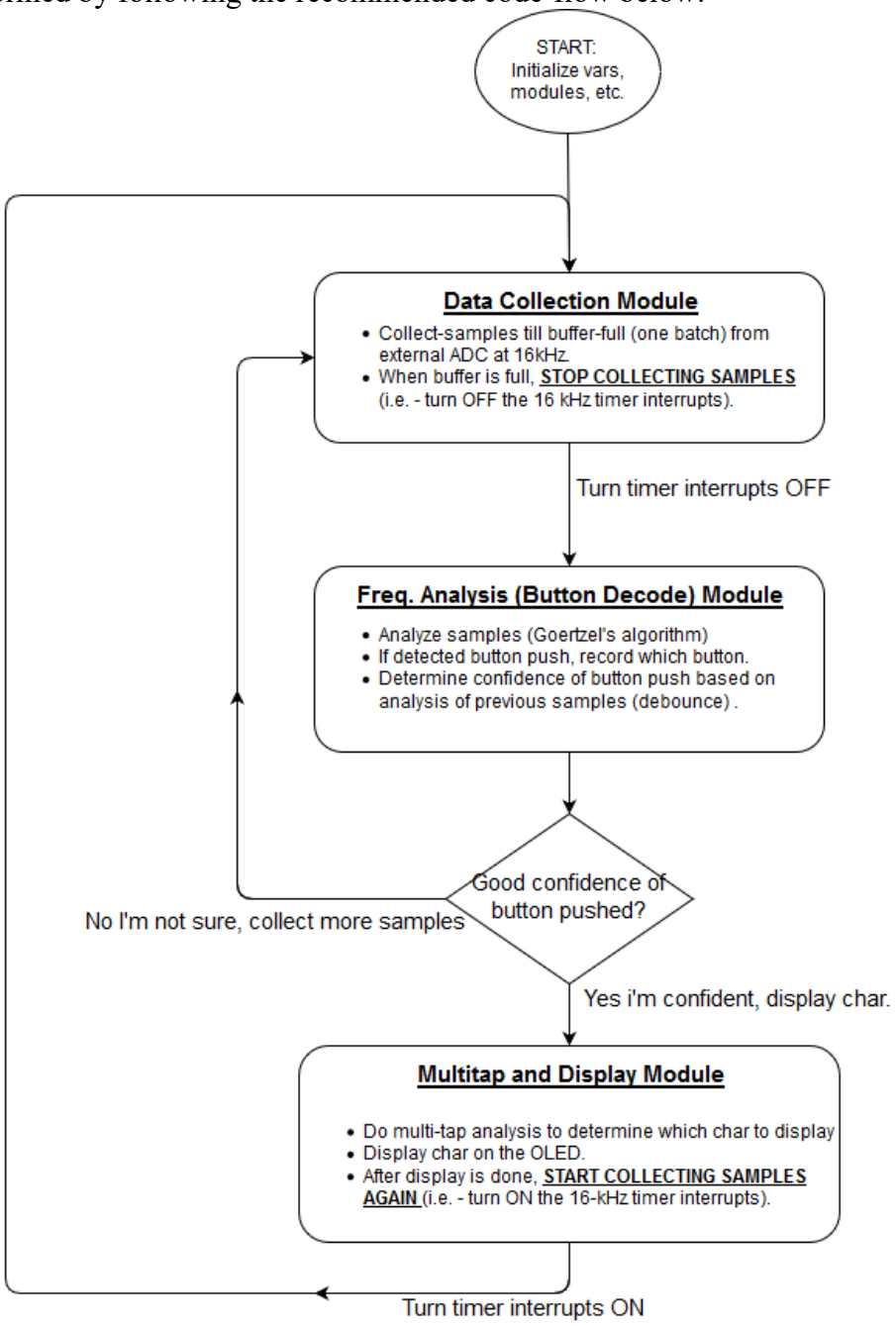
   https://github.com/OmaymaS/DTMF-Detection-Goertzel-Algorithm-

   This project is a good starting point for your detection code. However, if you use this code, some revisions are need:

   - Your system **must use a 16 kHz** sampling frequency. You should use a general-purpose timer to generate interrupts at 16 kHz to periodically sample the ADC.
   - The recommended block size should be **410 samples** (rather than 96 used in the example program) when using our sampling frequency.
   - Your SPI bit rate should be *at least* **400 kbps**.
   - For efficiency, your program should use *only* fixed-point arithmetic so that you do not need to use floating point libraries, which are large and slow. In the example GitHub project, floating point is used to calculate fixed point coefficients. Instead, this should be done off-line so that the program does not include floating point libraries. If you are unfamiliar with fixed point representation, see pages 234-235 of our textbook. To check that you are calculating the coefficient correctly, the correct coefficient for 697 Hz is about 31548, with a 16 kHz sampling rate.
     - Your system uses a different ADC from the GitHub project, which implies that you will need to write code specifically for the MCP3001. Note that the MCP3001 outputs data in big endian format. Also note that, your code (not your circuit) should remove the DC-bias of the signal from the microphone before processing it with the Goertzel algorithm.

2. Recommended Code Flow

Performing a continuous measurement and analysis from your microphone, while also performing OLED printing can make dealing with race conditions in the management of the shared SPI port more complex. To help guide you towards a quick and easy integration, we recommend you perform *batch processing* instead of trying to do continuous measurements.
This can be performed by following the recommended code-flow below:



Following this code flow for your program will ensure that your SPI bus does not attempt to interleave communication between your OLED and the external ADC chip (thus reducing complexity), though it does add some delay. Note that you are NOT communicating with the OLED while you are collecting data, and conversely you are NOT collecting data (communicating with the ADC) when displaying to the OLED. Don't forget to **disable your chip-select** at the end of the appropriate modules. You should try to make your analysis and display code as efficient as possible so that the delay is as short as

possible. Since the user is continually inputting tones, the program needs to collect/detect enough samples to ensure nothing is missed.

3. Texting

   As in Lab 3, you will connect your group's two CC3200 LaunchPads using asynchronous serial lines TX and RX of UART1. Your two CC3200 LaunchPads will each be interfaced to an Adafruit color OLED display.

   Develop an application that uses DTMF from your phone to compose text messages using the multi-tap text entry system and send text messages back and forth (i.e. - bidirectional) between your two CC3200 LaunchPad boards over UART1. The application must allow the user to input a text string via the DTMF signals and display the message on the local OLED. After the # key is pressed the message is transmitted to the remote board and displayed on the remote OLED display. In multi-tap texting, the intermediate characters should be displayed. For example, when sending the character 'z' the '9' button will be pressed 4 times, so the intermediate characters 'w', 'x', 'y' will appear at the current character position before 'z' appears. Pressing the * key should delete the last character, to allow for correction of an errant character before a message is sent.

   As in Lab 3, incoming text messages should be displayed on the bottom half of the OLED and messages under composition should be displayed on the top half of the display.

Verifications Needed:

- Demonstrate your working DTMF decoding program to your TA for verification.
- Demonstrate your multi-tap texting program via DTMF to your TA for verification.

Lab Report

At a minimum your lab report should include:

- Your verification sheet.
- A hard copy of your well-written, *well-commented* final program. Include your names in the header comments of the file containing the main program!
- A soft copy of your code uploaded to SmartSite.
- A description of any noteworthy difficulties you encountered in constructing your solution.
- A description of any intersesting, amazing, or amusing extension you made to the project that might warrant bonus points.