# Making a Sofware Defined Radio for the QRP Enthusiast: Part 1

Ward Harriman—AE6TY ae6ty@arrl.net

I am a relative newcomer to the world of amateur radio. I first became interested several years ago while casting about for a hobby to enjoy in my retirement. In my explorations, I found myself repeatedly drawn to various radio related topics and kept coming across publications by the ARRL. After repeated exposure, I found myself studying up for my license and even learning Code. I was hooked. Even more unexpected, I started to be intrigued with building my own rig. The primary reason for this desire was simple, I wanted to be able to say "rig here is homebrew" during my QSOs. And "homebrew" to me meant homebrew design, homebrew assembly, homebrew programming, homebrew in a wide range of disciplines both familiar and untried. AND: the rig had to work well enough to actually MAKE QSOs.

This article will describe the resulting rig, but even more importantly it will also describe some of the skills I had to pick up along the way and the reasoning I used to make design decisions. It is my hope that those readers who are newcomers to the homebrew art, or at least not practiced aficionados, will be encouraged to expand their skills and take on some new adventures in homebrewing. For those interested in replicating all or part of this project, the QRP ARCI website will have more detailed construction information to include schematics, software listings, and PC board designs.

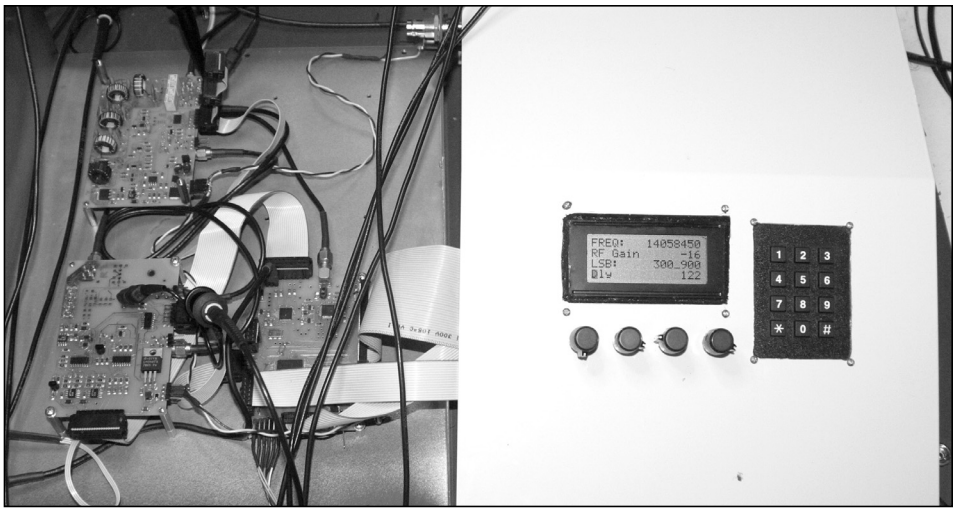


**Figure 1—The SDR Transceiver, with case in the open position.**

## The Quandary

There were two areas in amateur radio which intrigued me: QRP and Software Defined Radios. In many ways, these two facets of radio are polar opposites.

At one extreme is QRP. Now strictly speaking, QRP means reduced transmit power but in fact the QRP community is also concerned with homebrewing, low power consumption and (often) portable operations. QRP also often implies a certain frugality: doing more with less. One popular QRP radio, the PIXIE II has only two transistors and a single audio amplifier IC! With this discovery, the idea of designing and building my own rig started stirring in the back of my mind.

At the other extreme is Software Defined Radios. The most common implementation of an SDR starts with a small piece of hardware usually described as a down converter which translates the RF signal "down" to audio frequency signals. Once this conversion is done, the majority of radio functionality can be implemented in software running on a personal computer. These down converters can be quite simple; a state of the art down converter requires no more than a few ICs and a handful of discrete components. Most of the rest of an SDR usually resides in a personal computer, which is both the good news and the bad news so far as QRP is concerned. The good news is that Software Defined Radio has become quite inexpensive because it leverages the consumer Personal Computer market. It uses inexpensive processors and high quality sound cards to provide nearly all the hardware necessary to implement a Software Defined Radio. But while providing extremely high performance radios, the PC-based SDR has some significant shortcomings. Even the most frugal laptop PC costs hundreds of dollars, weighs in at several pounds and uses tens of watts. Further, the typical laptop PC is not tolerant of dirt and moisture and is difficult to operate in sunlight. For the QRP enthusiast who embraces low power consumption and portable operation, today's Software Defined Radio has some very serious drawbacks.

