CHAPTER 12

DSP Audio Mic-Processor

While most modern transceivers have extensive digital signal processing (DSP) functions for microphone audio signal conditioning, older rigs do not. Also, many inexpensive transceivers have no DSP capability. Regardless of their shortcomings, many amateurs still like to use their older (vintage) or less expensive units on a regular basis.

Just like the Chapter 11 DSP Post Processor (DPP), we have created a DSP-based audio microphone signal conditioner based on the Teensy line of microcontrollers. As with the Post Processor, we elected to use the newer Teensy 4.0 unit, primarily for its Audio Library DSP capabilities. The Teensy 4 (to be known as the "T4" from now on) is a real powerhouse, clocked at 600 MHz, with 1024 KB of RAM and 2048 KB of flash memory. (As we go to press, PJRC has announced the Teensy 4.1, which is the same as the T4, but with a higher pin count and larger footprint, and about $5 more expensive.) The T4 initially is being offered in a reduced-size development board package but it has sufficient GPIO pins for our application. Note that the Chapter 11 Post Processor required additional GPIO pins from the underside of the T4 board, while the Mic-Processor does not.

So, what can an audio mic-processor do for you? Some of the possibilities are:

- Automatic speech compressor
- Multi-band equalizer
- FFT display
- Interface both dynamic and capacitor/electret mics to an older transceiver.

There are several commercial units with some of the functions mentioned above, but typically at relatively high prices, some starting at several hundred dollars. The unit we are describing does all of the specs listed for commercial units and costs less than $70, if you build it yourself. In addition, the software, which is the heart of the unit, is open-source, so you can extend and

tailor the processor to your needs. We also present some ideas of ways to augment the features of our DSP Mic-Processor (DMP), if you choose to do so.

User Interface (UI) Overview

The stuff under the hood of our DMP is quite interesting, but everyday interaction with the user interface is what makes the high-powered DSP technology accessible. We have spent a significant amount of time and countless prototypes streamlining the way our DSP unit interfaces with you and the functionality it brings to the table.

A UI has several aspects, most important of which are: 1) the control/data input mechanism, 2) feedback to the user, and 3) the ease with which the user can make changes. We have elected to present data about the state of the processor using a high-resolution TFT color LCD screen. Data input/control uses the touch capability of the TFT screen as well as rotary encoders and an analog-to-digital volume pot. Our experience has shown that this combination of physical controls and TFT touch-screen input is quite efficient and minimizes the number of steps necessary to make changes and set the state of the unit.

Figure 12.1 shows one of the front panel menu options we elected to use, combining a TFT display, volume control, two rotary encoders, and three input/output jacks — there are more jacks on the back panel. With the TFT display, the user gets graphical feedback as to the state of the unit's functions and can select functions and set parameters. A real-time FFT display shows the spectral content of the various signals, after DSP. Because of the less stringent graphics requirements, this project uses a slightly lower resolution touch screen (320×240 pixels) than was used in the DPP.

Figure 12.1 — DSP Audio Mic-Processor.

Figure 12.2 shows a close-up of the TFT screen with the graphical display graphic equalizer function shown. Here the user can select bands and band levels, while monitoring the effect using the headphones. Real-time interaction makes fine tuning the unit a breeze. The UI is explained in detail later in this chapter.

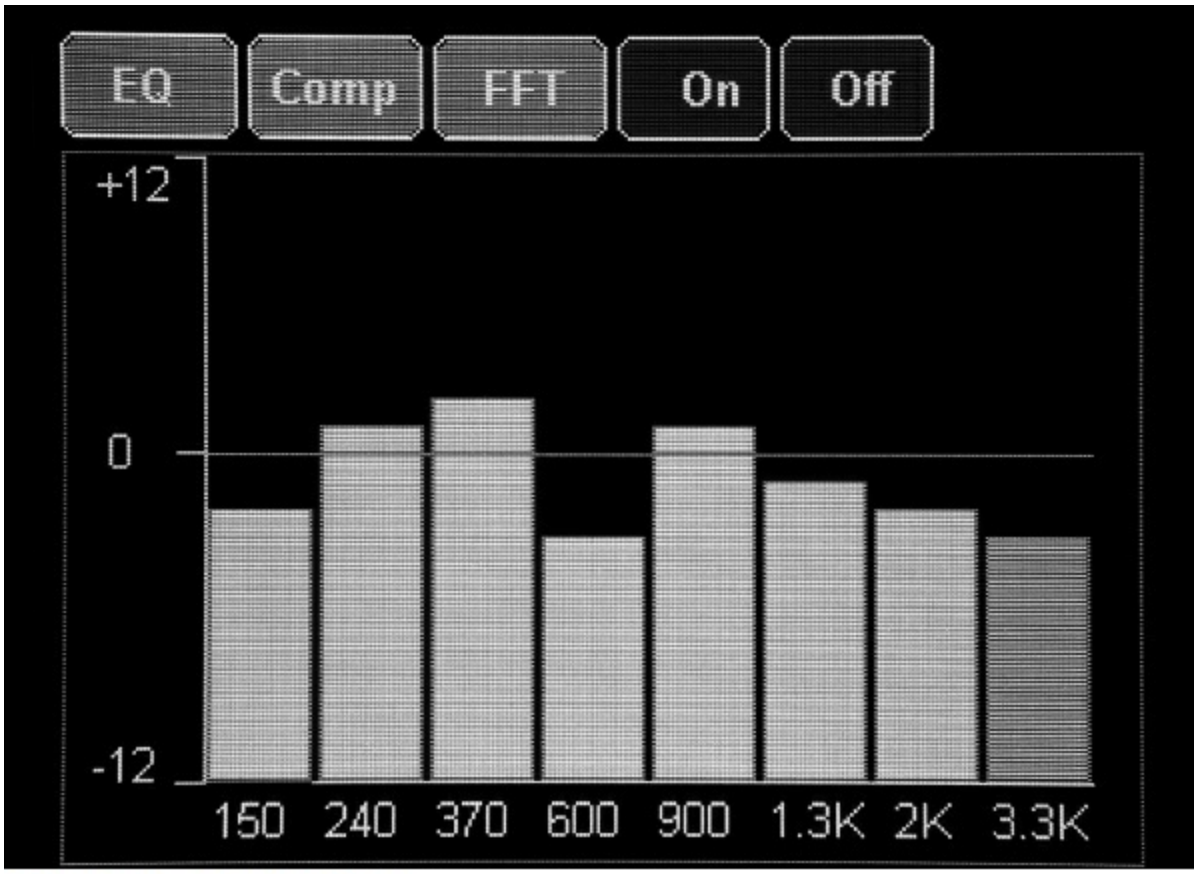

Figure 12.2 — User interface.

