PACKET STATUS REGISTER

President's Corner

The 2009 ARRL-TAPR DCC is fast approaching, with details on our Web site (www.tapr.org). Please register early and often <grin>. Our Chicago organizers, led by Mark Thompson WB9QZB and Kermit Carlson W9XA, have been working full-tilt for the past year and are putting together what no doubt will be a very well-executed affair (that's a polite way of saying they are over-achievers). I've kept just about all of our emails, and Steve and I feel as if we've been right there in Chicago through the whole process. Together with Steve Bible, N7HPR, TAPR's DCC chairman, the crew have put together a very full schedule. Whether you are at the bleeding edge of digital communication, or simply want to find out a bit more about new modes of digital voice on HF or UHF, you will find a lecture track to suit your taste and interests..

For those of you that arrive early, the TAPR BoD will be meeting on Thursday, September 25, and observers are welcome to attend. Topics for discussion will include future projects, and the status of various ongoing commitments to HPSDR (High Performance Software Defined Radio). Then later, on Saturday, September 27, we will host the TAPR Annual Meeting, which will include presentations to the membership, the Treasurer's report (hosted by satellite from Brazil), and a Q&A session at which your BoD will field questions from the floor. As always, we will be eager to hear of YOUR ideas for future TAPR projects.

As I alluded to above, we continue to support HPSDR with assistance with board production. Penelope, a low power HF transmitter, was completed before Dayton this year. Mercury, the HF receiver, and Alex, the preselector/filter board, should be out by year-end barring any significant design changes. Watch the Web site for announcements regarding other offerings.

ICOM had previously donated a D-STAR 1.2 GHz repeater to TAPR, and this will be on the air at the DCC as KT7APR, thanks to a donation of a Triplexer by TX-RX. I will also be bringing a UHF module to add to the stack at the DCC, and have prepared a gateway box with Kermit, W9XA, and power supplies to get the whole system on the Internet for the duration of the DCC.

After the DCC, the system will be on the air in Ohio together with another system I have put on-line. I even have my repeater system back in London, Ontario (VE3TTT) on D-STAR, thanks to the work and generosity of several amateurs there. D-STAR is certainly showing rapid growth and there will be a Friday night D-STAR session hosted by Mark, WB9QZB. This will be a very useful how-to session, as well as a forum to discuss new developments in D-STAR technology.

I hope to see a lot of you at the DCC. While reading the papers in the proceedings is certainly a good thing, the REAL benefits of the DCC come from meeting and talking to other attendees. See you in Chicago, September 26-28.

73,

Dave VE3GYQ/W8

###

TAPR Board of Directors Election

Three Director positions on the TAPR board are now open for nomination.

Nominations may be submitted now. Visit www.tapr.org/tapr_elections.html for information about what being a TAPR director entails and also to place your nominations (please use plain text when placing a nomination).

There will also be a call for nominations at the annual membership meeting at the DCC in Chicago on September 26, 2008.

Nominations close at the end of the DCC on September 28, 2008, and an online election will be held at www.tapr.org/tapr_elections.html from October 4 to October 17, 2008.

The three Director positions that are up for election are currently held by Steve Bible, N7HPR, Stan Horzepa, WA1LOU, and Darryl Smith, VK2TDS.

###

HPSDR Project Production Status, August 2008

By Scotty Cowling WA2DFI, wa2dfi@tapr.org

The HPSDR project is alive and well and building up momentum for an exciting fall. There are two "subsystems" currently approaching production stage. The first is the Mercury direct sampling receiver board for the Atlas bus. The second really is a subsystem; Alex is a set of two RF filter boards, along with an enclosure and interconnecting cables.

Mercury is in the (we hope!) final throes of alpha stage. Phil VK6APH is the project leader with Lyle KK7P, Greg ZL3IX, Bill KD5TFD and myself offering varying levels of assistance. The performance of the first alpha board was good, but there were some thermal problems that needed to be addressed. The thermal problems were fixed, and at the same time a few changes were added to the filtering between the preamp and the ADC. Switching was added to allow the preamp to be switched out. Five alpha 2 boards were assembled for testing.

