W5UXH-Hasak Description

PCB Layout Version-B, SN-2 is shown below. The OLED display is in the scrolling text mode.

Table of Contents

1 Introduction

Probably in early April 2021 I started experimenting with the Hasak keyer project by Roger Critchlow, AD5DZ, after it was brought to my attention by Chuck Vaughn, AA0HW. It uses a Teensy 4.0 or 4.1 board.

My goal for the Hasak is for use with iCW / Mumble or as a standalone keyer, not requiring any host computer software other than Mumble. I have added features that require a few edits in the original source code from Roger. Subsequent updates from Roger will require that I make the same edits again. I have attempted to include “W5UXH” at each point where I have done edits so that hopefully I can update when needed without too much difficulty.

This document has far more information than most others would want to dig through, so the “executive summary” is the following.

1.1 Summary

Roger has implemented a high quality sidetone plus a sound card in the Teensy processor. In the most basic implementation you have a plug-and-play iCW paddle keyer that requires one USB cable to a computer running Mumble, a paddle, and headphones or a powered speaker. Roger’s software also provides an audio mixer so the local sidetone is mixed with the received signal from the iCW Mumble server.

My implementation adds software that provides a CW keyboard using a USB keyboard, plus a decoder, and an alternate paddle keyer. I include the alternate paddle keyer because it is my software that I have used for years. But Roger’s paddle implementation works very well. I also added a tiny OLED display and a rotary encoder for controlling speed and sidetone frequency.

2 Roger Critchlow, AD5DZ GitHub link

Roger's text description of the project will be found by scrolling down the page on the GitHub site: https://github.com/recri/hasak

2.1 What does HASAK stand for?

From the GitHub link we find this name:

> **H**am **a**nd **S**wiss **A**rmy **K**nife

3 Important features provided by the Hasak software

3.1 USB Sound Card implemented

The project was of great interest to me because it implements a high quality sidetone and a USB sound card using the Teensy 4.0. When connected to a host computer USB port, the Hasak appears as an audio device with the name "hasak".

3.2 Audio mixer combines local sidetone and received CW

It also provides an audio mixer function so the local sidetone is heard in the headphone / speaker output along with the received CW audio from Mumble.

3.3 Serial I/O port over USB

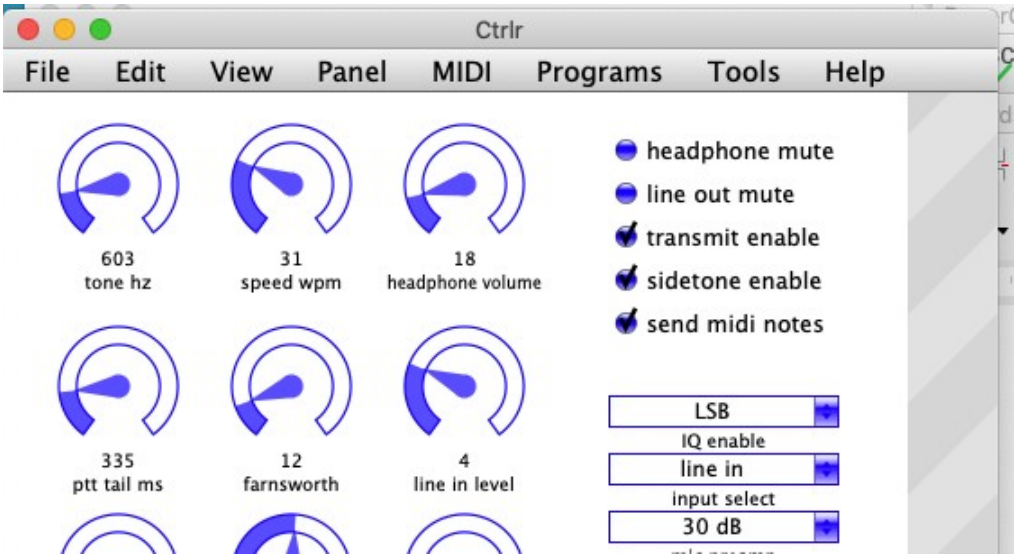
The Hasak software implements a FTDI type interface over the Teensy USB connector as well as the sound card. This allows using Terminal Emulator for both input and output.

4 MIDI Control Interface

I believe the project is targeted towards use with SDR designs, and the many parameters that can be set are done using MIDI commands from a computer.

I think the use of MIDI is only for controller commands, but am not certain because I do not understand a thing about that side of it. I think the high quality CW sidetone is generated using the Teensy DAC (Digital to Analog Converter) resource in the Teensy 4.x family of boards, with the Teensy Audio Library provided by PJRC (Paul Stoffregen). Again, I do not understand much of this. My keyer projects for years have used the analog Hi-Per Mite narrow filter to provide the proper rise and fall time shapes from a square wave input. So the Hasak is a great simplification.

