Overview

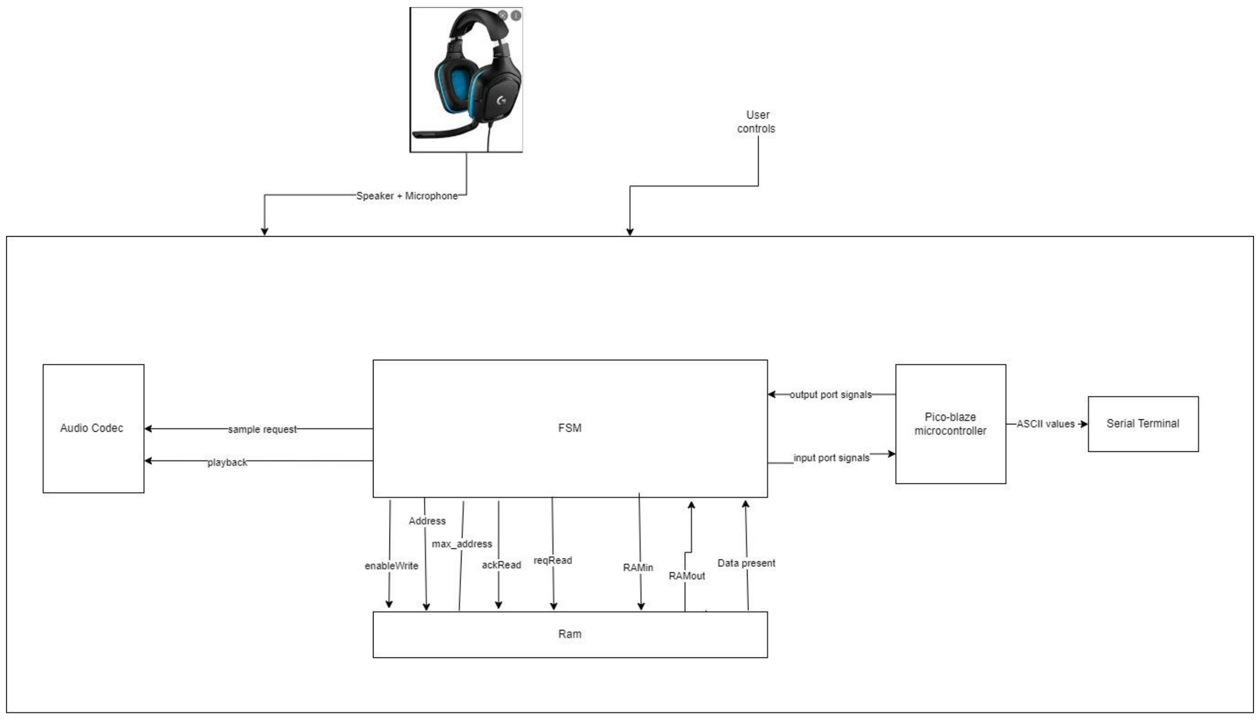
We combined three components to make our project functional. The picoblaze microcontroller, the audio codec, and the ram interface.

The picoblaze microcontroller was used to display the menu on the serial terminal. From the menu you may select options such as play a message, record a message, delete a message, delete all messages and volume control.. If playing a message, recording a message, or deleting a message is selected then you are to choose tracks one through five to specify which track the action should take place on. In the case you are playing a message you will also have the options to play, pause or skip a track. In the case you are recording a message you have the option to start or stop the recording. If you traverse to a menu option there is also an option to go back from each respectively.

The audio codec provided the ability to sample the microphone connected and output audio to the headset connected. Given proper signaling the audio codec will read from the ram component and play the requested track via its specified address space. In the event of recording a message it will sample the microphone and interface with the ram to store the information in the specified address space for each track.

