



President's Corner

By Steven Bible, N7HPR, President, TAPR

This year looks to be another great one for TAPR as the organization continues to fulfill its goal as the leader on the cutting edge of Amateur Radio.

HPSDR keeps right on cooking with new projects in the works straight from the HPSDR community of volunteers to you via the facilities of TAPR. For example, the beginning of the year saw the release of Metis --- the new HPSDR high-speed interface.

Meanwhile, there are also some new timing hardware projects in the planning stages at TAPR. So, keep an eye on www.tapr.org for announcements regarding the availability of new projects on both the timing and HPSDR fronts.

TAPR's plans for the Hamvention in May are now set in stone and it should be another important long weekend in Dayton. TAPR was instrumental in getting David Rowe, VK5DGR, to travel to the States to speak at the TAPR Hamvention Forum and the TAPR-AMSAT Hamvention Banquet. David is on the forefront of the CODEC2 project and is also very involved in the Village Telco project. He will be speaking on those topics at the Forum and Banquet, respectfully, and will be available at the TAPR booth the rest of the weekend along with the rest of the TAPR crew.



Joining David at the Forum will be Phil Harman, VK6APH, who will talk about the HPSDR projects, and yours truly, who will talk about the world of TAPR. Hamvention 2011 promises to be a must-see event so don't miss it.

It is also not too early to start thinking about the TAPR & ARRL Digital Communications Conference (DCC). This year, the DCC will be in Baltimore, MD, near the Baltimore Washington International Airport (BWI). Our host hotel is the Four Points by Sheraton at the airport.

The deadline for papers at the DCC is months away (July 31 to be exact), but it is not too early to begin composing your thoughts for the presentations you would like to make at the conference. Any topic dealing with cutting edge communication technology is fair game for a presentation; so if you are involved in such a project, please consider presenting in Baltimore in September.

Finally, we hope you will like the new look for PSR. The editorial staff worked long and hard on updating the newsletter layout and hope that you will be pleased with the changes.

Until next time, 73,
Steve, N7HPR

###

President's Corner	01
SDR Cube Transceiver	02
TAPR at Hamvention	15
2011 TAPR/ARRL DCC	16
A Volunteer	17
Metis	17
AVRS - Automatic Voice Relay System	18
Write Here !	21
AVRS Java Parsers	21
The Fine Print	22

SDR Cube Transceiver

By George Heron N2APB and Juha Niinikoski, OH2NLT

Software Defined Radio without a PC ... how can that be?!
With an embedded DSP and a connection to your favorite Softrock, you can have a rig that is small enough for portable use with enough of the bells & whistles of SDR, and you “don’t need no stinkin’ PC!”



The SDR Cube Transceiver. Housed in a 4" x 4" x 4" black powder-coated aluminum enclosure, this self-contained Software Defined Radio couples with a Softrock RF front end to perform as a full-featured SSB and CW transceiver for the HF bands. The Cube design contains a full user interface, DSP processing, I/O connectors on the rear panel, and 27 in³ shielded space internally for a Softrock. The Cube ideally interfaces to the NUE-PSK modem for digital mode support, and future growth will directly encompass digital modem functions and more. What a great way to eliminate the complexities and vagaries of using the PC as a radio!

Much has been written about Software Defined Radio in the last couple of years, as it is an exciting leading-edge technology for us experimenters in Amateur Radio. New companies and groups are providing new and improved hardware and solutions seemingly all the time, and an entire quadrature-based sampling “ecosystem” has appeared for USB sound cards, RF front ends in every flavor, and powerful and innovative software to process the digital samples being produced hundreds of thousands of times per second.

But there’s something they all have in common ... a personal computer (PC) is required to perform the heavy processing and provide the stunningly beautiful, multi-faceted and eminently capable graphical user interface! Nearly everybody’s got a PC, so why not take advantage of the powerful Intel/whatever processors in them, the large colorful displays, and the wealth of software libraries, drivers, APIs and other standards that already are mature?

Ahhh, but therein lies the problem. At least for us two designers, and for what we think are many other hams, there are three basic and fundamental problems in the “PC approach” to SDR ...

1) **You can’t (easily) take it with you when you’re gone** – There is a growing class of ops today wishing “go portable” with their rigs: sit on a mountaintop, in a field, at the beach, along a forest trail, Field Day, on a business trip, or just sit out at the pool in the back yard while catching a few Q’s. PC laptop screens are usually fairly difficult to see even in moderate daylight, and who wants to lug around even a small expensive netbook (that is usually underpowered for the conventional SDR task) in a backpack? Laptop batteries go quickly, you need a flat surface for the mouse or be wrist-cramped while heavily using a touchpad – yuck!

2) **You can’t get the dang thing all connected right** – Have you tried getting a computer connected and configured for SDR recently? Heck, just getting a new printer working on a Windows is sometimes a challenge, what with all the different operation system versions, driver compatibility, vendor compatibility, etc. Toss in sound card sampling rates, dynamic range needs,

Linux and Windows, OS patches, changing versions of the SDR apps ... and you have the average ham totally wrapped around the axle. There's no doubt the computer literate among us can handle these things, but if we ever want to expand our markets (i.e., products) to the larger ham population, this would not be the way to do it.

3) **A mouse and keyboard does not a radio maketh** -- As hams, we inherently have a visceral affinity to knobs, meters, switches, pots and other standard front panel controls that have been built into our radios over the years. Sure we are technologists too, and many of us are pushing toward newer radio designs that incorporate microcontrollers and computer interfaces. But at the end of the day, many ops feel that it just does not "feel right" when controlling a transceiver with a PC screen and a mouse, whether it is late at night during a contest or while ragchewing with friends in a regular sked.

So What's A Ham To Do?

Go embedded, lad, go embedded! This need to capture the advanced capabilities of software defined radio, coupled with the need to keep it looking and feeling "like a radio" in the field, formed the starting point for our project: the SDR Cube Transceiver. We wanted to benefit from the flexibility that SDR offers with its innate ability to handle virtually any operating mode – SSB, AM, CW, and digital – with just a new software file or program loaded into the radio. Yet we still wanted conventional knobs for controls and compact packaging. In a recent posting on the SDR-Cube e-mail list, one fellow stated, "We don't drive software defined cars or operate software defined microwave ovens. We use cars and ovens. So why must we have software and computers so much in our face when dealing with SDR in our hobby?" We agree completely!

Our intention here is not to provide yet another detailed tutorial on the design methodology and DSP coding techniques used to create the SDR Cube. There are many other good works in *QST*, *RadCom* and in some dedicated SIGs doing an excellent job at that and can serve as a good reference for those wishing to learn embedded programming or strike out in their own directions. We will

instead review the fundamental components that make up the Cube and explain the challenges at each of the design stages and the reasoning for our choices.

In the end, we hope that readers will better understand the task of constructing such an embedded DSP controller, and coupling it with a quadrature sampling RF front end like any of the popular Softrocks so they might attempt to replicate the design. We provide numerous paths that others can follow to easily hop onto the SDR bandwagon and end up with a very cool transceiver that is unlike anything else around. And most importantly, this SDR will have a conventional look and feel with knobs, switches and compact packaging. After all, our informal tag line for the SDR Cube is ... "We don't need no stinkin' PC!"

SDR Cube Overview

The SDR Cube is a QRP transceiver consisting of an embedded DSP controller coupled with a Softrock as the RF deck. A PC is not needed with this SDR because all the signal processing is accomplished "in the box". Sized at 4" x 4" x 4", the Cube contains a full complement of built-in user interface: graphic LCD for spectrum display, typical controls for frequency, mode and signal management, and I/O connectors for connection to the outside world. The Cube design is optimized to internally accommodate the popular Softrock RXTX 6.3 electronics. Alternatively, a connector on the rear panel allows the SDR Cube to connect with virtually any other quadrature sampling RF front end that uses the standard I/Q baseband audio signals. Different from other experimenter single board solutions, the Cube was designed from the start to serve as a complete and standalone transceiver – the power and flexibility that software defined radio brings, with the convenience of a full user interface with standard (physical) radio controls that most operators prefer, and the small size to allow easy portability.