Circuit Description

This project is about 25% electronic hardware and 75% software, so let's get the circuit out of the way first and then dive into the software.

Figure 12.3 shows a block diagram of the circuit. The heart of the unit is the Teensy T4 microcontroller. The second most important component is the Teensy Audio Adapter (TAA), which contains all of the audio I/O, such as ADCs and DACs necessary to capture the audio signals. It also sends the digital data to the T4 and converts and outputs the processed signals back to analog audio again. The TAA has 16-bit conversion and operates at a 96 kHz conversion rate, enough for full 16-bit 48 kHz CD-quality audio. The TAA can directly piggyback onto the T4, too, significantly simplifying the circuit wiring. Finally, the T4 has enough horsepower to run a real-time FFT processing at the same time as it is doing the DSP filtering on the audio channels. Note that there is a specific version of the TAA for the T4. The older TAA module does not work correctly with the T4.

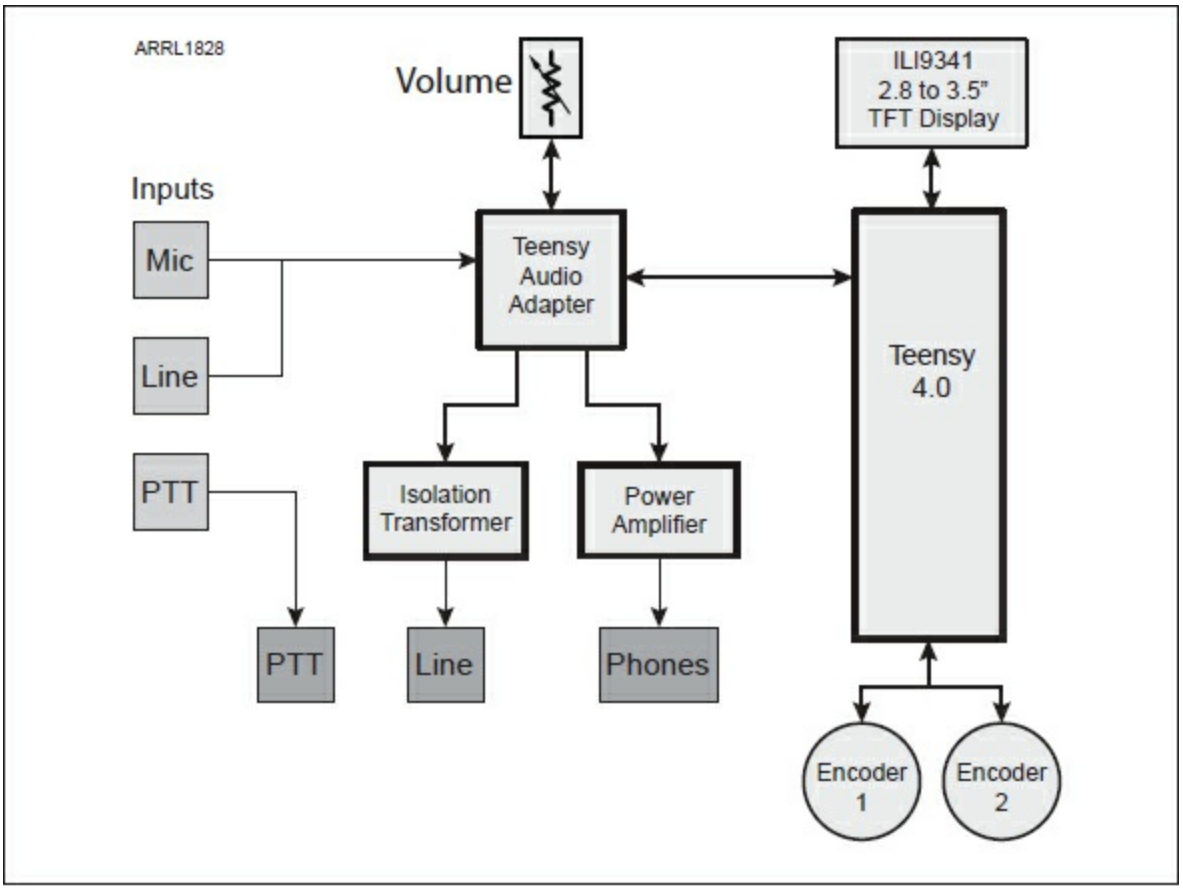

Figure 12.3 — DSP Audio Mic-Processor (DMP) block diagram.

Starting at the input, we see that provision can be made for two inputs which are to be routed to the computational channels. At this time, only the left channel is utilized. However, it would be very easy to add the DSP functions to the right channel if you wish. We did not need two independent DSP channels, so adding a second channel remains as an exercise for the reader. The microphone is connected to the TAA Mic input, which has extra gain to accommodate the low-level signals from the microphone. The Line input operates at line levels and is suitable for high-level line signals. This allows you to input signals from an audio mixer source that outputs line-level signals. Selection of the source must be done in the setup portion of the Comp Screen.

Signals from the TAA "headphone" output go to an isolation transformer. The isolation transformer eliminates the possibility of ground loop noise and isolates the grounds of the transceiver and Audio Mic-Processor.

The low-level mic inputs accommodate either dynamic, capacitor, or

electret mics. Capacitor and electret mics require a “bias” dc voltage, which is provided by the TAA. A Mic Gain setting allows the specific level characteristics of your mic to be accommodated.

The outputs include a line-level output as well as an amplified headphone output for monitoring. This amplified output is capable driving a small speaker as well, depending on the application.