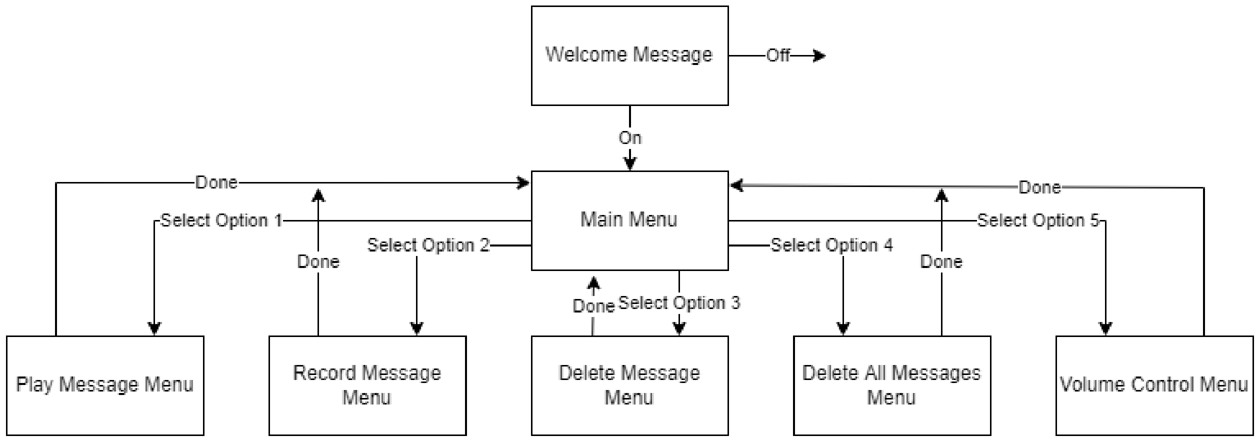
The ram component accessed a sixteen million bit address space split into five for each track. If a track was to be read the proper acknowledge and read signal would be given to it and then it would interface with the audio codec to provide the correct bits stored at the specified track address space to play a recorded message. If the a message was to be recorded then the write enable would be signaled and the proper address would be given so that the audio could be sampled from the audio codec and stored in the address space for each track until the max address space per track was reached or the recording was stopped.

<u>System Block Diagram</u>



System Functional Flow

We will put the FSMD here and detail its interfacing

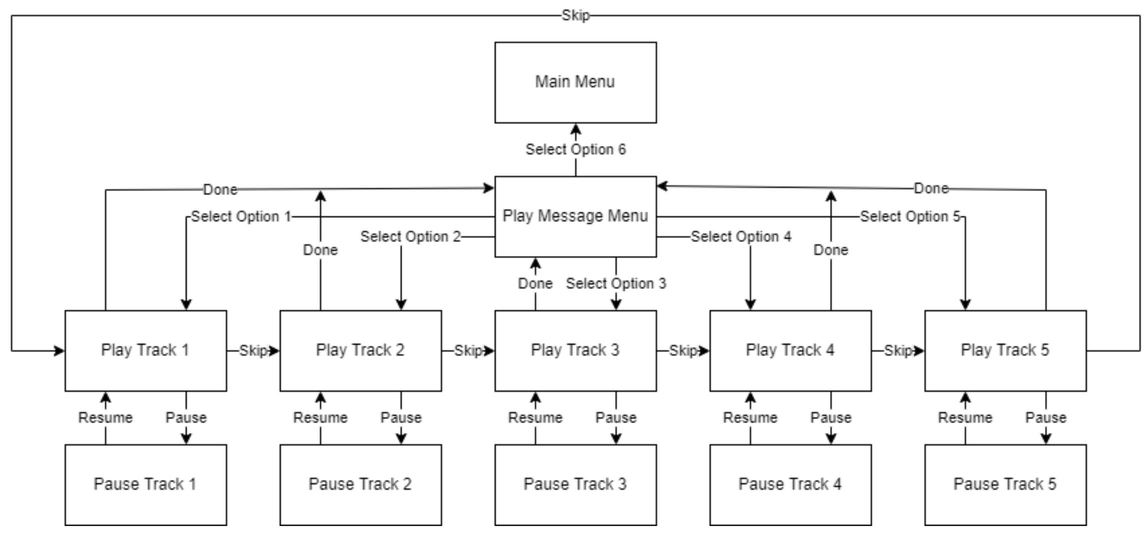


When we first turn on the Audio Message Recorder we begin at the Welcome Message Screen state. If the user hits the continue button, the main signal is set to 1 and we go to the Main Menu state.

