Software Overview

Year: 2024 Semester: Spring Team: 12
Project: USB Microphone Interface
Creation Date: 1/26/2024 Last Modified: 1/27/2024
Author: Jakub Kowalski Email: kowals14@purdue.edu

Assignment Evaluation: See Rubric on Brightspace Assignment

1.0 Software Overview

Software functionality of USB Microphone Interface:

- USB communication
- Digital signal processing
- Updating DSP parameters

1.1 USB Communication

USB communication will be between the interface and the host device. This will be done by following the USB 2.0 Protocol and configuring the interface according to USB Audio Class 1.0 specification. Upon connecting the interface to the host via USB cable, the host device will initiate communication via sending packets of data that request the connected device (our interface) to identify itself. These packets will be transmitted at "full speed" which is defined as 12Mbps. Once the device receives the request, it will send back necessary descriptors to identify itself as an Audio Class input device and identify parameters such as sample rate for the audio, number of channels, etc. At this point, the drivers on the host device will recognize the interface as an input device and it will be possible to start sending packets of data from the interface to the host device. The nature of packets is discussed further in section 3.

1.2 Digital Signal Processing

Digital signal processing will be implemented through our own library which will design basic DSP algorithms such as IIR filters and possibly the DFT transform using FFT algorithm as a stretch goal to visualize the frequency spectrum of the signal. We will integrate them into effects such as a parametric EQ, compression, and delay.

1.4 Updating Parameters

The rotary encoders and buttons on our interface will serve as inputs to adjust parameters for the DSP algorithms and cycle through which the effect is currently being edited by the user. Upon receiving input from the rotary encoders/buttons, we will raise an interrupt. The ISR will handle updating the value of the parameters / current effect being edited and set a flag for updating the GUI. This is done to avoid causing too much delay between sending the data from the codec, applying DSP, and outputting through USB. The flags will be checked after audio is output, and the UI will then be updated.

2.0 Description of Algorithms

2.1 IIR Filters w/ Lookup Table

We plan on building our own DSP library to implement FX such as parametric EQ, compression, and delay. A basic DSP algorithm we need to implement is filtering, more specifically IIR filters due to their faster computation time [1]. The IIR filter equation consists of finding the sum of weighted past inputs and outputs from the system. To calculate necessary coefficients for the IIR filters while saving time we must also implement a look up table for trig functions such as sin/cos [2]. The exact implementation of the table is still to be determined but the plan is to create a map of key-value pairs where the key will correspond to the input to a trig function and the mapped value is the result. To save on memory, we can also take advantage of the periodic nature of the sin/cos functions to define a base set of trig key-value pairs for a range of inputs and determine if new inputs will be equivalent to them.

2.2 DFT

In addition to the IIR filters, as a stretch goal we plan to integrate DFT into our EQ to visualize the frequency spectrum when filtering the signal. The standard way of implementing the DFT is using the Fast Fourier Transform (FFT) algorithm as described by Cooley and Tukey [3]. The algorithm consists of continuously dividing the signal buffer into smaller buffers for which you find the DFT and recombine them through butterfly operation. To perform this operation, you must multiply by what are known as “twiddle factors”, which are complex numbers. To make this more efficient, the twiddle factors can also be precomputed and stored in memory rather than computed dynamically.

3.0 Description of Data Structures

3.1 Lookup Table

When implementing the lookup table, an option to consider is using map as defined in the STL [4]. As described in the reference, maps are “usually implemented as Red-black trees”. Red-black trees are a kind of balanced binary trees, and they utilize rules to maintain their balance to guarantee that the time complexity for insertion, deletion, and searching is always O(logn) [5].

3.2 User Parameters

When reading in parameter values from the user, we will make use of structs which will define the variables of certain parameters such as filter type, bandwidth, center frequency, etc. These structs will be passed in as arguments to define the behavior of the current effect.

3.3 Audio Buffers

We will be making use of arrays to store the audio data to be buffered when applying filters and other effects to the audio signal.

3.4 USB Packets

Communication between USB devices and hosts is done via packets sent in transactions. These packets have several different types, which vary in structure and size. The general structure of a packet consists of a synch pattern, a byte-sized PID to identify the packet, varying amount of data, and finally an end-of-packet (EOP) [6]. Transactions consist of three phases [7]: Token, Data and Handshake. The type of transaction is defined by the Token packet being sent over, which indicates what direction the transaction will go in, and what kind of data is sent over. Collections of transactions are known as transfers [6]. In our device, we will be using isochronous transfers, which send data continuously without a handshake. These transfers are ideal for audio/video transmission and have a maximum data payload of 1023 bytes for full speed devices [7].

4.0 Sources Cited:

[1] S. W. Smith, "Chapter 19: Recursive Filters," in *The scientist and engineer's guide to digital signal processing*. San Diego, Calif.: California Technical Pub, 1997.