The detailed circuit diagram, including pin assignments is contained in **Figure 12.4** and **Table 12.1**. Note that some Teensy pins, such as USB, are not available because of system use. One voltage regulator provides power for the Teensy, display LED, and power amplifier. The 5 V fixed regulator drops the 12 V to the T4 Vin pin. 3.3 V is provided from the T4 on-board regulator.

Table 12.1
Teensy 4 Pin Assignments

Teensy 4 Pin	*Attachment*	*Teensy 4 Pin*	*Attachment*
2	Touch IRQ	17	Encoder2 A
3	Encoder1 A	18	Audio Adapter
4	Encoder1 B	19	Audio Adapter
5	Encoder1 SW	20	Audio
6	N/C	21	Audio
7	Audio	22	
8	Audio	23	Audio Adapter
9	LCD D/C	24	
10	LCD CS	25	
11	LCD, Touch MOSI	26	
12	Touch MISO	27	
13	LCD, Touch SCK	28	
14	Encoder2 SW	29	
15	Volume Pot	30	
16	Encoder2 B	31	

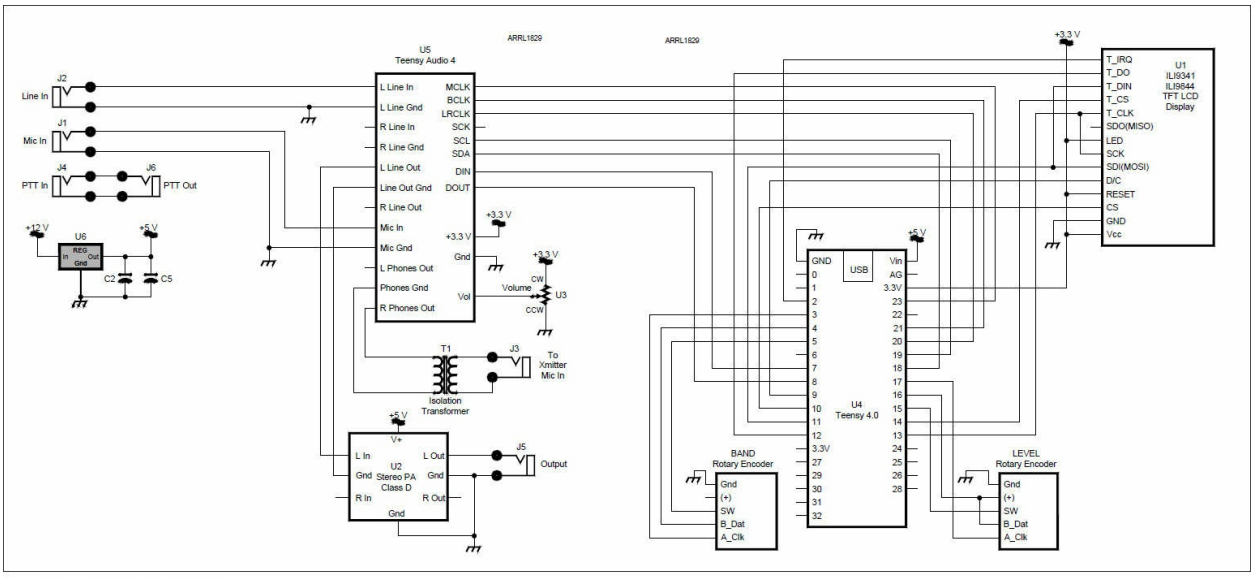

Figure 12.4 — Circuit diagram for DMP.

Building the DMP

One advantage of building circuits for the audio frequency range is the relaxed requirements on layout. The DMP can be constructed using any of the popular methods. We built our prototype on a modified breakout board. The Audio Adapter plugs directly onto the Teensy, so control wiring is minimal. We always use sockets for the microprocessors for ease of insertion and removal, if necessary. The other components are connected using IDC connectors and ribbon cable for convenience.

No special wiring precautions are required. We like to use #30 AWG solid hookup wire because of the close 0.1-inch hole spacing in the proto-board (same for most perf boards.) The main power connections use #24 AWG or larger wire to minimize resistive losses.

Figure 12.5 shows the top of the main circuit board, with the Teensy and Audio Adapter installed. Note the use of multi-pin IDC connectors for display, front panel connections, and encoders. **Figure 12.6** shows a view of the bottom of the circuit board. **Figure 12.7** shows the front and back panel wiring, with the ribbon cables and female IDC connectors.