At the main menu state, the user has the option to play an existing message, record a new message, delete an existing message, delete all messages, or control the volume. If the user selects option 1, the main_play control signal is set to 1 and we jump to the Play Message Menu state. If the user selects option 2, the main_record control signal is set to 1 and we jump to the Record a Message Menu state. If the user selects option 3, the main_delete control signal is set to 1 and we jump to the Delete a Message Menu state. If the user selects option 4, the delete_all signal is set to 1 and we jump to the Delete All Messages Menu state. If the user selects option 5, the change_v signal is set to 1 and we jump to the Volume Control Menu state.

Once we are finished with each of these states, each control signal is set back to 0 and we jump back to the Main Menu state.

Play A Message



In the Play a Message Menu state we are faced with 6 options. If the user selects option 1, the play_track1 signal is set to 1 and we jump to the Play Track 1 state. If the user selects option 2, the play_track2 signal is set to 1 and we jump to the Play Track 2 state. If the user selects option 3, the play_track3 signal is set to 1 and we jump to the Play Track 3 state. If the user selects option 4, the play_track4 is set to 1 and we jump to the Play Track 4 state. If the user selects option 5, the play_track5 signal is set to 1 and we jump to Play Track 5 state. If the user selects option 6, all play_track# control signals are set back to 0 and we jump back to the Main Menu state.When we are in each of the Play Track # states, we are given 4 options, to play, pause, skip, or go back.

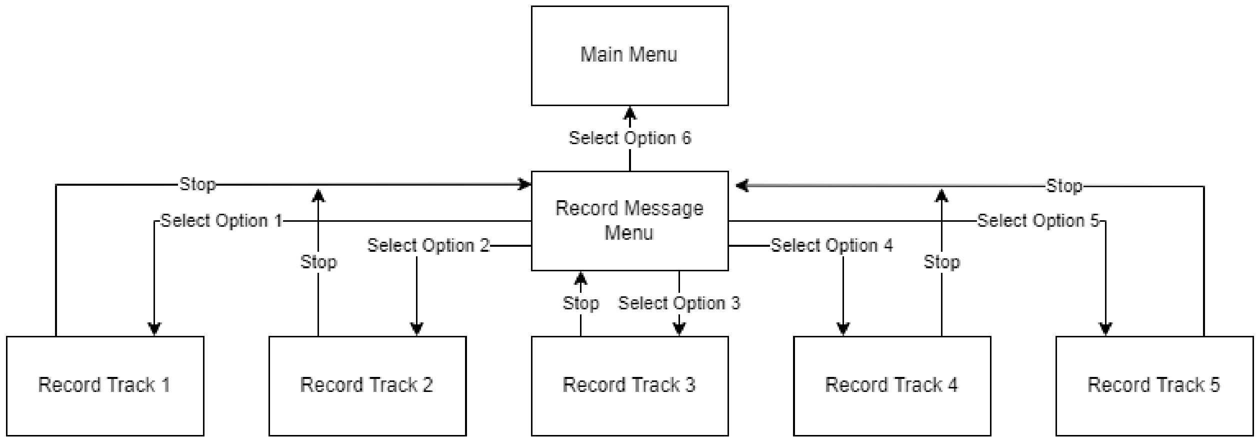
If the user decides to play, the playback signal is set to 1, enableWrite is set to 0, and reqRead is set to 1. We then set reqRead to 0, read the RAM at the given address of the track, and set ackRead to 1. Finally, we set ackRead to 0, increment the address by 1, and jump back to the start of play. We continue this loop until the address is greater than or equal to the max address of the given track.