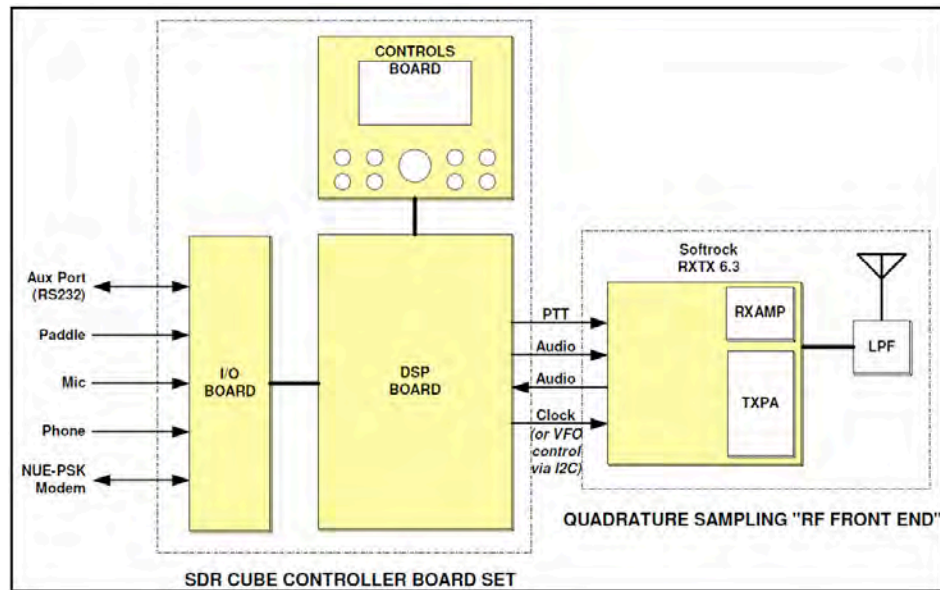


Figure 1: Block Diagram of the SDR Cube Transceiver

The SDR Cube controller consists of three interconnecting pc boards that serve to interface with the user and provide the signal processing necessary for processing the in-phase and quadrature (I and Q) audio signals to/from the RF front end. The I/O board provides for the connection of the usual radio peripherals – mic, paddle headphones – and an RS-232 port for enabling the Cube’s software to be easily updated by the user when needed.

Digital Mode Support ... While the initial release of the SDR Cube software provides support for SSB and CW modes, it relies on the use of an external modem for support of the digital modes. One can use an application on a PC connected to the audio output of the SDR Cube Transceiver, much as it would connect to any other SSB-capable rig. However, consistent with the

design goals of portability and PC-less operation, we also provided a special, optimized interface for connection to the NUE-PSK Digital Modem for the processing of digital mode communications. Further, while the initial release of the Cube software handles this interface to the modem in a standard way with analog audio signals, the next version will bypass the redundant conversion to/from analog audio and pass digital audio between the Cube and NUE-PSK modem. This optimized interface offers improved SNR benefits that are already seen on the bench prototypes. In follow-on versions of the SDR Cube software we will integrate the actual modem source code in order to natively support the digital modes in the SDR Cube without the need for any external digital modem.

The Softrock “RF Front End” ... The SDR Cube is designed to interface to any quadrature-based RF front end that provides I/Q baseband audio signals. We use the terms “RF front end” and “RF deck” to mean the electronics that perform the mixing, amplification and filtering of RF signals for an HF radio. The most popular and prolific RF deck around is the Softrock family of small and inexpensive kits. Some 11,000 of these boards in different flavors have been sold around the world already, and each depends on a PC for signal processing and user interface. So by designing the SDR Cube to easily interface with this huge installed base of Softrocks, we provide a way for every Softrock owner to decouple from the complexities of using the PC as a radio, while simultaneously tapping into a ready-made market without needing first to design the RF deck ourselves. A slight downside of this approach is that the simplified designs of the Softrock naturally offer some design compromises that result in lower RF performance – sensitivity, clock feed-thru, unwanted mixing by-products, etc – which may in turn be perceived as SDR Cube limitations. However there are other higher-ended I/Q-based products that also work in a stellar fashion with the Cube ... products such as the Genesis Radio SDR family, the original FlexRadio SDR-1000 electronics and the FA-SDR from Funk Amateur magazine. And who knows, in the future perhaps someone will

design a high-performance RF deck to mate specifically within the SDR Cube enclosure!

Features, Goals & Specifications

Good designs normally start with a description of the intended goals, and how well the designers keep those goals in sight usually determines the degree of success that can be achieved for the intended market. In our case, it was important for us to keep in mind the natural constraints that come from limited resources and computing power of an embedded signal processing system. Our overriding goal was to provide “just enough” features without over-taxing the available resources; or conversely without creating a radio with more bells and whistles than are actually warranted.

Portable, standalone, QRP-level, SDR transceiver ... We wanted to provide an SDR transceiver for voice and CW modes that did not require a PC for signal processing or user interface.

Built-in “RF deck” ... While designing an embedded signal processing engine is difficult and time consuming, it would take even more time to also develop unique RF electronics to handle the front end of the HF transceiver. So by designing around the most popular Softrock of all time, the RXTX v6.3, we were able to tap into thousands of users who already have this RF deck, thus enabling them to keep costs down when building the SDR Cube.

HF modem: Processor & Codec ... We decided to use the somewhat dated Microchip dsPIC33F processor, despite there being faster DSP chips around these days. There is a lot to be said for a designer being comfortable in a known development environment such as this, and another project using the dsPIC33F (the NUE-PSK Digital Modem) offered some helpful functions and drivers that we could inherit in the SDR Cube. And for similar reasons, the older TLV320AIC23B codec is still plentiful and cheap, and it served us well for a first implementation. Both of these all-important components can be upgraded in future hardware/software versions of the Cube if needed and warranted.

Physical User Interface (Display & Controls) ... We were very interested in having this SDR project not look at all like a computer, much less have to depend on the PC for anything along the way. Given the smaller form factor we wanted for portability, we determined the bare minimum number and type of controls that would look like a “real radio”, while at the same time still have it be a pleasure to use. So often in these days of microcontroller-based rigs and station accessories, multi-function pushbuttons and menus-within-menus displayed on an LCD make the radio confusing and difficult to control. While we do have four pushbuttons as modal controls menu, VFO, Tuning Rate and Mode), we limited the number of functions that were available with each pushbutton and made the state of the radio quite visible in the large graphical display at all times. Four standard mini-potentiometers were used for the often-changed functions: audio level, RF attenuation, keyer speed and audio filter. And for the most often used control on any radio, the frequency tuning dial, we took pains to ensure that we used a smooth-turning and good quality rotary encoder, topped off with a fairly nice aluminum knob. A small piezo speaker sits behind a hole in the front panel serving as an optional “click” for user functions and for an audio sidetone that can double as a code practice oscillator! Lastly, the most exciting part of the User Interface was the bright blue graphical LCD. At 128 by 64 pixels, it offers enough resolution to allow us to simultaneously display many different status indicators (audio and RF levels, keyer speed, operating mode, frequency memory bank in use, RIT/XIT, Rx-Tx) along with our biggest feature of the front panel: a bandscope displaying 8 kHz of live spectrum that the Cube is tuned to.

Software Field Upgradeability ... A feature that has served both of us well in our previous designs (Micro908, NUE-PSK, Juma TRX) is the ability for the SDR Cube software to be updated by the user whenever newer versions become available. This integrated bootloader enables the user always be up-to-date after flash programming the latest binaries via the RS-232 serial port available on the rear panel’s Aux jack.

Cube Hardware

Diving just a little deeper into the hardware architecture of the SDR Cube, Figure 2 illustrates how the dsPIC33F is at the heart of the design. This powerful embedded processor handles the many front panel input controls, displays text and graphics to a 128 x 64 pixel graphic LCD, and controls the audio data paths through the codec and over to the Softrock RF deck on the right. And of course the dsPIC performs the phase shifting of the I/Q digital signal streams as part of the HF modulation and demodulation operation.

The dsPIC33F is a 40 MIPS controller (40 million instructions per second), which is only moderately capable compared to today's technology. But unlike some of the more powerful embedded DSP controllers from Texas Instruments or Freescale, the dsPIC package is relatively straightforward for homebrewers to attach on pc boards. Even with the 100-pin package of the dsPIC model we use, it is able to be attached in any number of ways: flood solder and wick off, individually soldering of each lead, or using the KD5SSJ solder paste and heat gun solder reflow technique. Microchip offers a free, very capable and intuitive development environment called MPLAB, enabling experimenters to focus on the design and not the tool. For these reasons, the dsPIC controller remains popular in the experimenter community.