Speed and other parameters would be set from a host computer using a GUI to send MIDI commands to the Hasak. The GUI app is named Ctrlr. The controls look like this:



The last I knew, many of the controls had not yet been implemented in the software, they remain “boiler plate” ready for implementation, so when you use a mouse or other pointer to adjust the headphone volume, the dial moves, and commands are sent to the Teensy, but I do not think they actually have any effect.

5 Standalone implementation with rotary encoder

I made a few changes to the original sketch to allow stand alone operation without the need to run the Ctrlr program on a host computer. I then added sections of code from one of my Teensy projects that allows using a rotary encoder to adjust speed and sidetone frequency. I expect to add a few more parameters eventually.

6 CW Keyboard function added

I added software from my Teensy project to support sending CW with a USB keyboard connected to the Teensy 4.1 header. I currently have included only a minimal set of features from my other project. In addition to sending normal CW, the arrow keys on the keyboard can be used to control both paddle and keyboard CW speeds and sidetone.

The Teensy 4.1 supports a SD card with text files. I have the same set of text files on a card that I use in the eBook CW streams on the Mumble server, but I only support one special function that reads a chunk of text from one file and sends it as CW for testing. I have not yet included backspace for correcting typing errors (I do not type more than a couple of characters ahead). There is a key function to dump the buffer if one is typing far ahead and needs to cancel the output.

7 Alternate Paddle Keyer added

The Hasak project includes a paddle keyer, but I added my own code for my implementation that my fingers are used to. The little bit that I have doen with the Hasak keyer, I think Roger’s implementation is very good. I use two jumpers to connect the paddle to either the Hasak paddle keyer or the W5UXH paddle keyer.

8 OLED display added to project

I also added new code that supports a small OLED display. This is a very minimal display, but it is useful for simple things like displaying speed and sidetone frequency. It also has an alternate mode that displays a very crude scrolling text display of characters sent from paddle and keyboard.

9 Addition of Decoder function

This required a few additional analog components to provide a simple filter and buffer to condition the sidetone output for driving an ADC input pin for the decoder. The decoder is supported in both the Teensy 4.0 and 4.1. I tried briefly to support conditional compile to include or exclude the decoder function but have not succeeded yet. I need to eventually address this. Currently, if the filter / buffer is not installed, things do not work.

The decoder could decode all three streams of CW: paddle, keyboard, serial input (e.g. Mumble). This is because all three streams are present in the audio output that goes to the decoder input pin. But I have the display software written so it ignores the decoder output when the paddle or keyboard are the source of the CW. My paddle and keyboard CW generation software handles the display of characters while the decoder output is inhibited. (I need to come up with a better description of this.)

10 Serial Interface to Terminal Emulator running on the host

I normally use a VT220 Terminal Emulator with my keyers so CW can also be sent by keyboard from the host computer, and all transmitted and received characters can be displayed in the terminal emulator window.

I support the use of the arrow keys to control speed and sidetone, the same as from the USB keyboard connected to the USB host interface on the Teensy.

I use an emulator running of MacOS (named ZOC). But on Windows and Linux there is a free one, PuTTY that can be used. I intend to eventually provide an appendix with the normal PuTTY configuration settings I would use.

11 Operation with Mumble

In early testing it was found that there were periodic glitches in the CW sent over the sound card. This was not heard on the local sidetone. After discussion with Roger (who was aware of the problem) I determined that an update to the Teensy audio library he had released on the GitHub site appears to have fixed the problem.

11.1 Mumble must use TCP, not UDP

Currently, when trying to use the normal low latency UDP setting in Mumble, there are nasty problems with the audio. AA0HW found that using “Force TCP” mode cleans this up.

I would prefer that UDP could be used because of the low latency and the tracking of data packet loss. But the relatively simple “plug and play” nature of the Hasak implementation certainly makes it ideal for newcomers as a fairly easy way to get running on iCW.

I have not had the opportunity to use my implementation in normal iCW QRQ QSOs, but suspect it is going to be usable. AA0HW has reported he thinks it is fine for QSK style QSOs. (Later update: I eventually did test and think this setup works fine for QRQ / QSK operation in TCP mode.)

I have no idea if this is something that can be addressed by Roger et. al. in the future.