If the user decides to pause, the pause signal is set to 1 and the play_track# is set to 0, and we remain in the Pause state. When we want to resume playing the track, the pause signal is set back to 0 and the play_track# signal is set back to 1 and we jump back to the Play Track # state.

If the user decides to skip the track, the skip control signal is set to 1, the address of the next track is stored and we jump to the Skip state. The skip_ack signal is set to 1 and play_track# signal is set to 1. We then set the skip_ack signal to 0 and we jump to the Play Track # state.

If the user decides to go back, the play_track# signal is set to 0 and we jump back to the Main Menu state.

Record A Message

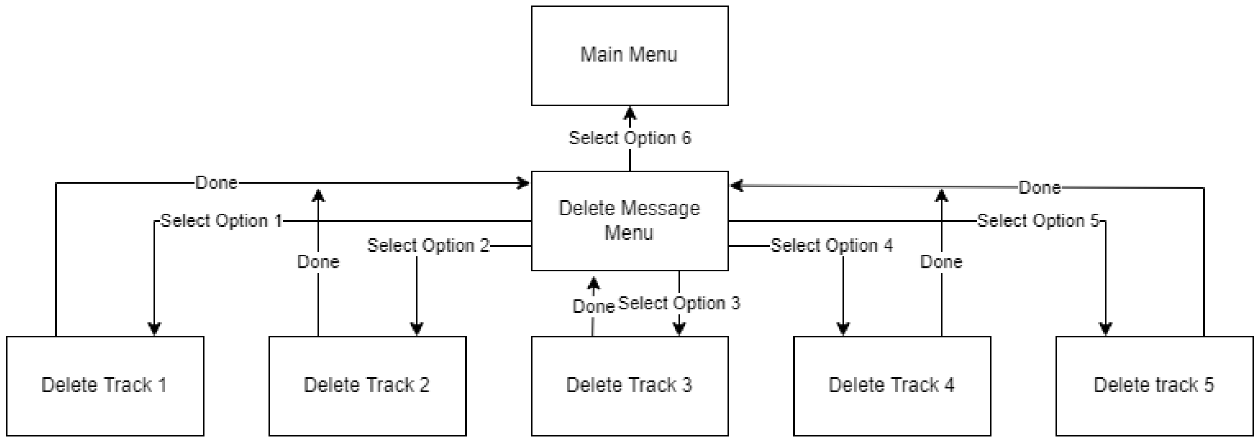


In the Record a Message Menu state the user has 6 options: record to track 1, record to track 2, record to track 3, record to track 4, record to track 5, and go back.

If the user decides to record to track 1, the record_track1 signal is set to 1 and so on for all 5 tracks where we jump to Record Track # state. Then the playback signal is set to 1. If the sample_available signal is 0 then the sample_a_check signal is set to 1. Otherwise if sample_available and sample_a_check are 1 then sample_a_check is set to 0, we increment the address, and set enableWrite is set to 1. Then we set enableWrite to 0, if the address is less than the ending address of the track, we jump back to the Record Track # state. If the address is greater than or equal to the ending address then the record_state# signals are set back to 0 and we jump back to the Main Menu state.

If the user decides to go back, the record_track# signals are set to 0 and we jump back to the Main Menu state.

Delete A Message

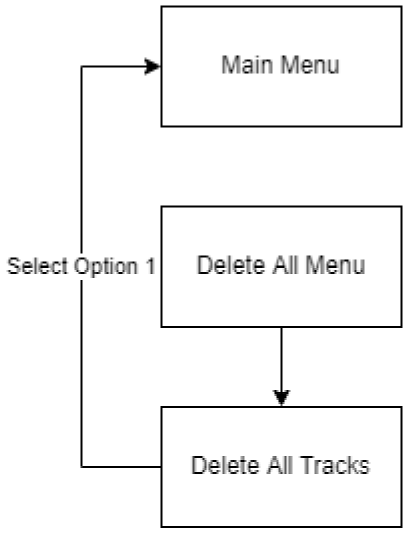


In the Delete a Message Menu state we are given 6 options: to delete track 1, delete track 2, delete track 3, delete track 4, delete track 5, and go back.