[2] S. W. Smith, "Chapter 4: DSP Software," in *The scientist and engineer's guide to digital signal processing*. San Diego, Calif.: California Technical Pub, 1997.

[3] J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of Computation*, vol. 19, no. 90, p. 297, Apr. 1965, doi: https://doi.org/10.2307/2003354.

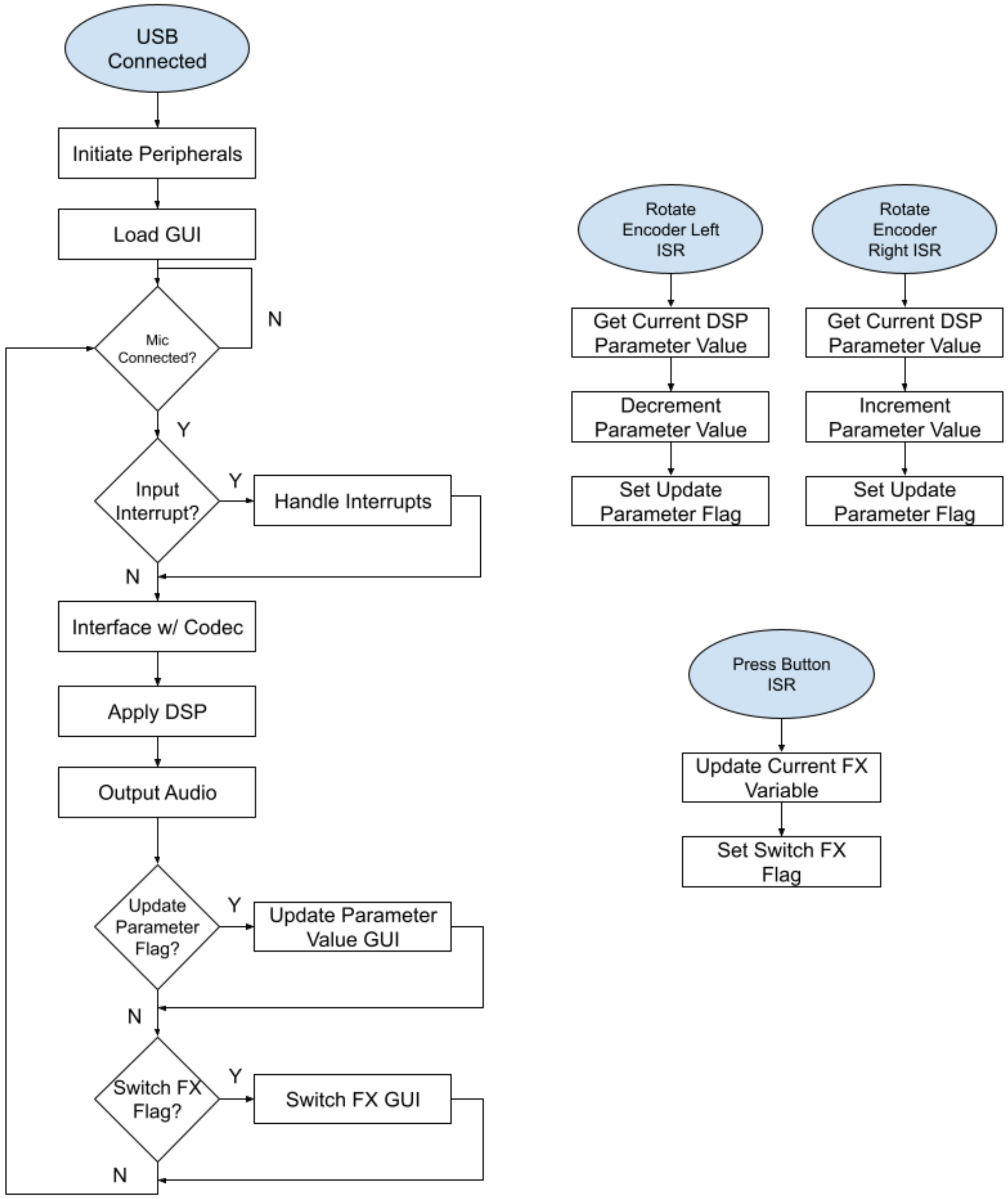
[4] "std::map - cppreference.com," *Cppreference.com*, 2019. https://en.cppreference.com/w/cpp/container/map

[5] GeeksforGeeks, "Introduction to Red-Black Tree," *GeeksforGeeks*, Feb. 04, 2014. https://www.geeksforgeeks.org/introduction-to-red-black-tree/