Unfortunately, the alpha 2 modifications caused some problems with filter matching and noise. With severe modification to his alpha 2 Mercury board, Phil was able to achieve the performance of the alpha 1 board. Phil's changes were tested out on three different alpha boards to insure that they worked as designed. The modifications were too extensive to risk going straight to production from alpha 2, so an alpha 3 build is currently in progress. The goal is to complete the third (and hopefully!) final alpha Mercury build will be completed in time for Phil to complete his testing before he leaves for DCC. Look for production Mercury boards by the end of the year.

The second HPSDR project is Alex, and is actually closer to production than Mercury. Alex is one of the most exciting HPSDR projects, in that it has more uses than just as an SDR front-end filter. Alex is a stand-alone set of switched RX/TX low-pass filters and RX high-pass filters. It also includes a low-noise 6m preamp, T/R switching, multiple RX and TRX antenna switching and connection circuitry for a transverter. Alex is controllable directly from Mercury or most any general-purpose three-wire serial interface. Graham KE9H is the project leader, with help from Phil VK6APH and Bill KD5TFD. Graham has conducted Alex through the alpha stage and various modifications and has blessed the boards for production. TAPR has not actually decided on a final production configuration, due to the large number of toroidal inductors (there are 38 of them!) Alex will likely be a partial kit with some assembly (read: coil winding) required. A very nice external enclosure will house the Alex board set. Look for Alex information beginning before DCC in September with late fall production.

Don't forget that TAPR has Atlas backplanes, Pinocchio extender boards, Ozy USB interface boards, Janus audio A/D-D/A boards and Penelope exciter boards all in stock for your SDR building pleasure.

###

Microwave Engineering Project

By Michelle Thompson, W5NYV, w5nyv@yahoo.com

The Microwave Engineering Project (MEP) aims to design and build a high-speed digital microwave band system for Amateur Radio that supports high-definition video, point-to-point, and multiple-access communications.

We would like to invite anyone interested in microwave communications to participate in the project. We have just kicked off our exploratory phase.

This is the part of the project where (according to several engineers I've had the opportunity to spend time with) the sentence "Because it's COOL!" is plenty good justification for bringing up a function or an idea. I very much want to know what interested Amateur Radio operators would like to see us try and accomplish together as a design and development team.

The only credential needed here is intellectual curiosity, a willingness to express yourself, and the intent to learn along the way. We are a group of ordinary people talking about doing extraordinary things. My motivation is to enable a supportive and collaborative engineering process, learn new things, and produce something at the end of the day that we can all be proud of.

Consider this phase to be open season with the goal being as good a description (vision) of the project as we are able to write down. This means taking a fresh look at what we can offer Amateur Radio, and what we want to work on and experiment with.

There are many ways to approach this type of phase of a project. Being able to concisely summarize what problem is being solved by this project, or what need is being met is very important because it provides a real foundation for producing a set of requirements, which will be the focus of the second phase of the project. That phase will be called requirements analysis.

Here is a brief description of what we have come up with so far as what we would like to produce.

We would like to design and build a high-speed digital system for the relatively under-utilized microwave bands of 3.4 and 5.6 GHz. We want to support high-definition video. We would like to design something that could adapt from point-to-point use to multiple-access use without a lot of fiddling around. We would very much like to include a satellite simulator in order to explore the development of, for example, delay-tolerant protocols and techniques that might be useful for experimental and educational purposes. We would like for the system to be durable, and portable, and fun to use. We would like it to be affordable, highly integrated, and high power.

I would like for it to do a lot of things, but most importantly, I would like to know what you all think it should do (or not do), and why.

The team has a Web site, a mailing list, and a podcast. If you subscribe to the podcast, all documents, videos, and audio recordings from the project will be delivered to the feed reader of your choice. The mailing list archives are open for public reading. The hardware will be designed using TAPR's open hardware license (OHL). The software will be open source according to the GNU General Public License (GPL).

Please feel free to sign up for the mailing list and RSS feed at the Web site:

www.delmarnorth.com/microwave/

I serve the team as a coordinator. I have an MSEE in Information Theory from USC, worked at Qualcomm Incorporated for five years as an engineer in the Globalstar and Handset divisions, and am a life member of 10-10 International, ARRL, and AMSAT. I serve as newsletter editor for the Palomar Amateur Radio Club.

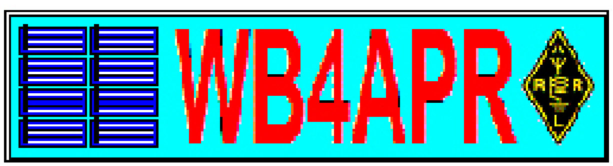
If you know of anyone that might be interested in this project, please forward this invitation. I'm happy to answer any and all questions.