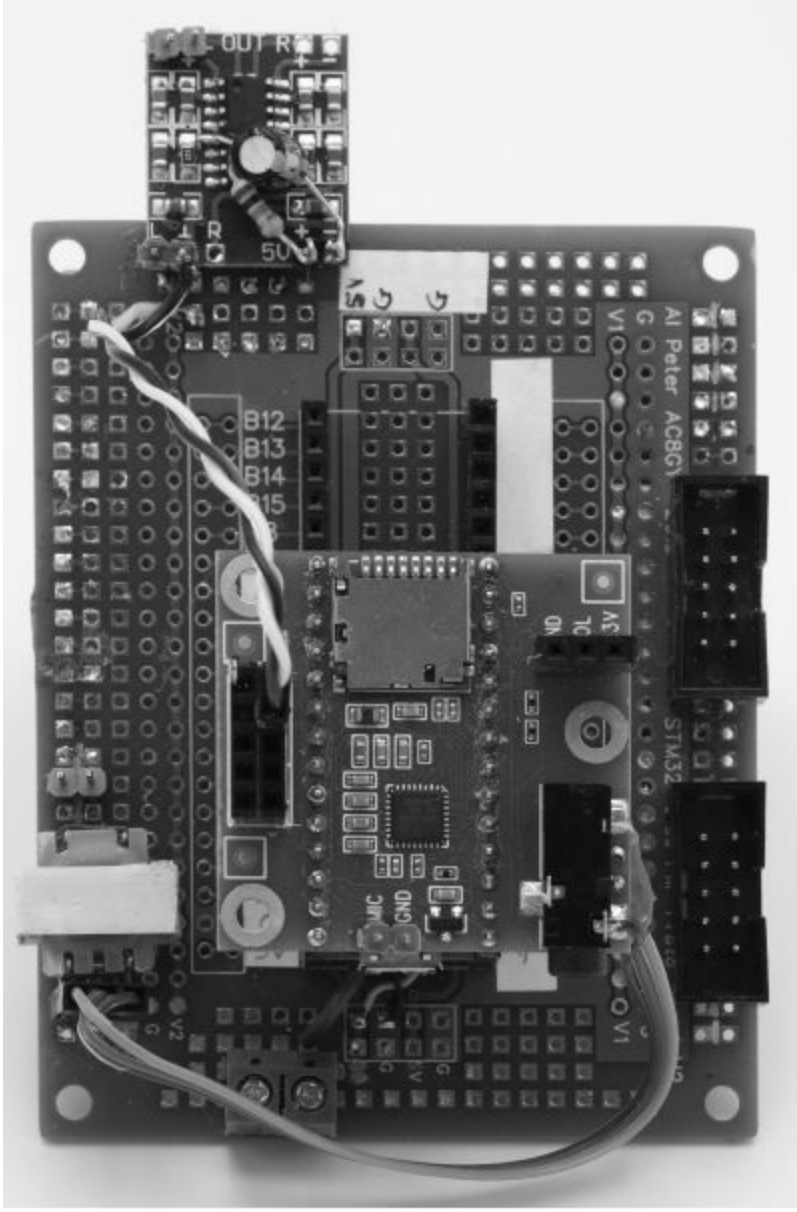

Figure 12.5 — DMP circuit board (top).

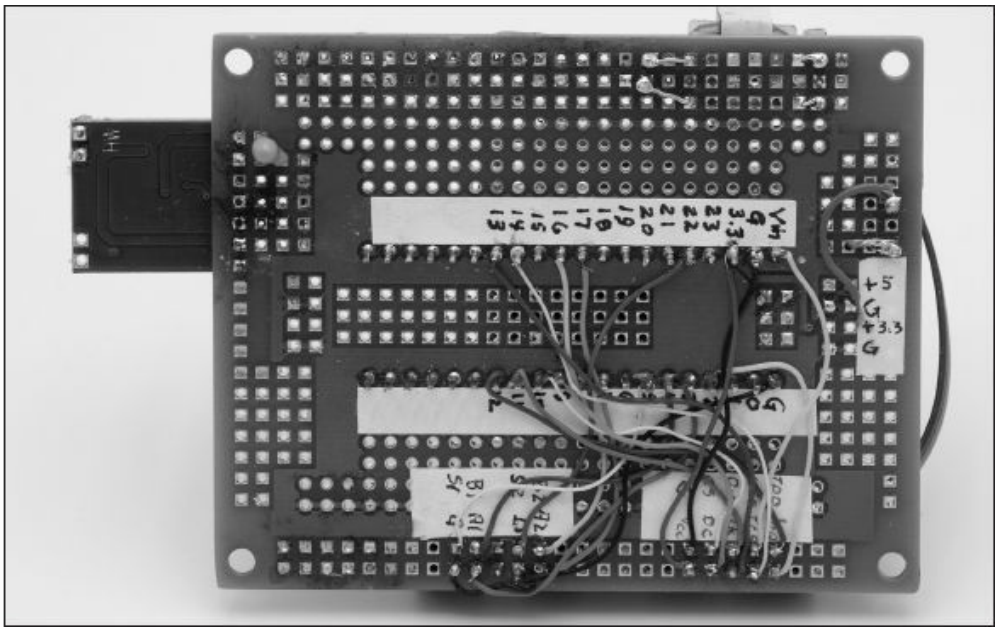

Figure 12.6 — Bottom view of circuit board.

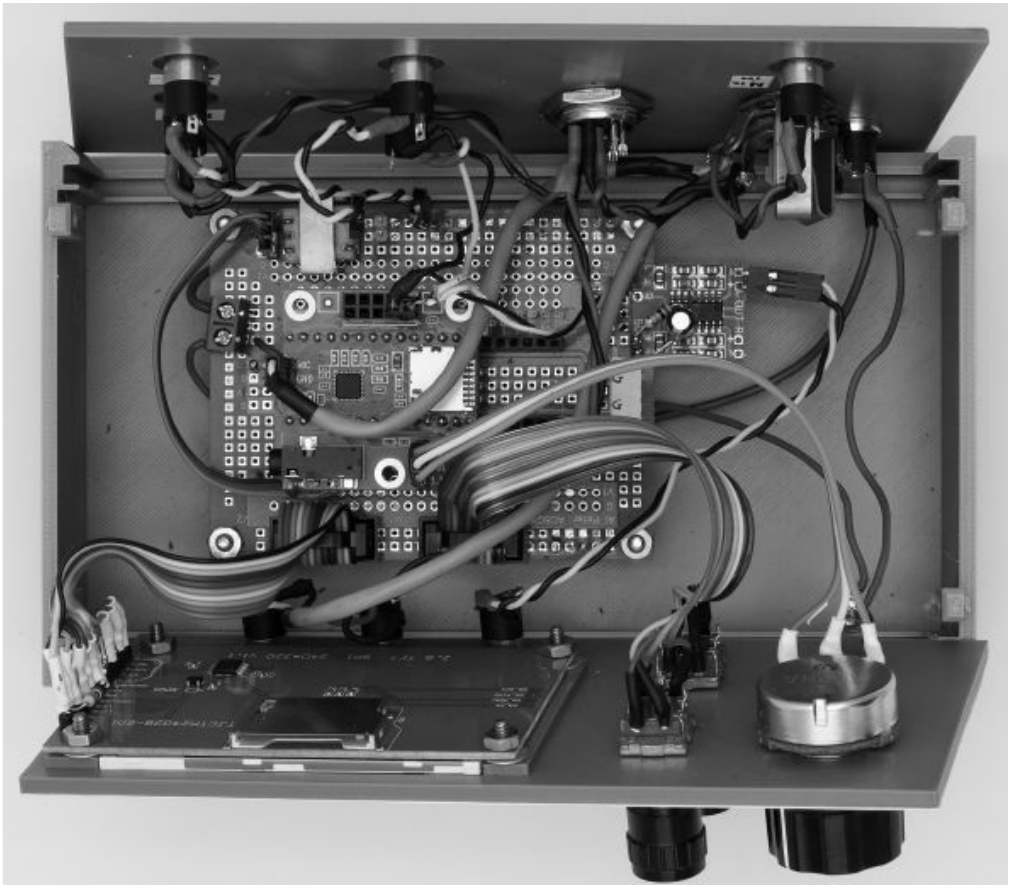

Figure 12.7 — Front and back panel wiring.