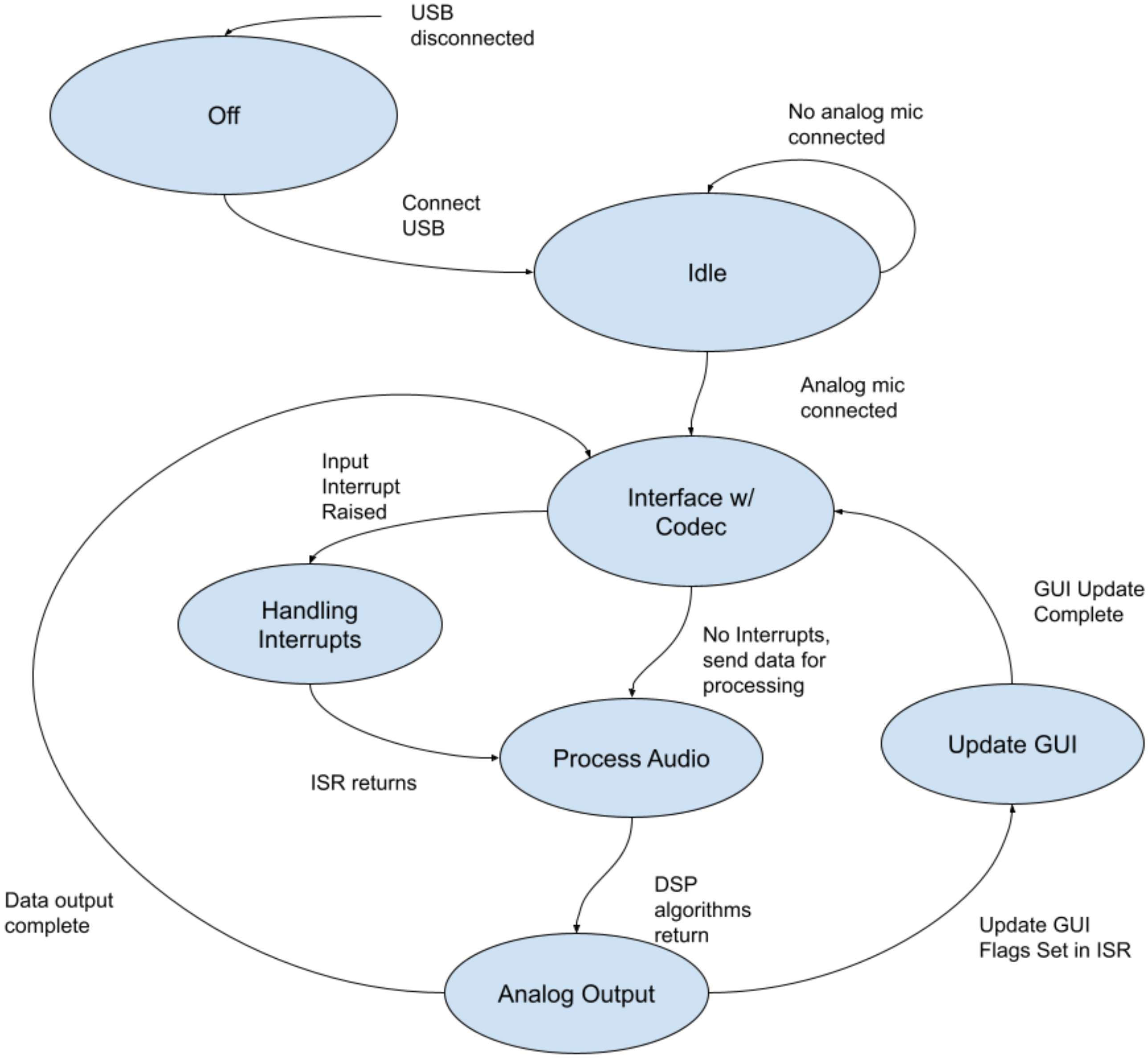
[6] Sine Lab, "Add USB To Your Electronics Projects! - The USB Protocol Explained," *www.youtube.com*. https://youtu.be/HbQ6q3skZgw

[7] A. Singh, "USB Protocol: Types of USB Packets and USB Transfers (Part 2/6)," *Engineers Garage*. https://www.engineersgarage.com/usb-protocol-types-of-usb-packets-and-usb-transfers-part-2-6/#:~:text=Types%20of%20USB%20packets%201%20%20For%20low

Appendix 1: Program Flowcharts



Appendix 2: State Machine Diagrams


USB disconnected
Off
Connect USB
No analog mic connected
Idle
Analog mic connected
Input Interrupt Raised
Interface w/ Codec
Handling Interrupts
No Interrupts, send data for processing
GUI Update Complete
Process Audio
Update GUI
ISR returns
Data output complete
DSP algorithms return
Update GUI Flags Set in ISR
Analog Output