Hence, an impasse: how can one explore the world of Software Define Radio within the value structure of the QRP community? It became clear that the major obstacle to QRP style SDR was the presence of a full-blown PC with its attendant disk, LCD screen and full sized keyboard. The goal, then, was to provide a low cost alternative to the PC which was at the heart of most SDR projects.

## The Solution

It became clear to me that in order to "do it myself" I would have to get rid of the PC and replace it with a small, low power digital signal processor (DSP). The DSP would be programmed largely from scratch. For example, there would be no need for a "Signal Processing Library"; the goal was to write one. Further, there would be no need for an Operating System; the processor was to be dedicated to the application. There would be no file system needed and there were no standard peripherals to be controlled.

Although I have been involved with electronics for quite some time, this project was really my first run at radio frequency design. Therefore, a primary consideration was to provide a learning platform for both RF design and for SDR technology. Thus, modularity was a primary goal; changing one portion of the circuitry should not require replacement of unrelated portions. Also, it was important to provide easy access to the circuits themselves for prob-

ing and rework. Multiband operation was desirable but not an overriding concern. Since this was a "Software Defined Radio" project, software was to be the largest fraction of effort and so providing a good software development environment was paramount; a high level language "Integrated Development Environment" was essential.

There were, as usual, several "non-goals" for this version of the transceiver. It was not a goal to minimize power consumption. It was not a goal to provide portable operation. It was not a goal to maximize RF performance; for example, no shielding was to be provided. It was not a goal to minimize circuitry or cost. All of these non-goals could be realized in later versions of the transceiver, incorporating the lessons learned from this first exercise.

Figure 1 shows the rig prepared to provide the three basic functions: on the air operation, circuitry debug and software development. In normal operation, the cabinet is assembled and only the display and controls shown on the face of the cabinet are visible. However, the transceiver will operate as shown. Generally the enclosure is buttoned up even when doing software development, mainly to free up bench space.

As can be seen in Figure 1, beauty in packaging was not a primary goal. Leaving lots of room to allow easy access to the various boards was a primary requirement. Indeed, the entire rig can be removed from the enclosure and placed directly on the workbench. When working on a new board, the board can be placed on the bench and stitched in using long cables. Circuit debug almost never takes place inside the enclosure. Originally it was assumed that a backplane of some type would be used to connect and support the boards. This may prove desirable in the future but for now simple ribbon cables work well and provide more flexibility.

In operation, the rig is controlled by the four multifunction knobs and the numeric keypad on the front panel. The leftmost knob controls the parameter listed on the top line of the display. The second knob from the left controls the parameter listed on the second line of the display, and so forth. Control parameters are chosen from a menu by scrolling through the menu using the keypad. Pushing the 4 button moves up the menu, pushing the 6 button moves down. By hitting the # key, the keypad can be used to enter a parameter value directly. This is particularly useful when entering frequency. Parameters and functions that can be controlled from the front panel include:

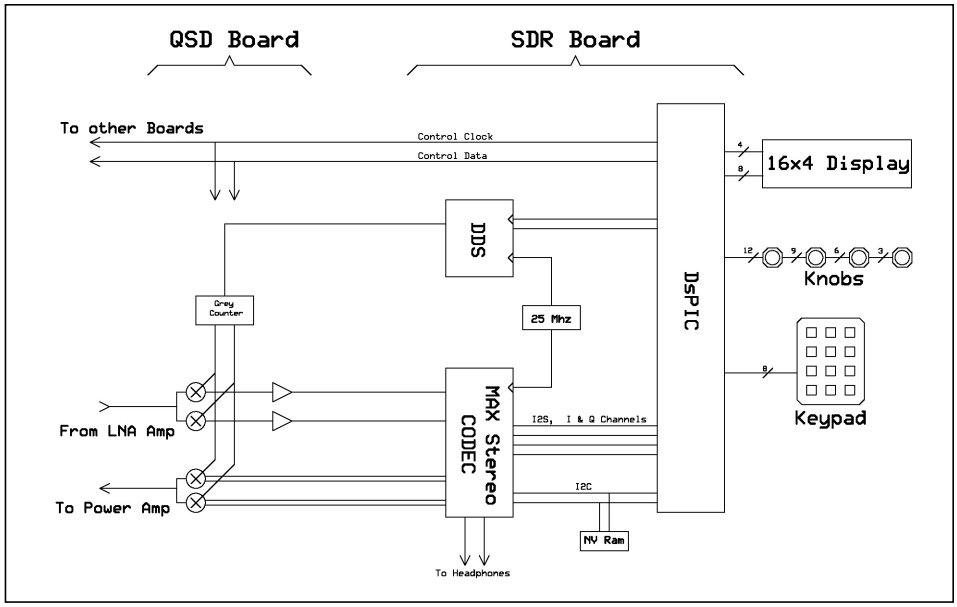


**Figure 2—Hardware block diagram for the transceiver.**

a. Frequency
b. RF Gain (gain between QSD and the A/D converter inside the CODEC)
c. AF Gain (gain from D/A of CODEC to the headphones)
d. Mode (USB, LSB, BOTH, Binaural)
e. Receive bandwidth
f. Receive Bandwidth Filter Length
g. Delay (delay data to right ear by this amount)
h. Sidetone Volume
i. Sidetone frequency
j. Words Per Minute (CW Transmit speed)
k. Analog front end gain adjust
l. Analog front end phase adjust
m. RF Receive Level (monitor only, essentially an S meter)
n. Interrupt instruction count (# instructions executed for each CODEC sample)

The list of parameters is continuously in flux as features are added or removed. Because of this, it is essential that any SDR project provide a mechanism to easily add and control parameters.

Figure 2 shows a hardware block diagram for the transceiver. Signals come into the transceiver through a band-specific low noise amplifier (not shown), through the receive quadrature sampling detector (QSD) and into the CODEC (coder/decoder) where they are digitized and sent to the embedded microprocessor. Likewise, transmitted signals go through the CODEC where they are converted to analog format and sent to the QSD board for up-conversion to the transmit frequency. From there, signals go to a broadband driver amplifier (not shown) located on the QSD board and on to a band-specific 5 watt Class E power amplifier. The band-specific components for each band, namely the low noise receiving amplifier and power amplifier, are located on single boards that can be stacked on top of each other. Switching controlled by the microprocessor selects the appropriate board. In Figure 1, a single band-specific board appears in the upper left corner. At present, boards have been constructed for 40, 30 and 20 meters.

**Making Choices**

Probably the most fundamental choice one makes in designing a radio is the basic architecture. Does one build a superhet? How about a Direct Conversion? Does one use phasing techniques or narrow filters for sideband rejection? These and many other questions continued to circle the subconscious for several months. In the end, the decision was made to use phasing techniques to create I and Q signals at baseband. The I/Q signals would be digitized by an A-D converter and processed by the processor. The processor would provide

filtering and side channel suppression and deliver an audio signal to the user.

At the heart of the SDR is the processor itself. The most significant criterion for choosing a processor was that it be mountable by a hobbyist with moderate skill and determination. Hence, ball grid arrays and even packages numbering hundreds of pins would present problems. Second, the chosen processor must require modest initial investment and be easy to program using a high level language. An in-circuit debugger was mandatory. Third, the processor would need to be low power; a power budget on the order of a watt seemed reasonable. Finally, it would need to connect to the A/D and D/A converters of choice as well as switches and displays.

Ultimately, the processor chosen was the dsPIC33F256 from Microchip. In many ways, the dsPIC product line is underpowered compared to modern DSP processors from vendors like Analog Devices and Texas Instruments. Still, the dsPIC has much to offer: it is low cost, low power, and comes in convenient packages. The development environment (called an Integrated Development Environment or IDE) runs on a PC and is the right price (free). The single upfront cost for using the dsPIC line was a small investment in an "In-Circuit Debugger." This will be discussed later in the sections covering software development.

Once the processor had been chosen the next step was to identify a viable A/D and D/A converter, or CODEC. Because the dsPIC is a 16 bit processor, a 16 bit CODEC seemed to be the best choice. It is well known that more bits in the CODEC translates directly to better dynamic range but it was decided that 16 bits would probably be adequate for this project.

The range of CODECs is just as large as the range of DSPs. Ultimately, the MAXIM MAX9851 stereo CODEC was chosen. A block diagram of this part is shown in Figure 4. The target market for this CODEC is cell phones and MP3 players. The only significant drawback of this device was that the packaging was a little scary. Still, what project is complete without at least a little fear?