The dsPIC offers enough capabilities for designers

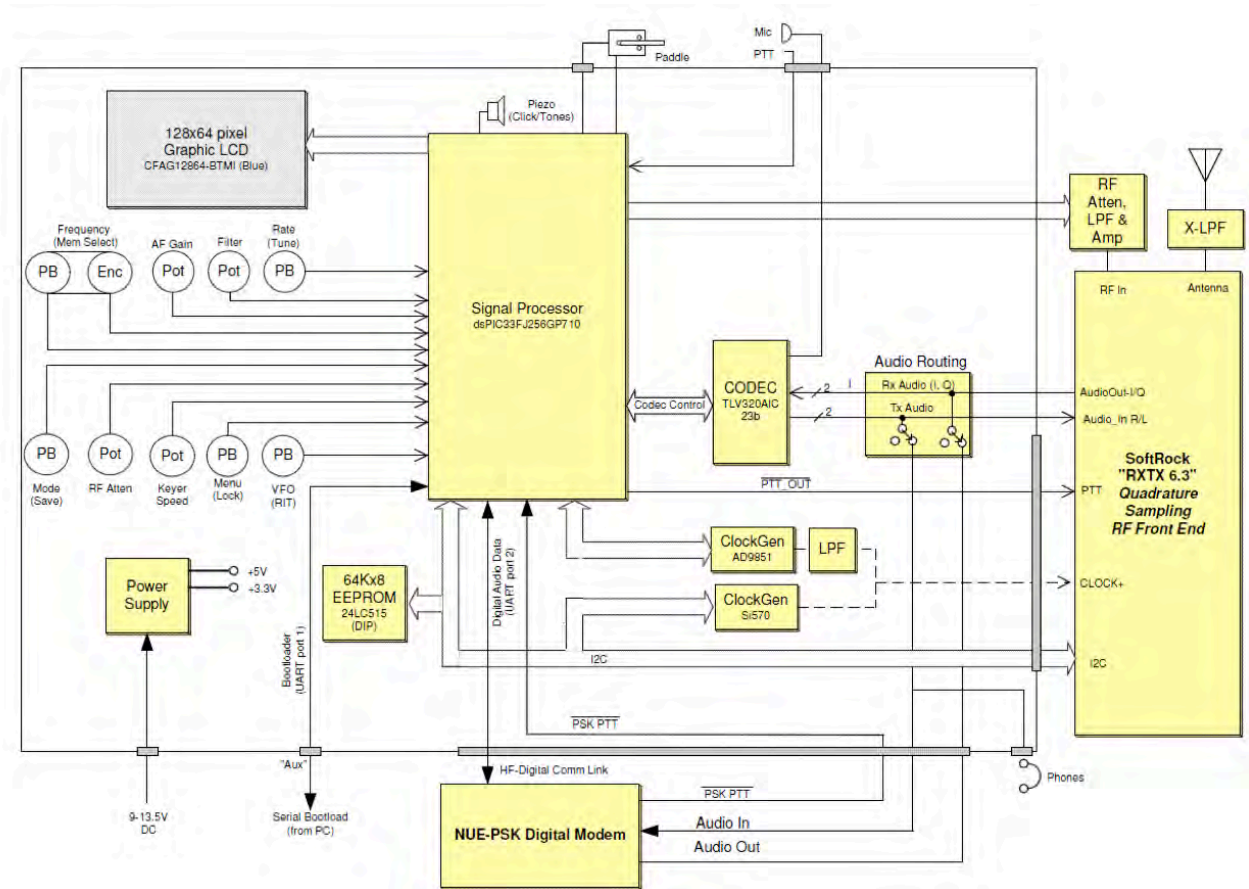


Figure 2: SDR Cube Hardware Architecture. Illustrates designed-in controls, graphic display, and tight integration with NUE-PSK Digital Modem.

to provide some demanding solutions: a 16-bit fixed point architecture, yielding 96 dB dynamic range in its computations. It has 40-bit accumulators and hardware support for division operations, barrel shifters, multipliers and a large array of 16-bit working registers – all of which provide ample DSP power for the operations performed in such an embedded SDR. Admittedly, the dsPIC controllers are getting “long in the tooth”, however Microchip is committed to continuing the product line, perhaps due to continuing applications like these here in the ham radio community.

For the all-important A/D and D/A conversion block we used the Texas Instruments TLV320AIC23B high performance stereo codec. In our first implementation, an 8 kHz sample rate is used, as anything much more than this is difficult for the dsPIC to handle. A higher sampling rate has a two-fold performance cost: (1) Filtering must be performed more often and at the pace of the sample rate; and (2) More filter taps are needed, resulting in more computation time required, which is a subtle and often-overlooked consideration in design. In later versions of Cube software we may be implementing additional processing hardware to accommodate increased computational power; for example, providing a wider spectrum band scope and digital modem functionality. So keeping the computing overhead lower is just as important as achieving adequate quality. The integrated headphone amplifier, programmable gain microphone amplifier, and SPI control bus were important selection criteria for this codec.

Optimal gain distribution in the system has not yet been determined, and AGC is an important (and difficult) feature to implement. Thus we have started getting our hands around receive path gain by implementing a controllable RF Attenuator as a plug-in for the Softrock board in place of its existing BFP board. In this way we can have several levels of signal attenuation and gain to assist in establishing optimal signal levels throughout the system.

Clock generation is optimally provided by the dsPIC's I2C bus controlling the Si570 present on the Softrock board. For those RF decks that do not have a

built-in clock generator, the SDR Cube design can provide the clock by its own Si570 clock chip or by a classical AD9851 DDS signal generator and low pass filter. Any of these three clocking methods may be implemented in the Cube.

Software

The design of the SDR Cube software is based on the phasing method of SSB generation and detection, whereby a 90-degree delay is imposed on the source audio signal coming from the microphone, or from a NUE-PSK Digital Modem. The resultant signals are applied to a pair of mixers – in our case these are provided on the Softrock board – with 0- and 90-degree LO signals. The output of the mixer is amplified on the Softrock and presented to the output BNC connector on the rear of the Cube.

The receive path works in a similar manner, but in reverse. The received signal is mixed with LO signals on the Softrock card that are at 0- and 90-degrees, producing the I and Q signals. These component signals are digitized by the Cube hardware and the dsPIC controller imposes a 90-degree phase shift between them by passing them through two balanced and matched FIR filters, one advancing the signal 45 degrees and the other delaying the signal by 45 degrees. Maintaining accurate phase delays and corresponding amplitudes has a direct impact on the quality of the demodulated signal. The demodulation occurs at the summing junction, shown in Figure 2 below; when the dsPIC performs the computations for the desired mode, the receive audio is able to be heard in the headphones.

Note that either USB or LSB may be demodulated merely by reversing the I and Q signals where indicated in Figure 2.

Codec ... The TLV320AIC23B is a relatively mature codec. The main reason we used this particular part was the accumulated SW experiences we have for controlling it. The '320 also provides handy adjustable input amplifiers, mic amplifier and head phone amplifiers. Most of the audio signal routing is done with the codec mux block and codec driver software. Only the audio path

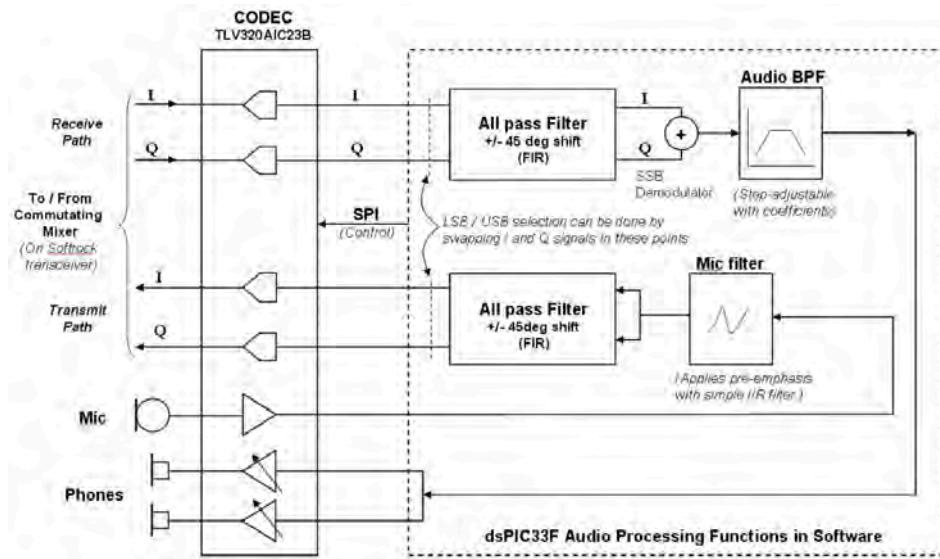


Figure 3: Software Processing Block Diagram

for the PSK modem needed external switching.

The TLV320AIC23B provides 24-bit ADC output. Our processing happens in 16-bit resolution but we can benefit from the 24 bits by selecting the magnitude that we use from it. For example, if we start from bit 22 instead of bit 23 we get 6dB gain, and so on. However the signal/noise ratio gets worse on every bit reduction, and practical numbers are 1 to 3 bits. More can be used, yielding 6 to 18dB of “free gain”.