###

RFID for ARRL Ham Radio Nametags and Station Identification/Reporting by DTMF

By Bob Bruninga, WB4APR, bruninga@usna.edu

APRS - RFID is a project to provide all ham radio operators with an RFID name tag such as shown to the right that they can wear during any Ham radio or public service event that will identify them passively as being in the area or in a room or building. Traditionally, most of APRS position and identification depends on GPS, but GPS does not work indoors in most cases. This project will provide a seamless identification of ham radio volunteers where needed. It is less of a "tracking" capability than GPS in the traditional sense, but it is equally valuable in the realm of identification and availability of ham radio resources. This RFID identification process is just an extension of the DTMF identification system added to APRS in 2001.



ARRL RFID tag identifies the wearer whenever he/she passes through a door

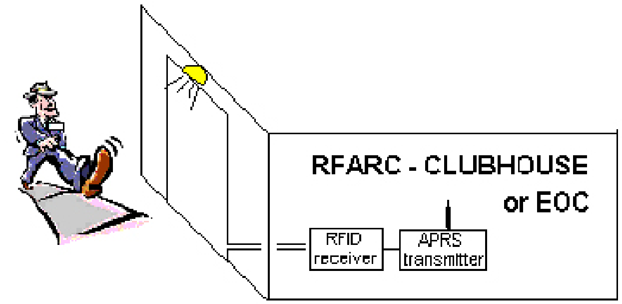
Requirements

The design of such a system would need to consider a number of unique requirements that are best for this Amateur Radio project. Here are some things to consider:

- Kit Format – This should be a great club project
- Inexpensive – The name tags should be under $30 (key chain transmitter) and the reader system under $150
- Detection Range – 100 feet area detector desired, or 5 foot doorway detector minimum
- Call Sign Unique – The device IDs must map as-is to existing call signs or be programmable
- Low Power – Should be passive or able to be powered from a small quarter size battery

Solar Power?

The nametag shown above has enough solar panels to generate about 0.1W in full sun, but more like about 0.01W in ambient room lighting. The device could charge up over several minutes such that when it passed through the sensor-door, it could fire off a 0.1 second data ID burst.



Identification Process

All we need is a call sign. Since the location of the RFID sensor tells us the position data, application, purpose, and event. Notice, that once this CALLSIGN ID has been received, all of the following information can be transmitted in a packet identifying the location of this individual. And in many cases, this is a full set of information for any APRS person on the move...

- His CALL, DATE and TIME are captured
- His POSITION (of the doorway) is plotted with a special RFID icon within a vicinity list of

that door

- Possibly his DIRECTION can be included
- The local FREQUENCY (and tone/shift) identified with that area is included so he can be contacted
- The FUNCTION or EVENT in progress at that location would be included in the position text

This RFID nametag allows every participant at a special event to be located when he passes through various checkpoints or data entry device. See how APRS is used for special event reporting [R1].

Where we use APRS handhelds for entering the scores of scout troops from dozens of checkpoints at scouting events. Just knowing where all the hams are is important in many venues.

RFID Background

This RFID concept dates back to the mid 1990's when we conceived of an APRS system for tracking APRS users inside the buildings at Dayton and other special events. It subsequently evolved to use DTMF and was called APRStt and was initially introduced at the RAC convention in 2001 and subsequently at Dayton in 2002. By the next year, Voice response had been added. This APRStt system demonstrated the ability to let not just the 5% of ham radio operators with APRS, but now 100% of all mobiles and HTs or traveling operators out of their local area to be identified from anywhere in the world when they are operational. Since Kenwood introduced global APRS text-messaging and e-mail into their APRS HT in 1998, APRStt was developed to include the other 95% of ham radio operators by acting as a gateway between their existing DTMF HTs and the rest of the APRS community. This RFID project is just an extension of that to every member of a local club or event. See the APRStt web page [R2]

Katrina Incentive

After Katrina, APRS responded to the ARRL initiative to make sure that every ham radio operator in a disaster area can be located by his frequency (and location). This initiative was called the Automatic Frequency Reporting System [R3]. At that time, the Voice FREQUENCY field was added into the APRS system so that we could find the voice contact frequency for all APRS operators. In addition, not only could APRS users report their operating frequency, but also the wide area Voice repeaters used by travelers could beacon their frequency onto the front panel of all APRS radios.