Having chosen a processor and CODEC it was time to choose the implementation of the down converter. Fortunately for the newcomer, there are many fewer choices available. This area was completely unfamiliar, so exploratory prototyping was done. Using ugly construction, several different mixers including a multiplexer based commutating mixer, a high level diode mixer (TUF3) and a FET based H-mode mixer were prototyped. Ultimately, a variation of the Quadrature Sampling Detector was chosen. This circuit has been popularized by Dan Tayloe (N7VE) and is the basis of the popular "Softrock" series of down converters. There are numerous examples of this circuit in the public domain and so this circuit is not discussed further here.

Once a down converter was chosen it was necessary to choose a VFO technology. After a variety of experiments, the Analog Devices AD9951 Direct Digital Synthesis chip was chosen. This circuitry for this subsystem was lifted almost exactly from the application notes provided by Analog Devices and is not discussed further here.

**Getting Started on Hardware Design**

A great deal of construction in this wonderful hobby can be done using "ugly" construction. Indeed, the author did a great deal of prototyping of the various circuits from EMRFD. However, it was all too clear that the dsPIC processor and the MAX9851 CODEC would require the development of a printed circuit (PC) board. Further, the number and spacing of pins on the IC's to be used implied that any of the homebrew techniques for making PC boards would not be adequate. After spending several evenings in ugly construction, this author has come to consider the cost of a PC board a good investment. The level of frustration is much reduced and the pride in craftsmanship is much enhanced. Thus, it was decided to use PC boards for all the circuitry.

A little research on the web led to the discovery of several companies offering custom PC board fabrication. During this research a few essential features were identified. The first essential feature was the ability to manufacture standard industry boards: .064 inch thick, fiberglass, plated through holes, minimal etch width of 6 to 7 mils (thousands of an inch), 1/2-ounce copper and tinned. The second essential feature was that the PC board vendor supply or directly support a complete tool suite. The tool suite should provide three basic functions integrated into a single, seamless design and fabrication process. These are "schematic capture," "pc board layout" and "fabrication" including ordering, payment and delivery.

The schematic capture tools must provide a library of common components and the ability to add user defined components. Further, it must support multi-page schematics and provide minimal "design rules checking" to help ensure a well-defined design. Finally, the schematic capture tool should provide a parts list and a net list for use by the printed circuit board layout tools.

The printed circuit board tools should use the netlist provided by the schematic capture system to help ensure proper connectivity during the pc board layout process. Navigation around the design (panning, zooming, selection, searching, library commands, etc) should have the same "look and feel" as the schematic capture software. There should be no need to learn a new set of commands for these basic operations.

The pc board layout tools should provide for "power planes." When a signal is run on a power plane, the tools should provide spacing around the signal automatically. Additionally, the tools should provide thermal relief of pins connected to a power plane. Support for arbitrary shapes of power planes is highly desirable.

As with the schematic capture tools, the p.c. layout tools should provide an extensive library and allow the designer to easily add components as desired. Building a new component should be straightforward and take only a few minutes to create a typical footprint.

The p.c. layout tools should support common layout techniques such as "rats nests" and component rotation. It must be easy to lay down and modify etch. Changing etch width should be simple and moving an etch from one layer to another should result in automatic via generation.

Once the design is complete, the ordering and fabrication should be a simple matter of submitting a design file and paying with a credit card. The design should be conveyed through the internet. The vendor should enforce any "design rules" (board size, etch widths, hole sizes, spacing, etc) that are necessary to ensure proper fabrication. There should not be the possibility that a submitted design is "unmanufacturable."

In the end, several companies met all the above requirements. Ultimately, a company called "ExpressPCB" was chosen, in part because of their "miniboard" service. A miniboard is a 2.5 × 3.8 inch, two-layer board made of FR4 fiberglass. It allows etches down to 6 mils and via holes down to .014 inches. A maximum hole count of 350 is enforced. This miniboard service does not provide a silkscreen or solder mask; things often considered luxuries by the homebrew community. Once a design is complete, ExpressPCB can fabricate and deliver 3 copies of the board for $51 plus shipping. More information can be found at www.ExpressPCB.com.

This author was familiar with the principles of p.c. board layout and so it took only a couple of evenings to be comfortable with the tool suite. Of course, as with any tool, the more it is used the more productive one becomes. Complete mastery probably takes three or four projects. Those unfamiliar with p.c. layout techniques can often locate an introductory course at a nearby community college or hobby center.