System Gain Distribution ... And speaking of gain, we have plenty of adjustment opportunities throughout the system, from antenna to speaker. Each of the blocks shown in Figure 4 offers an opportunity to change the gain, and the goal would be to set each signal to the greatest level able to be processed by that stage without overflowing the digital registers, yielding the most dynamic

range for each stage. Different from an analog radio, when digital registers overflow much more noticeable (bad) effects are evident. So it becomes most important to ensure that the digital levels are well managed.

And therein lies the challenge (opportunity) for providing automatic gain control (AGC). A designer’s first tendency is to implement a simple algorithm such as “when the codec signals are too high, reduce the line gain”, or even something a tad more complex such as a log filter based on envelope detection, with separate up/down slopes. However if the other gain blocks are not also properly handled, simple solutions fall apart in real world conditions. Further, the design needs to handle a number of conditions: in-band signals that pass through all filters, out-of-band signals that do not go through the DSP filters, and signals greater than the Nyquist rate that can overload the codec input. VK6APH wrote an excellent paper describing the AGC challenges, as did Sabin et al. (See our References section.) These papers are guiding us for implementing AGC in one of our next releases. At the end of the day, “no AGC” is better than having poorly-implemented AGC – and in the initial release of the SDR Cube software, the operator is provided with good manual control of the system gain via the RF Attenuator control on the front panel.

Basic System Timing ... Of course there is no operating system in our embedded SDR Cube design. If anything, there would normally be an embedded OS such as VXworks (or equiv). But here the processor is quite simple and the timing needs are somewhat extreme, so we are running on “bare metal” with a simple master loop performing the computations and I/O data movement as needed. DSP calculations are done on every sample at 1/8000 Hertz, or 125 us sample rate. The filter calculations for each sample takes about 16-to-20us. The bandscope FFT is calculated in main loop when the FFT data buffer is full at 16 ms intervals. The keyer and other time critical tasks operate as timed with a 1 ms interrupt. And the processor’s PWM generator hardware is used to produce clean UI tones.

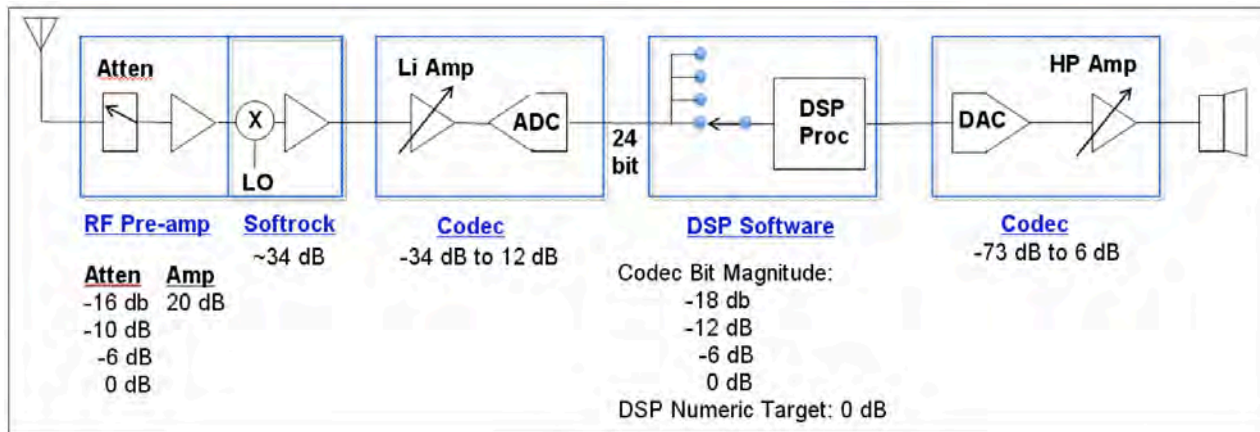


Figure 4: System Gain Distribution

Real-Time Bandscope ... This is a feature that we are quite proud of, and one which we think makes the SDR Cube stand out in a crowd of simple embedded SDR solutions. Besides being “sexy”, a live spectrum display showing all signals within the 8 kHz RF swath being processed by the Cube can be a very useful indicator for the operator. One can “see” nearby signals that might be of interest for contact or listening. And the characteristic wave shape of the spectrum signal is often a good indicator of the signals mode, which can be very helpful when operating the digital modes.

We added the ability for the user to select (in the Configuration Menu) the desired type of FFT algorithm: Hanning, Blackman or Rectangular. The computation is a 128-sample complex FFT, as performed with the (corrected) Microchip DSP library functions. We need the log() function to display the bandscope, which happens to be our only execution-optimized function. The whole process takes only 650 us.



Photo 3: Live bandscope view on the SDR Cube display

Calculating the FFT from the complex data yields both positive and negative frequencies. FFT results are produced for LO + 4000 Hz and also LO - 4000 Hz, which in the real world translates to having a +/- 4000 Hz band scope. The LO (dial frequency) is at the center, translating to 0 Hz audio, with USB to the right side of the center point. LO + 4000 Hz is at the right corner. LSB is on the left side of the center point, and LO - 4000 Hz is at the left corner. Display marker dots are shown at 1 kHz points. Fast attack and slow release of the display filter makes it easy to visually follow signals.

State-of-the-Art CW Keying and Wave Shaping ... Many PC-based SDRs have difficulty with fast transitions between Rx and Tx while in CW mode. “QSK” becomes hard to successfully implement when many layers of complex Windows software are used in processing the signals. However life on the embedded “bare metal” SDR Cube is quite different ... we have virtually no delay from key down to RF, because we use a “software DDS” function to generate the sine and cosine carrier signals for I-Q direct feed to the Softrock. Thus for a Cube user to break in, there is only 20ms lag (our receive path), and the transmit side is immediate!

Another CW feature is our sinc ($\sin(x)/x$) shaping of the CW edges ... smooth and glitch-free as silk. So in addition to the 0-100 wpm keyer, adjustable sidetone frequency, 300 Hz filter, adaptive Rx changeover delay, sinc waveform shaping, and Iambic-A, -B or Dot priority keying, we also probably have the fastest DSP CW processing in the market.

Binaural Audio ... Many ops feel that one of the most pleasing and useful audio features available uniquely on I/Q-based radios is that of binaural audio. As Rick Campbell, KK7B demonstrated in his seminal *QST* article in 1999, “the sound of CW signals on a binaural I-Q receiver is like listening to a stereo recording made with two identical microphones spaced about six inches apart. The same information is present on each channel, but the relative phase provides a stereo effect that is perceived as three-dimensional space.” The SDR Cube does not implement binaural audio in this conventional manner,

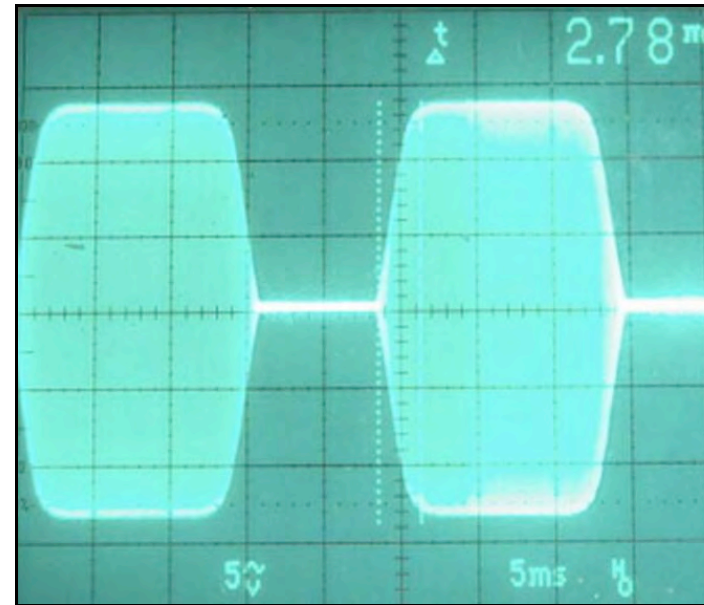


Photo 4: Sinc function applied to keying envelope on leading and trailing edges reduces energy at the transitions (clicks) and provides for a pleasing signal.

as illustrated on the left in Figure 5; but instead we approximate the effect by implementing a simple and adjustable 768-word delay line, as show on the right in Figure 5. Samples are shifted in and out at a 125 us rate (8 kHz), and the maximum binaural delay is $768 * 125\text{us} = 96\text{ms}$. The operator can adjust the delay from the menu in 1 ms steps to give the received signal a depth and richness that must be experienced to be appreciated. CW signals seem like they are traveling through your head, from one ear to the other, while tuning a station!