Finally, a top view of the assembled unit is shown in **Figure 12.8**. Shielded cables are used to carry the low-level audio signals from the front panel to the Audio Adapter and from the inputs.

Figure 12.8 — Top view of complete unit.

For most of the projects in this book we have assumed that the user has ample 12 V or 13.8 V power available from rig power supplies. This project doesn't require much power and could be battery operated in a pinch. If an adequate supply is not available, a small switch-mode 12 V supply capable of a couple of amps works just fine, or you can build the Utility Power Supply described in Chapter 5. Be sure to add adequate filtering of the incoming 12 V supply to minimize noise from getting into the audio path.

We made a 3D printed case with a custom faceplate. **Figure 12.9** shows the unit with the top lid off and **Figure 12.10** shows the back panel.

Figure 12.8 — Top view of complete unit.

Figure 12.10 — The DMP back panel.

Operating the Mic-Processor

Most of the difficult parts of using the processor are in the setup and, as we indicated previously, we spent a lot of time trying to make the user interaction as easy and intuitive as possible. The operational steps are a simple 1-2-3 process:

1) After hooking up the DMP to your rig and mic, you must set the parameters for the functions you want to use. To enter the “Setup” mode for any function, touch the on-screen function button to bring up the setup

routine for that function. Using the two encoders, set the parameters for that function. (A detailed description of each setup is given below.) If displayed, touch the "Set" button to store the parameters in (emulated) EEPROM.

2) Turn each function on or off using the on-screen buttons at the top of the display. The status of the function is displayed by the color of the button for that function.

3) Adjust the output level using the volume control. You are now ready to go!

Function Detail

There are only three functions available for the DMP

1) Equalizer (EQ)

The EQ utilizes eight bands: 150, 240, 370, 590, 900, 1300, 2000, and 3300 Hz. **Figure 12.11** shows the individual band responses.

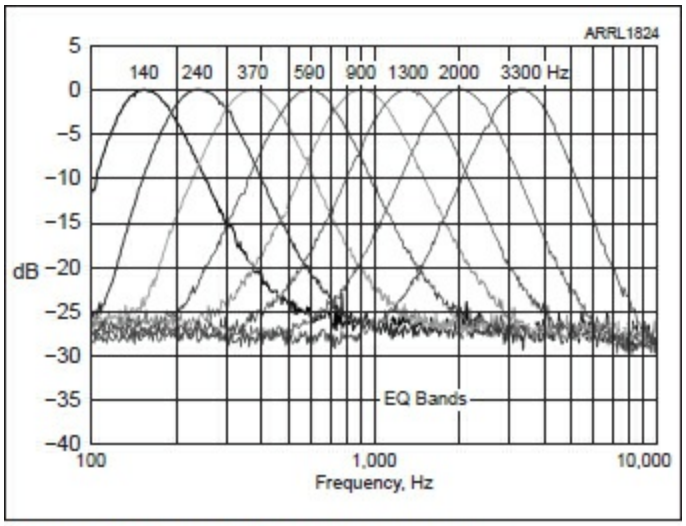

Figure 12.11 — EQ band responses.

In the EQ Setting screen, the bands are graphically depicted on the screen, with two parameter selections, as shown in **Figure 12.12**. The *Band/Adjust* encoder selects the band and the *Level* encoder sets the level for that band. Each band has a range of +12 dB to –12 dB. Note that there is interaction between adjacent bands, since the individual bands have finite skirt slopes and the filters overlap. In practice this does not affect the performance, because the amount of adjustment to achieve "natural" sound is typically not

large.