In response to that initiative, Kenwood developed the D710 mobile radio that not only automatically includes the operators present voice operating frequency in every position report, but also can tune to any other APRS operators frequency or travelers repeater with just the push of the TUNE button. See the LocalInfo initiative web page [R4].

The combination of the original APRS, the addition of Frequency in AFRS, the use of

Voice Repeater objects and the users including their operating frequency in their position reports plus the incorporation of DTMF radio users gives a universal, global, system for identification and localization of all radio amateurs by only their call sign alone.

Any station in the world can be located (or contacted live) through any of the live RF APRS application programs or on the web by any of the WEB based APRS systems such as FINDU.COM as shown in Figure 1.

RFIDs or APRStt Users on the Map

On the APRS map (Figure 1), not only do you see all of the usual APRS operators (about a 32 mile wide map between Washington DC on the left, and Annapolis to the right), but in the lower right, you also see the 147.105 "travelers repeater" along with a simulation of what the DTMF users or RFID users would look like. These non-GPS/ non-APRS users (with the -12 SSID) show on the map as a LIST in the vicinity of the repeater where they were last heard.

Figure 1. Typical APRS map, but notice the DTMF or RFID stations in the lower right shown as being in the vicinity of the 147.105 repeater.

APRStt or RFID Voice Response

In advanced dedicated APRStt nodes (usually with their own frequency), or special RFID applications, these DTMF users can also "visualize" APRS data or receive APRS messages by hearing voice reports from the APRStt voice synthesizer. In effect, the centralized APRStt engine lets everyone in the club or at an event participate in the exchange of APRS data, not just those with the APRS D7 or D700's. Even the old codger that shows up with his venerable 20 year old HT can participate!

The RFID project if taken on as a national ham radio project could revolutionize our ability to find each other. Implementing the DTMF system is as simple as getting some repeater controllers, Echolink, and IRLP nodes, and any other ham radio applications that receive DTMF to accept the DTMF call sign burst and generate the APRS data for transmission over to the APRS network.

But even if we don't get controller manufacturers to add this to their products, we can probably add a DTMF chip to any of the small APRS PIC devices such as the Micro-Track [R5] or the Opentrackers [R6] and design them to be DTMF ==> APRS gateways. They can be either stand-alone devices on any frequency or easily added onto older non-programmable repeater controllers:

Or for special events, or simplex nets, or any place you want to make it easy for non-APRS voice users to be visible to APRS, a simple Opentracker+ with a DTMF chip could go on a