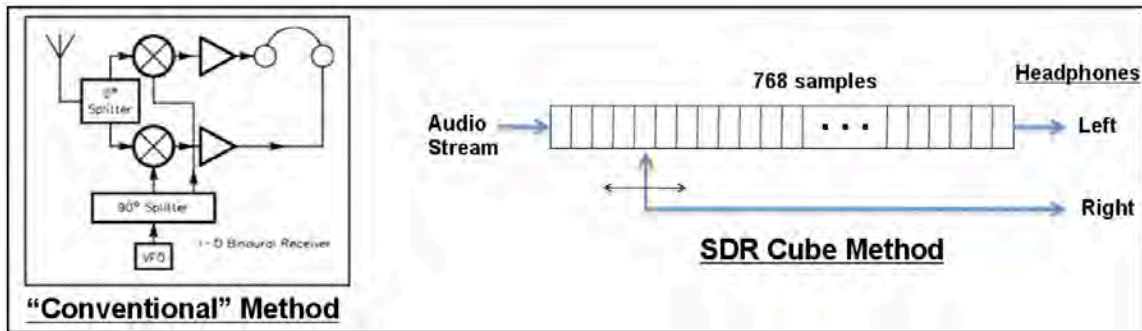
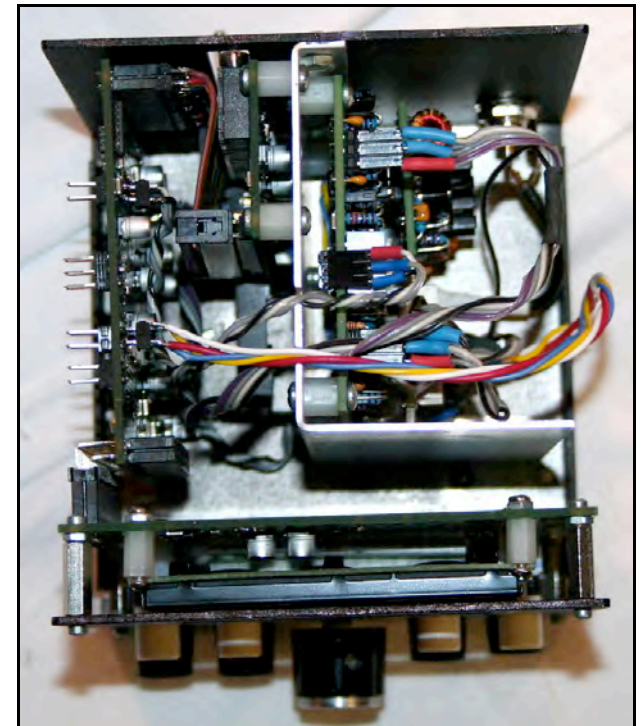


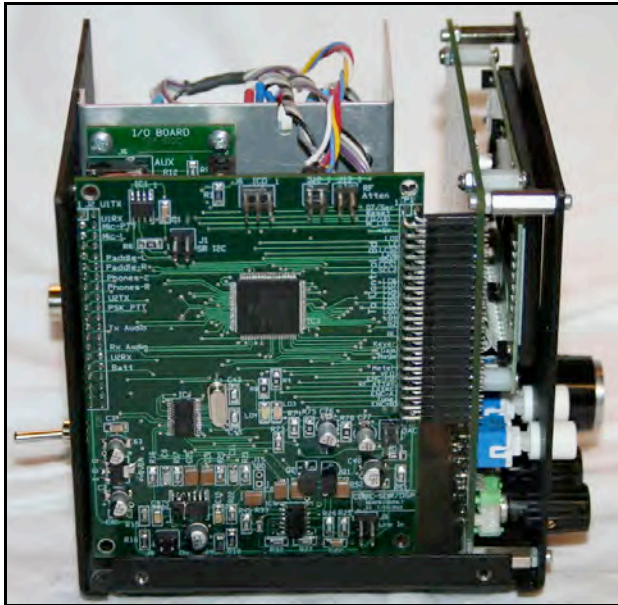
Figure 5: Quasi-Binaural Audio implemented in the Cube. “Once my ears got used to the effect, they had to drag me away from this radio. This is one I gotta have!”—Ed Hare, W1RFI, ARRL Lab Supervisor QST March 1999, Rick Campbell KK7B article: “A Binaural I-Q Receiver”

Mechanical Construction

The SDR Cube design is comprised of three printed circuit boards, as depicted in Photos 5 & 6. The Controls board is oriented behind the front panel of the enclosure and contains all the user controls, graphics display, and clock generation options. The DSP board plugs into the end of the Controls Board at a right angle and sits front-to-back along the left side of the enclosure. And the I/O board plugs into the DSP card and contains most of the electrical connections to the outside world. An inner L-shaped aluminum bracket, serves to shield the digital environment of the Cube electronics from the RF compartment in which the Softrock resides. This RF compartment is sized to ideally contain the Softrock RXTX 6.3, however other electronics (or an entirely different Softrock transceiver) may be fit into this space. The custom-designed 4” x 4” x 4” aluminum enclosure is a clamshell to facilitate easy access to the inner workings of the Cube during assembly, troubleshooting or modification.



Photos 5: Top view of the SDR Cube



Photos 6: Side view of the SDR Cube

In Reflection

Software defined radio is a relatively new topic in the Amateur Radio community, and one that is extremely exciting with many barriers regularly being broken. Perhaps the most enticing is the “embedded SDR” barrier, whereby designers are now starting to break free of the artificial limitations imposed by bloated and lethargic PC operating systems that cripple the very technologies which we seek to push forward. By instead approaching an SDR solution on the “bare metal” of an embedded DSP architecture, we are able to more quickly achieve our design goals, and do it in a more comprehensive manner.

Can the SDR Cube approach compete with the horsepower and beauty of the current PC-based SDR products? Certainly not in every category. The slower sample rate yields a smaller swath of the RF spectrum being viewed by the operator. Fewer graphics and limited information content is available on the small 128x64 pixel display. The front panel is somewhat crowded because of competing needs for portability and having knobs and physical controls. And the embedded processor itself is very slow compared to its PC counterparts.

But because it does not have the lethargic, immutable and often arcane multi-layered software architecture of an operating system, the embedded and OS-less SDR Cube is able to run circles around the PC in several important ways. By running “on the bare metal”, the Cube can effect near-immediate changes in hardware when needed – e.g., for T-R switching – thus giving contest ops an edge and ragchewing a splendid character. The Cube is self-contained – no operating system patches to install, drivers to obtain, no Microsoft Cluster Server required to configure.

The SDR Cube Transceiver is among the first of its kind, providing performance-oriented SDR in an affordable, modular and extensible fashion. Built initially upon the ubiquitous and wildly popular Softrock “RF front ends”, the open software Cube design is applicable to thousands of hams worldwide ... and we eagerly anticipate novel applications of it going forward. Whether it is the addition of a truly high-performance RF deck, or the tight integration of digital mode processing, or someone’s clever packaging to more easily take it to the field, we look to the future with pride and confidence.

But of all our excitement and feelings of accomplishment, we believe the most notable achievement with the SDR Cube is making SDR look and feel like a real radio, while simultaneously untethering the radio from the grips of the dreaded and unnecessarily complex PC. To again paraphrase Alfonso Bedoya in the 1948 classic movie: *The Treasure of the Sierra Madre* ... “We don’t need no stinkin’ PC!”

Epilogue: The Evolution of Software Defined Radios

Software Defined Radio (SDR) in the Amateur Radio community has been making great strides in recent years. From the innovative and ground-breaking products of Gerald Youngblood, KSDR of Flex Radio some four years ago, to the most recent state-of-the-art designs in the HPSDR group, SDR technology has really now come of age. This current state-of-the-art has also been greatly enabled by the tireless work of Tony Parks, KB9YIG, the father of the Softrock designs, who has empowered thousands of hams worldwide with his series of inexpensive radios that work with PC sound cards.

Each of these radios in their most basic form consists of electronics that sample the incoming RF after it has been converted down to baseband and send the results to a PC for digital conversion. The PC then performs the complex demodulation computations so we operators can understand the SSB, AM or digital mode communications coming in. And of course the reverse happens for transmit, whereby the PC presents electronic signals to the front end electronics for mixing and ultimate transmission.

However throughout all the excitement of PC-based software defined radio, there has also been a quieter background quest for a form of SDR that is not tethered to a PC. While the PC offers seemingly unlimited processing power, gorgeous user interfaces and lots of memory, the PC is still an expensive and cumbersome accessory to take to the field when the need arises for portable operations. Classic problems are encountered with regard to the ability to see the PC display in bright sunlight, and powering the PC and the power-hungry SDR front-end electronics is tough with limited-capacity batteries. In general, lugging around an expensive and delicate laptop is not something one wants to regularly do. Also, many hams just do not care to operate a ham radio with a mouse controlling knobs shown on a PC screen.