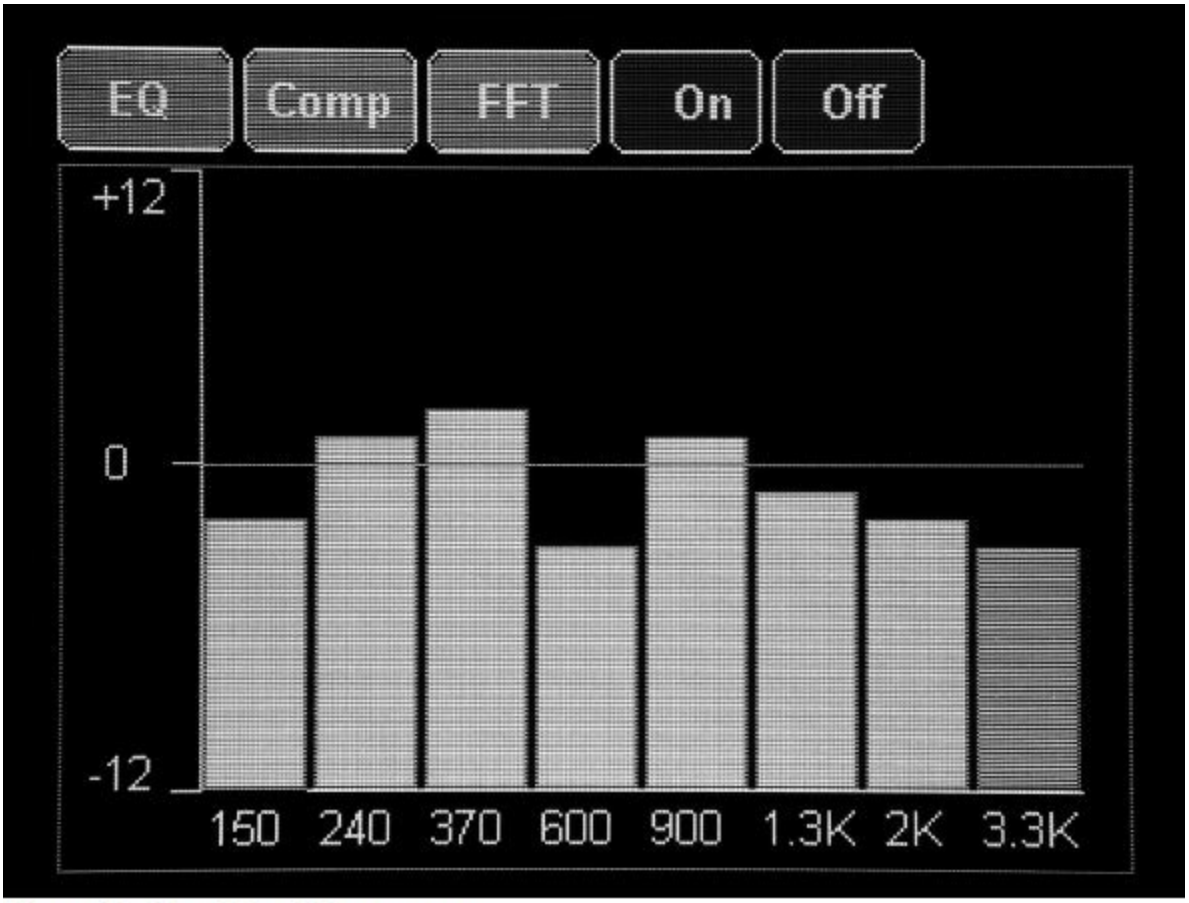

Figure 12.12 — EQ setting screen.

The best way to set the EQ profile is to monitor your speech while adjusting the bands and levels by altering the band levels to achieve the most "natural" sound. The EQ may be used in conjunction with any of the other functions. We suggest recording your speech and listening to it later. This eliminates the aural feedback present when you speak. Just hook an audio recorder to the headphone output as you speak.

A typical response plot of the EQ is shown in **Figure 12.13**.

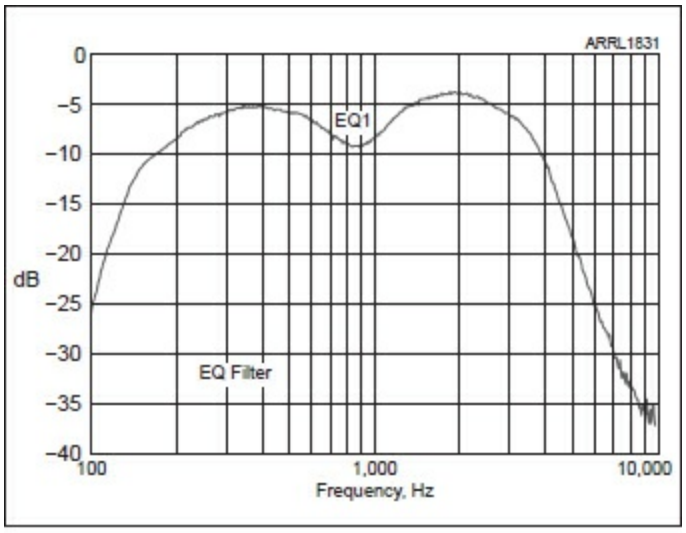

Figure 12.13 — Typical EQ frequency response.

2) Automatic Level Control (ALC) — Compressor

Speech audio levels vary considerably during the typical QSO. Perception of the strength of your SSB signal depends on having the modulating level more nearly uniform than is usual with typical speech. The ALC function here augments the perceived levels by boosting low volume levels and limiting the maximum volume levels. Both the upper Threshold level, at which the auto level control limits the top volume and the lower Threshold level, at which the gain is increased, are adjustable using the first encoder. Simply touch the button for the Threshold parameter to set and turn the encoder. When the level is where you want it, press the “Set” button to save the values in EEPROM.

Mic Gain may also be set on this screen, to account for differences in microphone sensitivity. **Figure 12.14** shows the Comp setting screen.

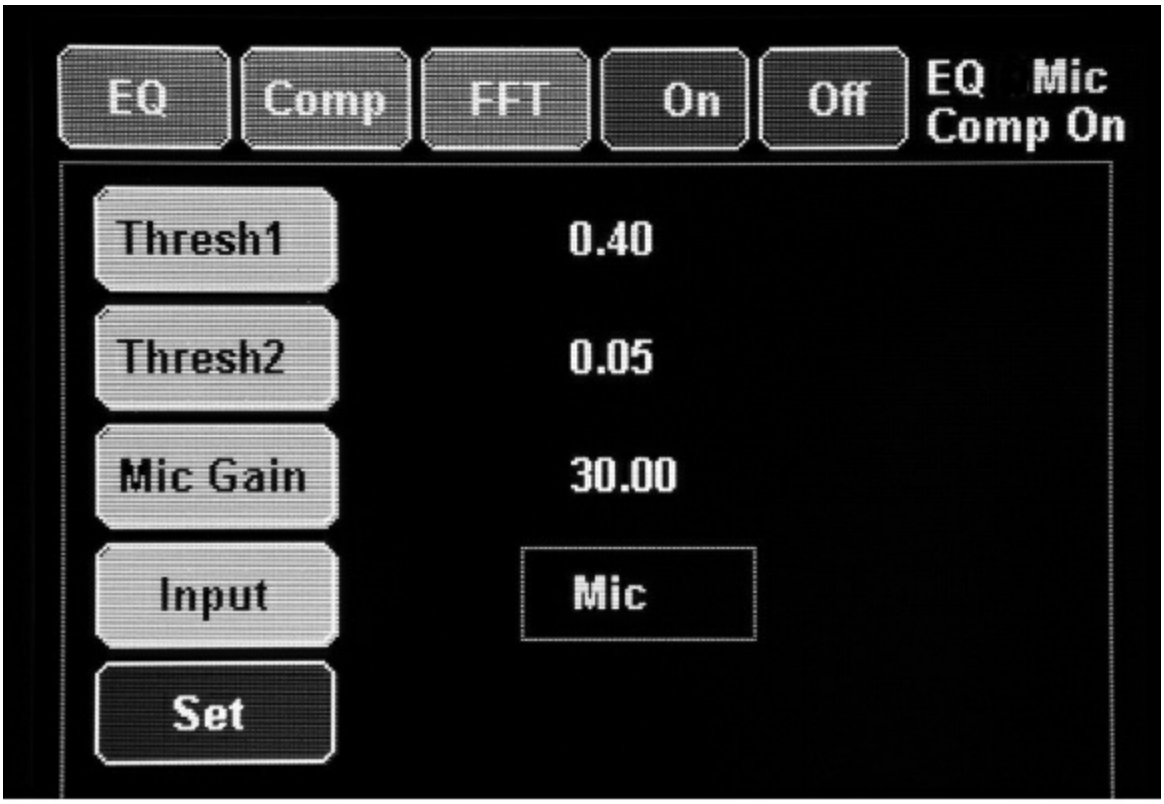

Figure 12.14 — Comp Parameter setting screen.