11.2 Drive Level into Mumble and local volume levels

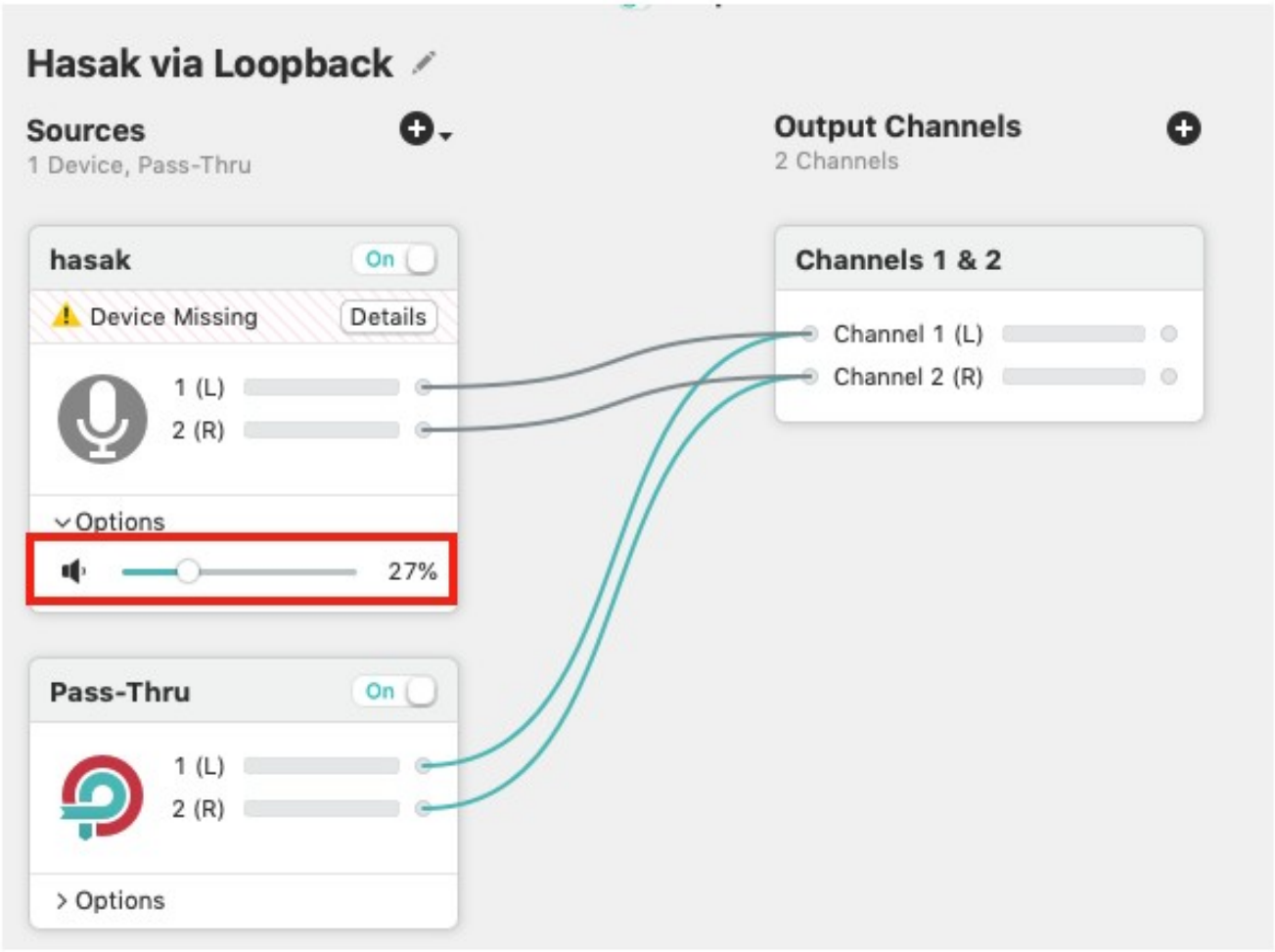
11.2.1 Drive level

Currently, as far as I know, the Hasak software does not yet implement the Ctrlr controls for. If this is the case, and if Roger eventually provides the Ctrlr functions for certain parameters like drive level and audio volume, I should be able to access those in my stand alone version.

As it is, even when the Hasak is overdriving Mumble quite a bit, Mumble handles it very well (much better than some years ago).

On MacOS, I use an app that allows me to create a virtual device that takes the Hasak audio as input and provides a control for the level so I use this to set the level to the nominal level appropriate for driving Mumble.

The app is Loopback Audio and I set the control to 27% for my environment:



11.2.2 Headphone / speaker level

I have found that in my particular hardware implementation the headphone volume is quite reasonable. For a speaker, I use powered speakers with volume controls. AA0HW has reported using non-powered speakers, but I am not sure how that can work from the Teensy output pin.