Most would still agree that the *performance* of such a PC-based radio is of great value, given the flexibility offered with SDR's innate ability to handle virtually any operating mode with just a new software file or program loaded into the radio.

Numerous experimenters have been seeking to develop an integrated SDR transceiver to address the shortcomings of using a PC in the field, while retaining enough of the benefits of SDR in general so as to have an inexpensive-yet-powerful transceiver. Embedded SDR projects are in progress with in the High Performance SDR Group (HPSDR) and yet another is being described in an excellent article series by AE6TY in QRP Quarterly Magazine. Many of us experimenters are standing on the shoulders of previous pioneers in the field of DSP for Amateur Radio: Rob Frohne KL7NA, Lyle Johnson KK7P, Rick Campbell KK7B and others. The handbooks from the ARRL and RSGB, as well as the seminal work of "Experimental Methods for RF Design" by Hayward/Campbell/Larkin, are replete with the basic building blocks that we designers of today are employing in our SDR implementations. ~ N2APB

About The Designers

George Heron, N2APB, is a cyber security professional in the Baltimore-DC region helping to protect users from viruses and other types of malware. First licensed in 1968, George holds an Amateur Extra class license and is an avid homebrewer in RF and digital circuits, with special interests in DSP and microcontroller applications to QRP. His design accomplishments include the NUE-PSK Digital Modem, the Micro908 Antenna Analyzer, the DDS-60 Daughtercard, and many other QRP kits. He co-leads the New Jersey QRP and the American QRP clubs, and has previously edited/published *Homebrewer Magazine*. George can be reached at 2419 Feather Mae Ct, Forest Hill, MD 21050, or at n2apb@midnightdesignsolutions.com.

Juha Niinikoski, OH2NLT, lives in Espoo, Finland (near Helsinki) with his wife and two sons – one son is also a ham! Juha is a co-owner of a small hardware and software design house and is a co-developer of the Finnish JUMA kit line of ham radio products. Juha enjoys rag chewing and especially experimenting with new technologies and designing equipment. He received his first license in 1988 - a Technical class license which was later upgraded to General class. OH2NLT can be reached at Etuniementie 11 C, 02230 Espoo, Finland or at juha.niinikoski@sitecno.fi.

References

1. SDR Cube website, <http://www.sdr-cube.com> ... contains schematics, photos, parts lists, kits and all other technical information
2. SDR Cube on Yahoo Groups, <http://groups.yahoo.com/group/sdr-cube/> ... active email list
3. CheapDSP, Juha Niinikoski, OH2NLT, http://www.kolumbus.fi/juha.niinikoski/Cheap_dsp/Cheap_dsp.htm
4. FlexRadio, Gerald Youngblood K5SDR, <http://www.flex-radio.com/Default.aspx>

5. NUE-PSK Digital Modem, George Heron, N2APB and Milt Cram, W8NUE, <http://www.nue-psk.com/>
6. *Experimental Methods in RF Design*, Chapter 11, Wes Hayward, W7ZOI, Rick Campbell, KK7B, Bob Larkin, W7PUA, <http://www.qrp.pops.net/cmrfd.asp>
7. *The ARRL Handbook*, American Radio Relay League, <http://www.arrl.org/shop/The-ARRL-Handbook-for-Radio-Communications/>
8. Various articles in *DSP Processing*, ARRL, <http://www.arrl.org/dsp-digital-signal-processing>
9. High Performance SDR (HPSDR), <http://openhpsdr.org/>
10. HPSDR Boards, TAPR, <http://www.tapr.org>
11. *RSGB Radio Communication Handbook*, Radio Society of Great Britain, <http://www.vhfcomm.co.uk/rsgb-handbook.htm>
12. Softrock Transceivers, Tony Parks, KB9YIG, <http://www.kb9yig.com/>
13. Softrock RXTX V6.3, http://www.wb5rvz.com/sdr/RXTX_V6_3/
14. SMT Soldering Tools and Process, Cash Olsen KD5SSJ, http://kd5ssj.com/index.php?option=com_content&view=article&id=125&Itemid=129
15. Phil Harman VK6APH on AGC, <http://support.flex-radio.com/Downloads.aspx?id=98>
16. Sabin on AGC, Excerpt from *HF Radio Systems and Circuits*, <http://www.ab4oj.com/icom/agc.pdf>

###

TAPR At Hamvention

TAPR will make its annual appearance at the Dayton Hamvention. (What would the Hamvention be without TAPR?)

Big news is that David Rowe, VK5DGR, will speak at our forum about his work on CODEC2. During the past year, there has been a lot of progress on the CODEC2 front as documented on David's website (http://www.rowetel.com/blog/?page_id=452).

CODEC 2

CODEC 2 is an open source CODEC designed for low bit rate speech over HF/VHF digital radio. Whereas most low bit rate CODECs are proprietary and require licensing fees, CODEC 2 is unique in that it is open source, allowing experimentation and modification.

Author and developer David Rowe, VK5DGR, will explain CODEC 2 and how it works, breaking down the complex DSP into simple terms. He will talk about the development process and the challenges he has faced along the way and present examples of CODEC 2 use with Amateur Radio.

David will also be the after dinner speaker at the TAPR-AMSAT Banquet where he will reveal everything you want to know about the Village Telco Project (<http://www.villagetelco.org/>).

There are now billions of mobile phones all over the world. However in many developing countries, mobile phones are simply too expensive to use. For example, in a country where people earn \$1.50/day, a phone call can cost 21 cents/minute. This is a huge tax on communications.

David will describe how a community of people from around the world set out to fix this problem, using a mix of Wi-Fi and open source software. The result is the Village Telco - a DIY telephone company that anyone can set up.

Village Telco Project

A Village Telco is constructed from a network of "Mesh Potatoes" - small Wi-Fi boxes that are mounted on homes in a village. They form a mesh network,

relaying telephone calls for each other. Each Mesh Potato connects to a regular telephone inside village homes. Phone calls on the network are free. A local person in the Village can be trained in a few hours to set up and maintain the network.

To build the Village Telco we borrowed many ideas from Ham Radio, such as sharing hardware designs and community networks.

The talk will include real world stories of how Village Telcos are helping people in several developing countries, and explain why a simple phone call that we take for granted can be so important.

The Rest of the Story

Here is the complete rundown of TAPR's activities at this year's Hamvention.

TAPR Board of Directors Meeting

Thursday, May 19, 7 PM – The board meeting is open to all TAPR members and will be at the Ramada Plaza Dayton, 2301 Wagner Ford Rd, Dayton, OH 45414.

TAPR Booths

Friday, May 20 to Sunday May 22 – TAPR will be at booths 455-458 in the Ball Arena of the Hara Arena (same place as last year in the same arena with ARRL and AMSAT). The booth schedule is Friday 9 AM to 6 PM, Saturday 8 AM to 5 PM, Sunday 8 AM to 1 PM.

TAPR Forum

Friday, May 20, 9:15 to 10:55 AM – The TAPR Forum will be in a Hara Arena meeting room TBD featuring:

TAPR Update – Steve Bible, N7HPR

HPSDR Update - Phil Harman, VK6APH

CODEC2 Update - David Rowe, VK5DGR

2011 TAPR ARRL DCC

TAPR-AMSAT Banquet

Friday, May 20, 6:30 PM – The TAPR-AMSAT Banquet will be at Kohler Presidential Banquet Center, 4572 Presidential Way, Kettering, OH 45429, featuring David Rowe’s after-dinner talk about the Village Telco Project.

Reservations are required and must be made by Monday, May 10, 2010. Seating is limited, so make your reservations early. The price for the banquet is \$30 per person. Reserved tickets may be picked up at the AMSAT booth at Hamvention on Friday or at the door. You may make reservations for the 2011 AMSAT/TAPR Banquet online in the AMSAT Store (<http://www.amsat-na.com/store/item.php?id=100158>).

For more information and the banquet menu, visit <http://www.amsat.org/amsat-new/hamvention/2011/Banquet.php>.

###

Mark your calendar and start making plans to attend the premier technical conference of the year, the 30th Annual ARRL and TAPR Digital Communications Conference (DCC) to be held September 16-18, 2011, in Baltimore, MD. The conference location is the Four Points by Sheraton BWI Airport, Baltimore, MD (<http://www.tapr.org/dcc#hotel>).