If the user decides to delete any track, the delete_track# signal is set to 1. We then set playback to 0 and enableWrite to 1, and input a 0 into RAM at the given address location of the track. EnableWrite is then set back to 0 and the address is incremented. If the address is greater than or equal to the ending address of the track then deleted_all is set to 1, delete_track# is set to 0, we jump to the Delete a Message state. If the address is less than the ending address then we jump back to the start of Delete Track # state.

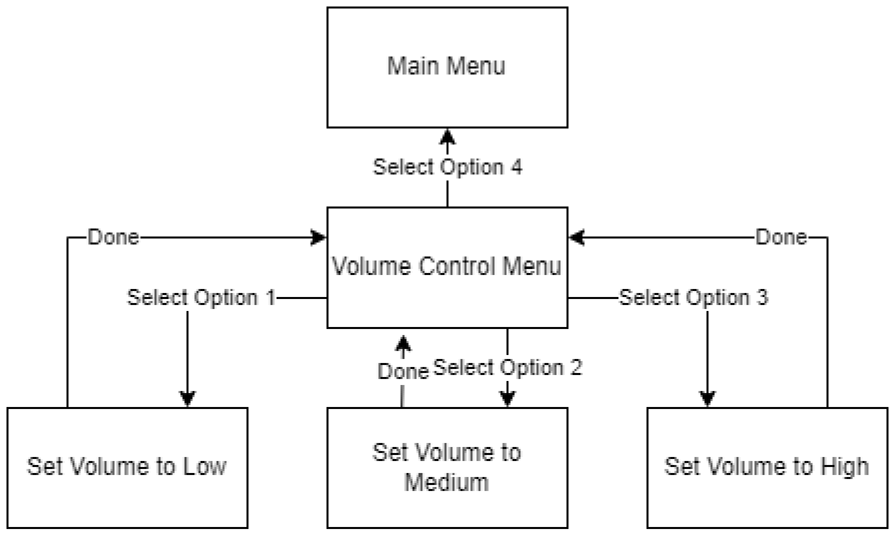
If the user decides to go back, the main_delete signal and the delete_track# signals are set to 0 and we jump back to the Main Menu state.

Delete All Messages



In the Delete All Messages Menu state playback is set to 0 and enableWrite is set to 1, and input a 0 into RAM at the given address location starting at address 0. Enable write is then set to 0 and the address is incremented. If the address is greater than or equal to the max_ram_address the deleted_all signal is set to 1, we jump to Main Menu state and reset the deleted_all back to 0. Otherwise if the address is less than the max_ram_address then deleted_all is set to 0 and we jump back to the start of Delete All Messages state.

Volume Control



In the Volume Control state the user is given 4 options: set the volume low, set the volume medium, set the volume high, and go back.

If the user decides to set the volume to low, the low signal is set to 1 and a 1 is sent to the volume_control register. We then jump back to the Volume Control state.

If the user decides to set the volume to medium, the med signal is set to 1 and a 2 is sent to the volume_control register. We then jump back to the Volume Control state.

If the user decides to set the volume to high, the high signal is set to 1 and a 3 is sent to the volume_control register. We then jump back to the Volume Control state.

If the user decides to go back, the change_v signal is set to 0 and we jump back to the Main Menu state.

Discussion and Conclusions

In this section, discuss how you arrived at your major decisions (why you put X in hardware and Y in software, how you selected the memory chips you used, etc.). Discuss any aspects of your design that you would do differently if you had it all to do over again, and why. Include any and all thoughts you have on the project, how it could have been done differently or better, etc.

We were required to use the audio codec and ram files provided so that we could interface with the anvyl board so we did not have the freedom to choose otherwise. In terms of design decisions we would do differently I think we would try to condense our fsm to have fewer states so that it would be easier to work with and increase readability. If we could also work on a way to clock the audio codec at a similar speed as the ram it would negate the need for the clocking wizard as that caused many timing issues having multiple different clocks.