3) FFT

A Fast Fourier Transform (FFT) display is available during operation, showing the frequency spectrum from 100 Hz to 5 kHz. 128 frequency bands clearly show the frequency content of the audio. The display is rapidly refreshed for a real-time depiction of the spectrum.

To activate the FFT, press the FFT button. FFT is available during operation, but not while changing settings. All other functions may be used with FFT, so the effect of EQ and Compression can be observed. **Figure 12.15** shows a typical FFT spectrum. The vertical axis is a log scale in dB.

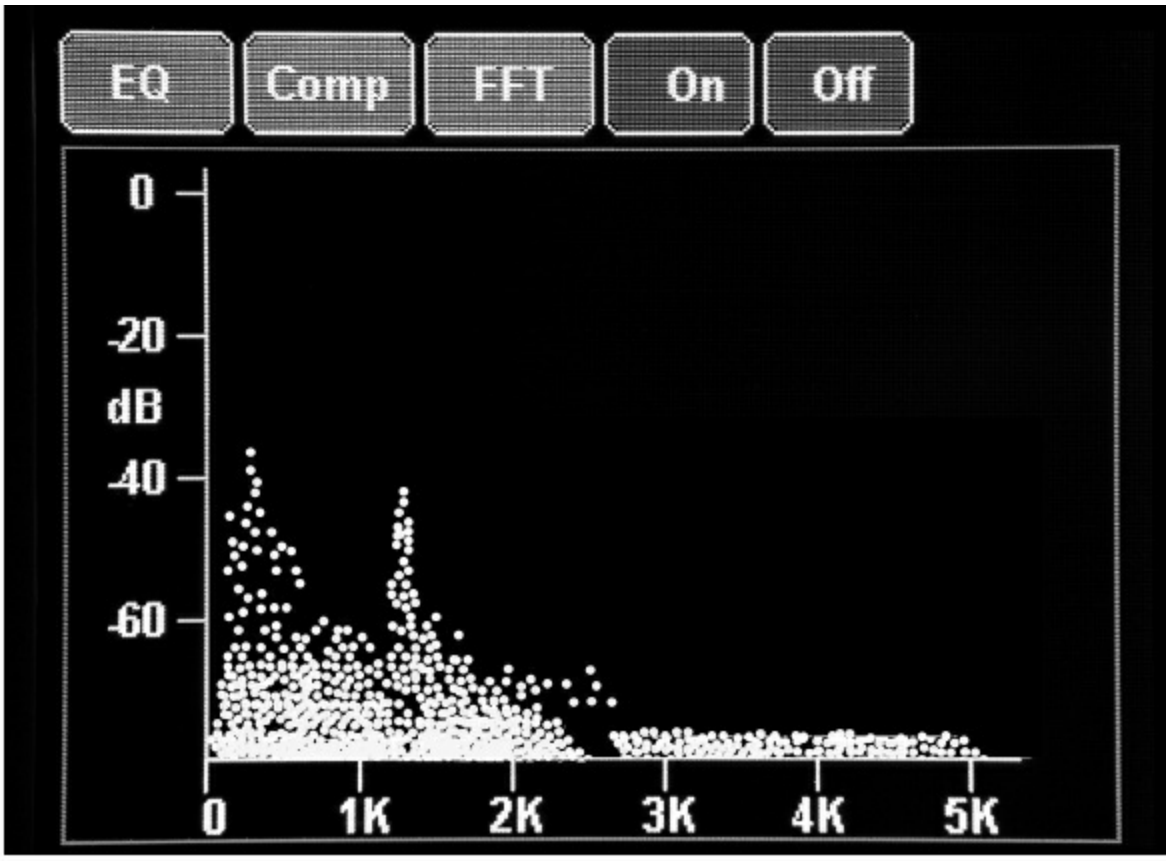

Figure 12.15 — FFT display.

Software Code Description

Much of the DMP code is like that of the Post Processor, and a general introduction to digital signal processing (DSP) was presented in Chapter 11. Again, the Teensy Audio library is used to implement DSP functions. The library functions consist of a number of modules that are connected to form a signal path. Each module also has a set of individual functions with parameters. The software design starts by using an online graphical design tool to lay out the data flow for the audio DSP and then copying the automatically generated code to the Arduino IDE. The graphical design tool is available from **www.pjrc.com/teensy/gui/**. (The design software is very slick, but takes some time to master. It's worth the effort if you want to extend our DMP.)

Function parameters are set with the code to tailor the DSP responses to suit. The flow diagram is shown in **Figure 12.16**. The DSP Mic-Processor performs a number of functions to modify and improve the audio signals

going to your transmitter. Some of these functions are similar to circuit functions in the analog world, some are not. In any case, all of our processing is done digitally by performing mathematical operations on the audio signals to alter frequency response, change levels and reduce noise. The speed of the T4 is sufficient to do all of this seamlessly (which is why we selected the more expensive T4).