Call for Papers

Technical papers are solicited for presentation at the 30th Annual ARRL and TAPR Digital Communications Conference to be held September 16-18, 2011 in Baltimore, MD and publication in the Conference Proceedings. Annual conference proceedings are published by the ARRL. Presentation at the conference is not required for publication. Submission of papers are due by July 31st, 2011 and should be submitted to:

Maty Weinberg, ARRL
225 Main Street
Newington, CT 06111

or via the Internet to maty@arrl.org

Full Details on Call for Papers: <http://www.tapr.org/dcc#dcccallforpapers>

Submission Guidelines: <http://www.tapr.org/dcc#dccsubmissionguidelines>

###

A Volunteer

By John Koster, W9DDD

As you may know, we have had problems with serving up the Real Audio content reliably over the years. To make matters worse, during one of the server upgrades at tapr.org, the files were lost.

A few months back, Paul Maruna, KD8HWP, was waiting to listen to a talk given by Greg Jones, WD5IVD, at Hamvention. Once the problem was described to Paul, he volunteered to help convert the Real Audio files to MP3 format.

Paul was sent CD-ROMs of DCCs 1-20, which had the DCC RA files on them. At present, he and I are in the process of converting files and updating page links to the new MP3 version files of talks at Digital Communications Conferences. Much work remains to be done, but Paul deserves a bigatta boy for stepping forward.

Greg Jones, WD5IVD, has been able find his archives of the older audio files and have presented us with a zip file of over 1GB of audio.

Once the project is completed we should have all the audio on-line for conferences from the first year recorded (1996?) up through the year DVD production commenced, except for the year the recording equipment had a meltdown at DCC in Tucson.

###

Metis

High Speed HPSDR Interface

METIS is a high speed PC interface. While the original Ozy board has served us well to date, in order to implement some of the future HPSDR projects we are going to need a faster interface between the various boards on the Atlas bus and the PC. METIS is a kick off point for this project. Since this board will not need to support the SDR1000 there will be room for testing other high performance interfaces. Metis an Atlas size board that contains a large, leaded, Altera Cyclone III FPGA connected to a Gigabit Ethernet PHY.

More detail are available at the Metis Wiki Page (<http://openhpsdr.org/wiki/index.php?title=METIS>).

Ordering Information

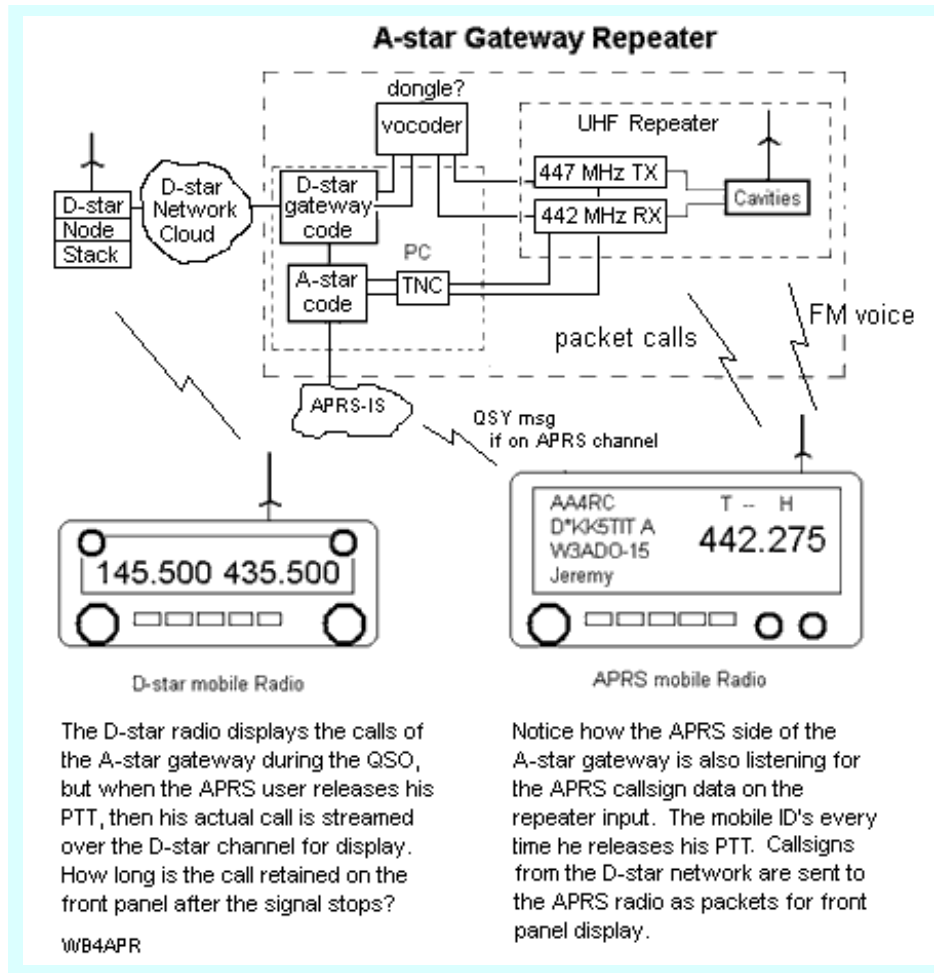
The price for the Metis is: \$279 US for members of TAPR, \$309 US for non-members plus shipping/handling if applicable.

Order online at https://www.tapr.org/tapr_addorder.php?add=604

###

AVRS - Automatic Voice Relay System

By Bob Bruninga, WB4APR



AVRS is the ham radio Universal Voice Contact System equivalent to the Universal Ham Radio Text Messaging (<http://www.aprs.org/aprs-messaging.html>) initiative. Both of these initiatives attempt to cross connect all ham radio systems so that only a call sign is needed to establish TEXT or VOICE contact. For Voice, we are trying to cross connect all of the VoIP ham radio link Programs (IRLP, ECHOLink, WIRES, D-STAR etc)! . Think of AVRS as Ham Radio's Cellular-by-Call sign system where you use your radio to make a call to a call sign instead of a phone number. . APRS is the backbone signaling system locating the end CALLSIGN users and the Links needed to set up the call. This is just like D-star, except it works for any analog system such as Echolink, IRLP, or Wires. D-star already has end-to-end call sign connectivity. APRS can bring this to any analog repeater! Here are the steps in making a call (anywhere on the planet):

Monitoring:

- * Puts APRS band to desired A*star frequency
- * Enables PTT mode (APRS ID goes out with release of PTT)
- * Kerchunks repeater
- * Repeater "welcomes his callsign" and updates D*star registry

Making a Call:

- * Finds CALLEE in station or message list (or enters call)
- * Presses MSG button to initiate a message
- * Enters "A*" and hits OK

A*star Gateway Actions:

- * Makes the normal D*star connection to CALLEE
- * Announces connection
- * QSO begins

Receiving a call on normal APRS channel:

- * Message from CALLER is announced
- * Object fm A*star shows Freq/Tone/shift range & bearing
- * User presses QSY button and makes voice response.

Receiving a call on A*star channel:

- * You will hear the distant CALLEE voice call
- * Front panel will display above message and INFO

The above boxes show how an A*Star call is made from both the sender and receiver perspective. The A*star QSO can be between any other A*star user or any other D*star user seamlessly. Future versions of the D710, D72 and FTM350 and others can automatically QSY the radio if the Menu option has been selected.

Original AVRS Goals

In the year 2000 or so, we introduced this AVRS concept of marrying APRS and Echolink/IRLP ham radio systems so that end-to-end mobile-to-mobile voice is possible with automatic set-up based only on call sign. . There were many long term objectives to reach this goal.

- Make local recommended voice repeaters show up on APRS travelers radios
- Make Echolink, IRLP, Wires, D-star Freq-objects show on radio front panels
- Make these objects show their operating freq, tones and node numbers
- Have D-star users show up in the APRS system
- Have mobile radios that can ONE-BUTTON QSY to these frequencies

- Introduce the Item-in-msg format so objects can be pushed through IGates
- Develop the AVRS engine on the APRS-IS to be the AVRS connection system



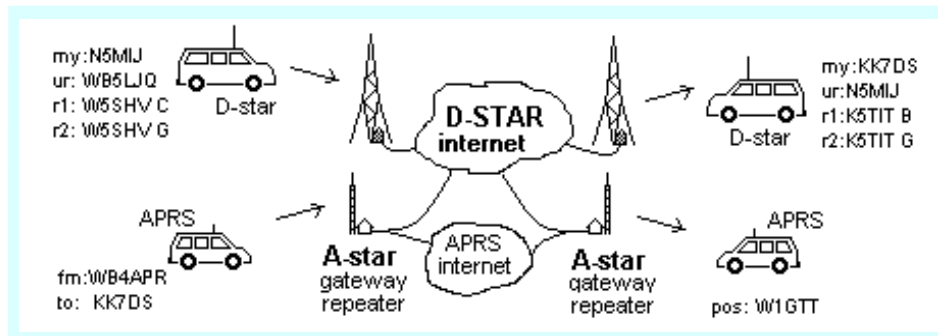
Node Objects

As of 2007, we have made substantial progress on the first 5 items above. . Many Echolink, IRLP, D-STAR and WIRES nodes now appear on the air and most importantly on the front panel displays of mobiles and HTs as shown to the right on the D7 HT. These photos need to be taken again, since they are not showing the Frequency of the node which is also very important. The proper formatting of these objects is shown in the Echo-IRLP-Win.txt file (<http://www.aprs.org/info/echo-irlp-win.txt>).