When a p.c. board is designed the result is called "artwork" and for good reason; the design of a p.c. board is indeed a work of art. Through experience, each designer develops a "style" that often becomes almost a religion. For this author, all designs have several things in common:

- All designs use the same board outline and mounting hole placement.
- All designs are two layers and have the bottom layer as ground.
- All bypass capacitors are on the bottom layer
- All non-RF connectors are .025 inch posts on .1 inch centers
- All power entering the board is polarity protected with diodes.
- All connectors are through hole to provide strain relief.
- All ICs are surface mount whenever possible.
- All passive components (except bypass caps) are SMD-1206 whenever possible.
- All traces are run on the top layer except when absolutely necessary (the ground plane is kept as intact as possible).
- All power is 'locally' regulated.
- Signal etches are as wide as possible and spaced as far apart as possible.
- All designs have multiple "ground posts"; places to connect a ground clip.
- All through holes are larger than absolutely necessary as this simplifies rework considerably.

There is much to be said for making the design "pretty," for "pretty" often translates to "working." Take advantage of the flexible pinouts of modern ICs; choose convenient pinouts to simplify PC board layout. Spending time on the layout allows the subconscious to think about the design and find problems. It is very common to discover a circuit design flaw during layout.

**The First Subsystem**

Having chosen to use printed circuit boards and chosen a tool suite, the actual design and construction of the various modules could begin. The first project was the QSD down converter board. It is simpler than the others and less dense. The schematic was drawn up and checked several times. Then, the two layer PC board was laid out. The ExpressPCB tools provide a way to check your schematics against the actual PC board layout. After confirming the actual layout matched the schematic, the design was almost ready for fabrication; "almost" because there is one more crucial step…

Once the p.c. board is designed, one should print out all board layers at a 1/1 scale. Since the board is two layers, the top layer is printed in red on an overhead transparency and the bottom layer is printed in green on ordinary paper. This approach allows both layers to be examined at the same time. Using the printed etch as a template, each and every component is placed on the transparency to make sure the leads were properly spaced and no component interfered with any others. Alternatively, a part can be laid on its back and the transparency laid on top. This approach was used for the CODEC chip.

Even though the design was extremely simple, several parts were moved to simplify assembly and testing. After doing this mechanical check several times, the design stabilized and it was submitted to ExpressPCB for fabrication. Three days later the board arrived and, as usual, the fabrication was flawless. Assembly was uneventful.

As an aside, some of the extra room on the down converter board was used to place a footprint for the CODEC chip which was to be used on the SDR digital board. This allowed me to practice mounting the CODEC.

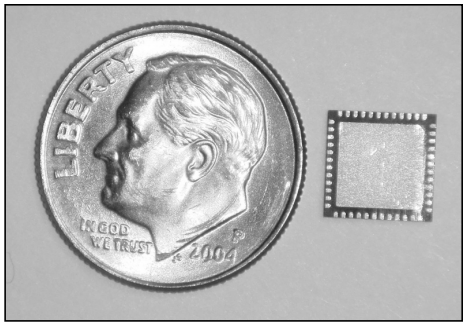


**Figure 3—Bottom side of the CODEC.**

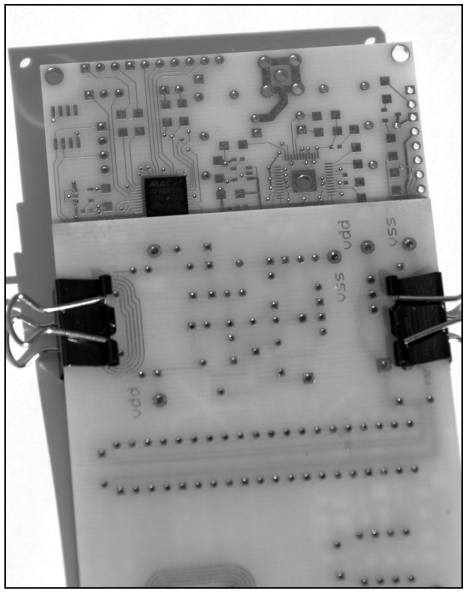


**Figure 4—Holding the CODEC in place.**

**Surface Mount Components**

Having completed the down converter section of the SDR it was time to start working on the unique part of this SDR project: the digital module. The first step in this effort was to ensure that it was possible to mount the CODEC onto a p.c. board of home design using home methods. Figure 3 shows the bottom side of the CODEC. It is very difficult to see the pads involved much less believe one can actually solder them to a p.c. board using homebrew techniques. So, one of the spare QSD boards was sacrificed to experimentation.

A great deal of experimentation followed but the final approach turned out to be simplicity itself. After a little introspection, it was realized that the biggest problem was getting the chip aligned and held in place long enough to solder it down.