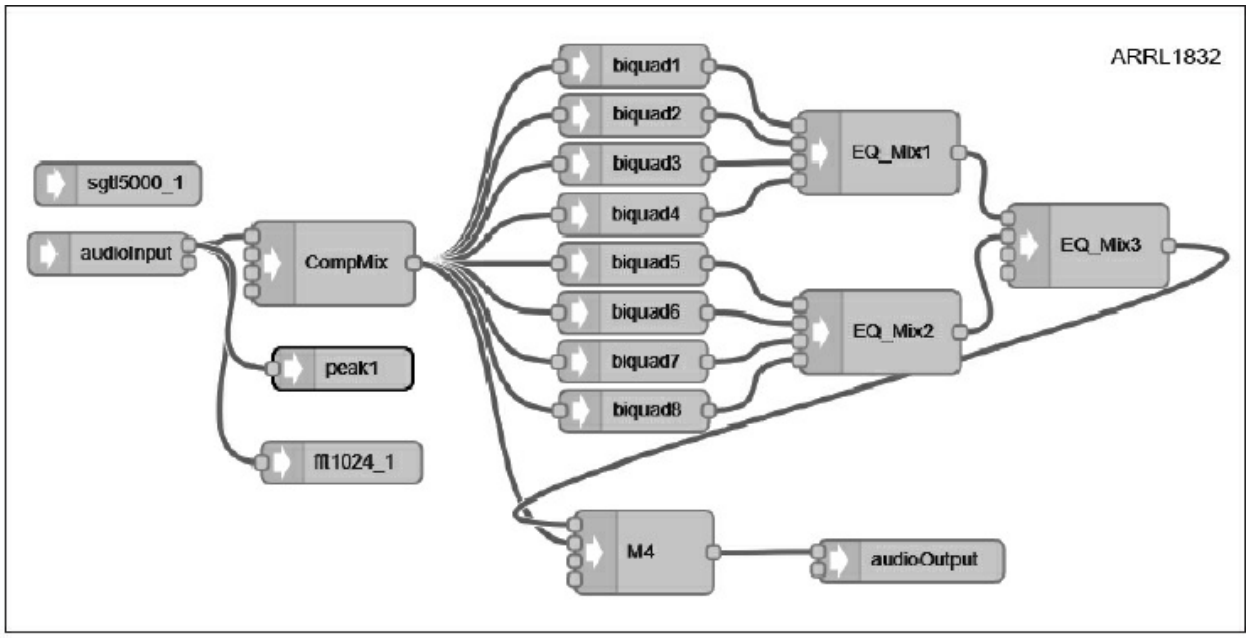

Figure 12.16 — Single channel DSP signal flow diagram.

There are analog circuit equivalents to some of the frequency domain filters we use. For instance, the low-pass filters we used in the Post-Processor are equivalent to a cascade of RC (resistor-capacitor) filters with gain. Each single analog RC stage yields a frequency roll-off of –3 dB/octave. Our eight cascaded digital filters have a total roll-off of 48 dB/octave or 160 dB/decade, which is the equivalent of 16 stages of RC filters! Imagine wiring up such an array, with appropriate gain stages. The band-pass filters used in the DMP are ever more complex to implement in hardware. We do it all in software.

The Compressor or Automatic Level Control is the equivalent of a variable gain stage, controlled by the peak signal level — again a complex circuit. We do all of that with a few lines of code and the very nice audio library that works with the Teensy's Audio Adapter.

The filters for the EQ we use are called biquad or biquadradic cascaded filters, which are a variety of recursive linear filters based on a mathematical

formula that has feedback incorporated. The interested the reader can find out more at **peabody.sapp.org/class/350.838/lab/biquad/**. In essence, the filters we use have a grouping of four filters in a module that can be cascaded with additional modules to obtain better performance. The biquad filters can be configured as low-pass (LP), high-pass (HP), notch, or band-pass (BP) filters. We use the band-pass configuration in our DMP.

In addition to digital filters, we also use mathematical operations to calculate Fast Fourier Transforms (FFT) to give us real-time spectrum plots of our audio signals. Once again, the T4 is fast enough to calculate all of the filter function operations and do the FFT at the same time.

Signal Flow

At the input in Figure 12.16, an audio capture module is the starting place for the audio signal. This connects the audio input to the DSP functions. Each module is interconnected using a (graphical) digital "connector" wire.

Starting at the left in Figure 12.16, we see the CompMix module, which provides the variable gain necessary for the Compressor. The gain is set in the *Compressor()* routine, which samples the input signal peak values and adjusts the gain according to the sampled level. The response is fast enough to react to changing levels but averages the control signal sufficiently to ensure natural sounding speech. The speech levels are sampled using the "Peak" function.

Next in line we have the eight-band equalizer. Signals are sent to eight different biquad band-pass filters (biquad1-biquad8) each with a different center frequency. Human hearing is approximately logarithmic with frequency, so the center frequency spacing is set to be approximately 1.5× the previous band frequency, which gives an equal spacing on a log scale. The eight bands cover the range of 150 Hz to 3300 Hz in eight steps. The output of each filter is sent to the inputs of two mixers, in which the individual band gains are set. The eight outputs are then combined into one composite signal by another mixer. The spectrum is shaped by setting the individual band gains from +12 dB to –12 dB. Because the filter skirts drop off at a finite value, there is interaction among the filters, limiting the sharpness of peaks and dips. The purpose of the EQ section is to gently shape the spectrum to eliminate humps or dips in the ultimate frequency spectrum, compensating

for microphone variations. Note that these filters create phase changes, so the mixer gains alternate + and – to account for the appropriate phase reversals.

In the software, there are other code routines that perform various functions. For instance, there is an ongoing routine that reads the on-screen touch-activated buttons to determine when one was pressed. Each Setting function has three major routines:

1) Draw the graphical representation on the screen;

2) Read the encoders and set the appropriate biquad filter parameters and/or the mixer gains; and then

3) Turn the functions on or off using the front panel push buttons.

In the code there are also the usual menu functions, encoder interrupt functions, screen management, and so on, all of which has been adequately described elsewhere in the previous chapters. The code is broken up into nine different files for convenience of both coding and debugging. Adequate comments throughout should help anyone wishing to make extensions or improvements.

Possible extensions for the adventurous include:

- Independent functions for Channel A and Channel B. This is primarily a UI coding exercise, adding setting and selection routines for each channel. This would allow for more than one mic to be attached and compensated separately, or a high-level mixer input could be added to the second channel.
- A digital recorder could be incorporated to allow the operator to review speech quality off-line. The Teensy Audio Library has the necessary components to implement such a recorder.
- Adding real-time spectrum monitoring during setup in order to observe the spectral response variations while making changes.

Conclusion

In summary, the DSP Mic-Processor presented here can bring an older transceiver up to modern signal processing standards and could even be a valuable adjunct to lower-end current rigs that are perfectly adequate in all other ways. If you do enhance the DMP, we hope you'll post your work on the SoftwareControlledHamRadio website!