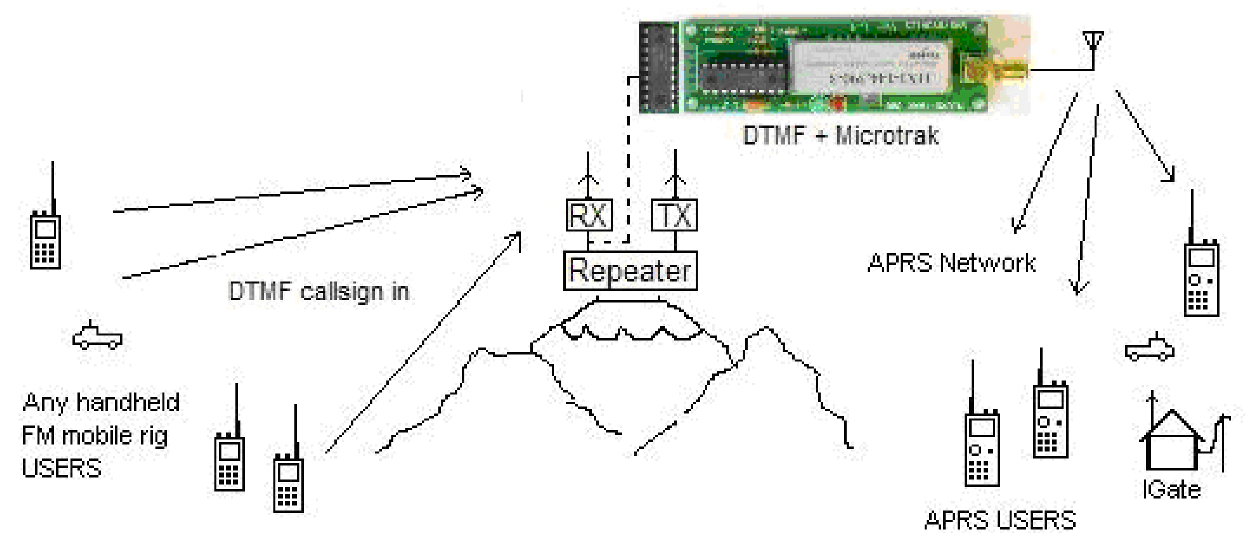


Figure 2. The simple addition of a DTMF decoder chip to a small stand-alone APRS device would make a stand-alone gateway from any repeater between its DTMF users on the voice repeater over to the APRS system. The Microtrack includes its own low power 144.39 RF transmitter.

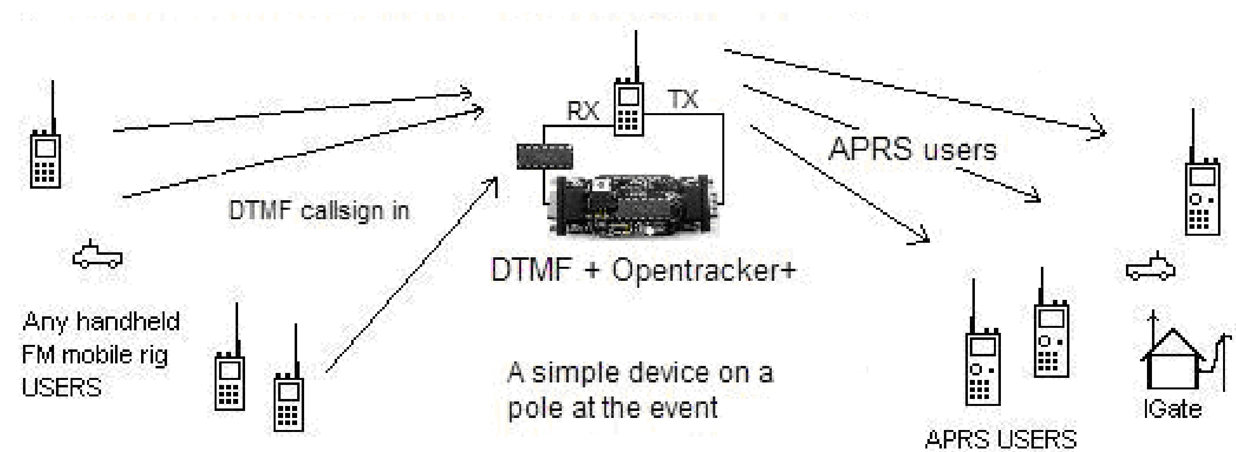


Figure 3. Simple APRStt Stand-alone Gateway for Events - The Opentracker can also make a stand-alone gateway when attached to any FM radio. The RX side listens for DTMF users on the voice channel and the TX side transmits the converted data onto the APRS channel.

dongle and be attached to any HT making that HT be an APRStt gateway for everyone else at that event as shown below.

Or for marathons or any large mass movement of ham radio operators, imagine if you placed one of these autonomous APRStt Gateways at every checkpoint along the Marathon route. As Hams moved about the event, if ever they changed locations from one area to another, all they had to do was send their DTMF call sign memory, then the APRS event map would know approximately where they were and/or what frequency they were monitoring. The APRS symbol for a DTMF user is a gray DTMF keypad. But for future expansion, it can have up to 36

Figure 4. A typical event map showing how those DTMF or even RFID users would be plotted on all APRS maps at the event. The DTMF users are shown as a list in the vicinity of the frequency object on which they were heard. The RFID users would be plotted in the vicinity of the RFID Reader device that picked them up.

overlay characters for special applications. The APRS symbol for an APRStt gateway is a green square with "TT" in the middle of it.

Field Data Entry

Since these same DTMF radio users can also enter small text messages, they can also be used for entering data from checkpoints or other field applications. See how to use an APRS HT for special event data entry [R1].

Or let's say for an event which has maybe 6 operating frequencies, repeaters, simplex channels or whatever. Simply placing one of these APRStt devices on each of those frequencies connected to an HT on a pole would then be able to localize all ham radio participants on the APRS map, at least showing what channel they were presently on as shown above in Figure 4.