QSY Transceivers

Also since 2007, we now have many APRS radios (D710, D72, and FTM350) that can TUNE/QSY to a frequency listed on APRS with the push of a single button as shown to the right. Not only can the radio tune to IRLP/Echolink Frequency objects, they can also tune to any other Repeater or operator that has frequency in its position packet. Please see the Local Frequency Object concept (<http://www.aprs.org/localinfo.html>) that we have been pushing for a few years now. A few examples are shown on the radio to the right.



As of December 2010, we now also have found a motivated individual who is looking into writing the software for the AVRS engine to implement this AVRS Cellular-by-call system. Stay Tuned!

A-STAR REPEATERS: A possible short-cut to more rapidly reach the AVRS goal of universal voice connectivity is to simply build what I call A-STAR gateway repeaters. These analog repeaters or link stations appear to the APRS user as just an analog repeater that happens to listen for APRS identification of its users. But then it is also connected as a D-star Gateway device on the D-star network as shown on the top of the page and to the right. Since D-star users already have call sign-routing and repeater-routing the only thing needed for an A-star analog repeater is the call sign signaling, and APRS provides that. See the A-star spec (<http://www.aprs.org/avrs/A-star-spec.txt>).

Network	Caller	Callee	signalling comments	
D-star:	my: N5MIJ ur: WB5LJQ r1: K5TIT B r2: K5TIT G	my: WB5LJQ ur: N5MIJ r1: W5SHV A r2: W5SHV G	These four callsigns are the signalling method used by S-star in all QSO's	
	A-star:	fm: WB4APR to: ASTAR msg: A*WB5LJQ		pos: WB4APR
The A-star repeater gateway converts between D-star signalling and APRS signalling			fm: ASTAR to: WB4APR msg: QSY 147.105	If WB4APR is not currently on the A-star repeater, then he is sent an APRS message telling him or his radio to QSY to receive the call.

D-star to A-star call: First, the APRS users on the A-star repeater are all fully known because of their APRS beacons. The A-star repeaters know where all 40,000 APRS users are already. Thus, as shown in the lower right corner of this diagram, a mobile A-star user can receive a D-star call from anyone anywhere because the A-star gateway appears on the D-star network with that knowledge. The D-star system provides the end-to-end connectivity and the A-star gateway does the D-to-A conversion back to voice. This knowledge can be extended for incoming calls even if the APRS user is currently elsewhere on another frequency. An APRS message is simply sent to the APRS user signaling him or his radio to QSY to the A-star repeater to answer the call! [All new APRS radios (D710, FTM-350 and D72) can do this now with one button push. In the future, it can be automatic!]

A-star to D-star call: This is as simple as making a D-star call on a D-star radio, except that the A-star user enters the target call sign into an APRS message beginning with "A*". The A-star repeater gateway converts that to the streaming D-star call sign and the D-star network handles the rest. If the callee happens to be another APRS user, then an advantage of this APRS signaling

AVRS Java Parsers

By John Gorkos, AB000

is that he does not even need to be on the A-star frequency but can receive the APRS message and he or his radio can instantly QSY to receive the call. Of course, once the A-star repeater makes the A-star user appear identical to a D-star user, then everyone can talk to anyone in the same manner. Everyone appears on the D-star network as a D-star user.

APRS-PTT-Mode: The key to this process is the APRS user operating in PTT mode when he is on the A-star repeater channel. This was the original APRS concept so that background APRS signaling could be done transparently on the input channel of any Voice repeater. The APRS information is transmitted in a 0.3 second burst at the end of a voice transmission when the PTT is dropped. The A-star repeater has a TNC on its input listening for this data. That is how the A-STAR repeater knows that the local mobile is currently there. The D-star call signs are transmitted as a normal APRS packet back to the APRS radio for display on the front panel.

###

Write Here!

PSR is looking for a few good writers, particularly ham radio operators working on the digital side of our hobby, who would like to publicize their activities here.

You don't have to be Vonnegut to contribute to *PSR* and you don't have to use Microsoft Word to compose your thoughts. The *PSR* editorial staff can handle just about any text and graphic format, so don't be afraid to submit whatever you have to wallo@tapr.org.

The deadline for the next issue of *PSR* is April 15, so write early and write often.

###

As I mentioned in an earlier message, I'm working on a new set of Java Parsers and the implementation of AVRS (Automatic Voice Relay System). I had several people request access to the code for various reasons, so I created a SourceForge project called AVRS which can be found here:

<https://sourceforge.net/projects/avrs/>

I've not made any official releases yet, but all of my code is available in the SourceForge SVN repository. Right now, everything is tagged as GPL3, and I am committed to creating free and open-source software for the APRS community.

I realize there are already several APRS related projects at SourceForge. Of particular note is the fine work that Gregg Wonderly did with *JeAPRS*. While none of Gregg's work appears in my libraries, his code has been inspirational. Similarly, Matti Aarnio's code from <http://repo.ham.fi/websvn/java-aprs-fap/> has provided a foundation for the position parsers in my project.

While my final goal is to realize the AVRS system as proposed by WB4APR, that project requires a solid foundation of parsers and other libraries that will be useful in any APRS software.

If you are a developer committed to OSS principles and want to use the code I've developed, you're welcome to it. If you see errors, oversights, gross incompetence, or plain stupidity, you're welcome to ping me for write access to SVN or simply submit patches to fix what I've broken or missed.

I'm actively developing AVRS at a pretty good clip. I'll continue to post updates and invitations on the APRSSIG list, at least until someone tells me to knock it off.

###

PSR

#114 Winter 2011, ISSN: 1052-3626

Published by

TAPR

Phone 972-671-TAPR (8277)

E-mail taproffice@tapr.org

URL <http://www.tapr.org>

Facebook <http://www.facebook.com/pages/TAPR/116614778354245>

Twitter www.twitter.com/taprdigital

TAPR Office Hours: Monday to Friday, 9 AM to 5 PM Central Time

Submission Guidelines

TAPR is always interested in receiving information and articles for publication. If you have an idea for an article you would like to see, or you or someone you know is doing something that would interest TAPR, please contact the editor (wallou@tapr.org) so that your work can be shared with the Amateur Radio community. If you feel uncomfortable or otherwise unable to write an article yourself, please contact the editor for assistance. Preferred format for articles is plain ASCII text (OpenOffice or Microsoft Word is acceptable). Preferred graphic formats are PS/EPS/TIFF (diagrams, black and white photographs), or TIFF/JPEG/GIF (color photographs). Please submit graphics at a minimum of 300 DPI.

Production / Distribution

Packet Status Register is exported as Adobe Acrobat and distributed electronically at <http://www.tapr.org>

PSR Editor:

Stan Horzepa, WA1LOU

One Glen Avenue, Wolcott, CT 06716-1442 USA

Phone 203-879-1348

E-mail wallou@tapr.org

TAPR Officers

President: Steve Bible, N7HPR, n7hpr@tapr.org

Vice President: Scotty Cowling, WA2DFI, wa2dfi@tapr.org

Secretary: Stan Horzepa, WA1LOU, wallou@tapr.org

Treasurer: Tom Holmes, N8ZM, n8zm@tapr.org

TAPR Board of Directors

Board Member, Call Sign, Term Expires, e-mail address

John Ackermann, N8UR, 2013, n8ur@tapr.org

Steve Bible, N7HPR, 2011, n7hpr@tapr.org

Dan Babcock, N4XWE, 2013, n4xwe@tapr.org

Scotty Cowling, WA2DFI, 2012, wa2dfi@tapr.org

Stan Horzepa, WA1LOU, 2011, wallou@tapr.org

John Koster, W9DDD, 2012, w9ddd@tapr.org

Jeremy McDermond, NH6Z, 2013, mcdermj@xenotropic.com

Darryl Smith, VK2TDS, 2011, vk2tds@tapr.org

Mark Thompson, WB9QZB, 2012, wb9qzb@aol.com

TAPR is a not-for-profit scientific research and development corporation [Section 501(c)(3) of the US tax code]. Contributions are deductible to the extent allowed by US tax laws. TAPR is chartered in the State of Arizona for the purpose of designing and developing new systems for digital radio communication in the Amateur Radio Service, and for disseminating information required, during, and obtained from such research.