On 26June2021 I finally had a QSO with Fred using the perfboard prototype. I found that I really want a headphone volume control to control the local

sidetone level. I tried using the Mumble “volume” control for the received volume control and it sounded sort of harsh or distorted when I cut it down.

11.2.2.1 Try external adapter with volume control

So I ordered this item to try:

I know I have something similar but have no idea where it is!

11.2.3 Source for powered speaker

Of course things like this are not always available on Amazon the next time you look, but this is one I am using:

https://www.amazon.com/gp/product/B06XKY1X4M/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1

Wired Speaker, Meetuo Computer Speakers for Desktop, USB Powered Small Speakers with 3.5mm Jack Stereo Sound for Laptop, PC, Desktop, Office(Black)

It cost around $12.

12 Arduino Setup

It has been a few months since I did this originally. For anyone who has some familiarity with Arduino, it should be fairly easy to configure the Arduino environment.

The information from Roger on the GitHub page, along with some notes that I believe were written by Roger to help AA0HW get things running, hopefully will be sufficient. I have those notes in my first project notes document and will paste them below. I am sure I got these from an email from Roger that AA0HW forwarded to me:

1) Download and install arduino-1.8.13 into arduino.
2) Download and install teensyduino-1.53 into arduino.
3) Download github.com/softerhardware/cores into arduino/hardware/teensy/avr/cores
4) Download github.com/softerhardware/Audio into arduino/hardware/teensy/avr/libraries/Audio
5) Download github.com/recri/hasak into hasak
6) Open the Arduino IDE
> 7) Open the hask sketch in hasak/hasak.ino
> 8) Set the Tools > Board to Teensy 4.0
> 9) Set the Tools > USB Type to Serial+MIDI+Audio
> 10) Check that the Tools > Port has found the Teensy 4.0
> 11) Compile and upload the hasak keyer to the Teensy 4.0
> 12) Open the Arduino serial monitor and type s followed by newline.

I the serial monitor says:

```
 sample rate 44100.000000 buffer size 128
active 0 ies/ils/iws 66.0/200.0/466.0 ms
total 2.227% 2.230% isr 0.651% 0.654% buffers 10 14
```

then you have the old cores and Audio, if it says:

```
 sample rate 48000.000000 buffer size 32
active 0 ies/ils/iws 66.0/200.0/466.0 ms
total 3.406% 3.420% isr 0.720% 0.719% buffers 8 10
```

then you have the new cores and Audio.

The two items in red above (cores and audio library) that are downloaded must replace the normal Arduino contents in the two directories specified above:

arduino/hardware/teensy/avr/cores arduino/hardware/teensy/avr/libraries/Audio

It is necessary to locate these two directories, delete the original contents and copy in the contents that were downloaded.

The next item is the actual Arduino sketch to be copied to the sketch directory:

Download github.com/recri/hasak into hasak

The next step is important and easy to overlook:

Set the Tools > USB Type to Serial+MIDI+Audio

On MacOS, the Arduino cores and audio library directories are found in the Arduino.app “show package contents” path (right click on the app and select this from the menu).

12.1 Additional items needed to add my W5UXH features

There are a few libraries required by my software additions. I need to clearly identify and archive them with my project (not done yet).

13 Fastest way to get a Hasak up and running on Mumble

Before I continue with information on my hardware implementation, note that you do not need anything that I have added to the basic Hasak.

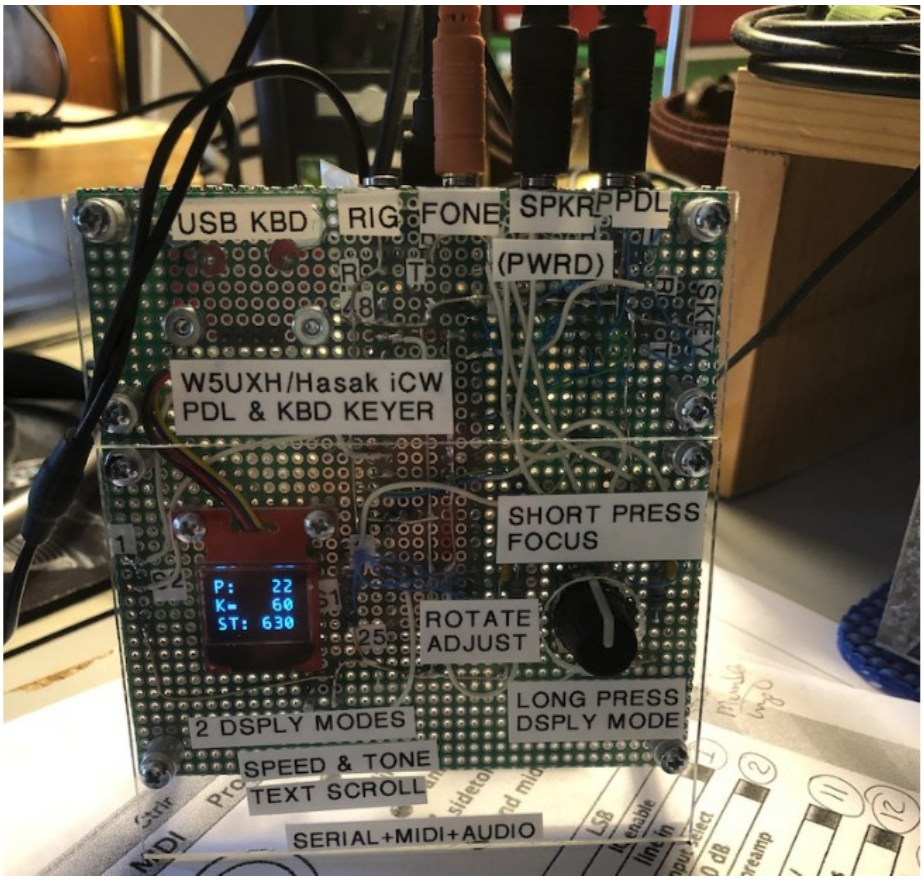
It is very easy to use a solderless breadboard to connect a Teensy 4.0 up to use a paddle to get on iCW. You do need the Ctrlr MIDI app running on a host computer in order to control the keyer.

But I much prefer a standalone keyer as described here. I started with a solderless breadboard for my initial tests, then built a perfboard prototype version, then did the layout for a PCB (discussed below).

14 Early breadboards

After the solderless breadboard, I wired up a perfboard version with the Teensy 4.0. I then moved on to a second perfboard using a Teensy 4.1 because of the USB Host port provided on a header. The Teensy 4.0 supports the USB Host port but it requires making contact to pads on the underside of the board.

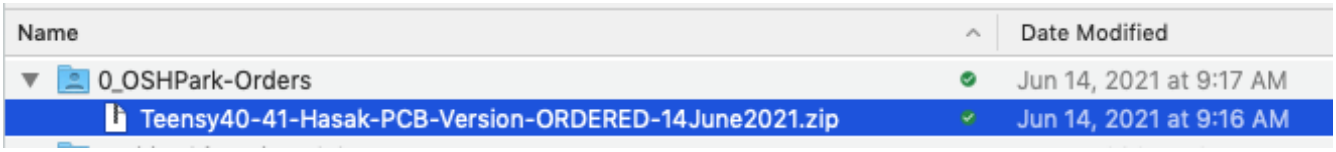
The Teensy 4.0 has 28 pins. The Teensy 4.1 has 48 pins. Currently, my design uses a 48 pin socket but it is compatible with either Teensy. Either Teensy plugs in at the same pin 1 position in the socket. Thus the Arduino sketch can be compiled to target either board. The photo below is the perfboard prototype. The PCB version will physically be nearly identical and "packaged" the same way.



15 Eagle CAD Design

I used Eagle CAD version 7.7.0. The version that was sent to OSHPark for fabrication is archived in this file:

Teensy40-41-Hasak-PCB-Version-ORDERED-14June2021.zip



16 Printed Circuit Board

The first PCB (Version A) is currently being assembled.

17 Update on 16July2021

I found one serious layout mistake that was easily patched with the xacto knive and a wire. I did a Version B layout that will be delivered in a few days. Hopefully I fixed the one serious problem, improved the silkscreen labels, and did not create any new problems.

18 Update on 1Nov2021

I see it has been over three months since I updated these notes. I received the Version B layout boards long ago and built up SN1 on mid July. This was constructed as originally planned with the OLED and rotary encoder on the "back side" of the PCB. All other components are on the other side. But because of the short shaft on the rotary encoder it was necessary for the side with the OLED and encoder to be the "front panel" of the assembled project.

I had picked the particular encoder because it was the one used in the Morserino-32 project I had played with. While assembling SN 1 of this Hasak Version B, I looked for encoders with longer shafts because the

finished project would look better if the front panel side was the same side with the Teensy and other components. I assembled SN 2 with the reversed front panel, so all components are seen in the front view of the assembly. This requires mounting the OLED physically to the back side of the acrylic front panel so the display is closer to the front than if it were mounted on the PCB.

I found Mouser part number 652-PEC12R-4230F-S24, the 30mm shaft length to work out best. In the photo below, SN 2 is on the left.