In the above view, the "location" of each APRStt receiver is given an arbitrary position so that it shows up conveniently in an out of way place on the APRS event map. Clustered around it are the DTMF call signs that have checked in (by DTMF) on that particular frequency.

APRStt will revolutionize ham radio because it lets 100% of ham radio users "check-in" to the global APRS system to facilitate end-to-end contact between operators. APRS users have been doing this since 1995 or so, but now the other 95% of ham radio operators can participate. The simplest form of APRStt is each user putting his DTMF call sign into a DTMF memory in his radio. If there is an APRStt engine monitoring his favorite repeater, then all he has to do to appear on APRS is to send his DTMF memory! He will appear on the global APRS system as an object within ambiguity range of that voice repeater and showing that frequency!

APRStt Voice Response

On hearing the DTMF call sign and sending out the position and frequency packet on the APRS channel, the more advanced APRStt implementations will respond by voice with "Welcome W3XYZ!" as confirmation. This is really no different than what mobiles do now "WB4APR Listening", except that using the DTMF method is machine readable and allows this presence to the global ham radio community via APRS instead of falling on deaf ears.

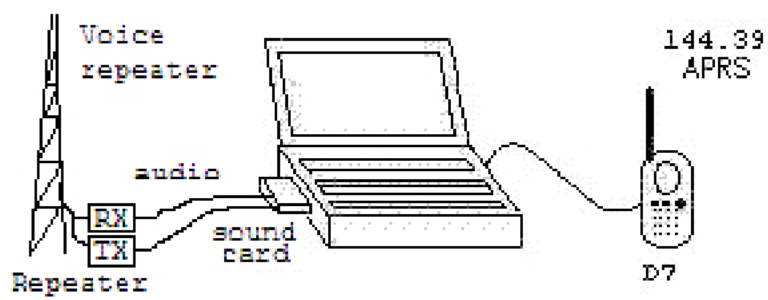


Figure 5. APRStt Repeater Gateway

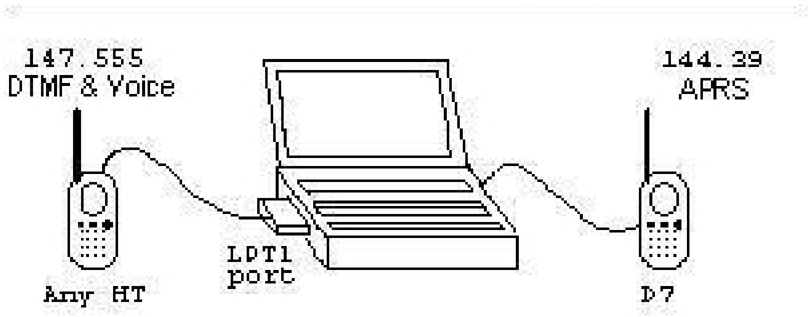


Figure 6. APRStt Portable Gateway

APRStt Two-way Messaging

Now then, these same more sophisticated APRStt systems can also watch the APRS system for any messages addressed to you. Hearing any, they will SPEAK them back to you on the voice channel! The specialized full two-way APRStt Engine can be as simple as a laptop at any repeater site or located in the valley at a home station as shown below.

APRStt will revolutionize how you do special events! Everyone with any HT at any event will be able to keep the overall APRS map and communications picture updated with their position, status and other needed data! Now the Kenwood APRS HTs and Mobiles will have someone to talk with! As long as there is an APRStt gateway nearby, DTMF users can send POSITS, MESSAGES, EMAIL and Queries with their TTone pads and hear VOICE responses (on the APRStt channel). APRStt was demonstrated during Dayton in May and the RAC convention in Vernon B.C. in July 2002.

For field events, the APRStt Engine can be as simple as a laptop and two HTs supporting a special event or field activity as shown to the right. Once this suite is activated at an event, it lets EVERYONE listen to APRS information and input APRS information using any two way radio (usually on a dedicated simplex channel).

APRStt is the gateway for DTMF voice users to report themselves to the global APRS community of users. It enables all non-APRS HTs and Mobile radios to be located and this information is fundamental to facilitating ham radio communications. See some examples:

- Use DTMF to “check in” as noted above
- A check-in indicates the date and time of your immediate availability
- A check-in puts you on the local/ national or global map
- A check-in also identifies your voice frequency to everyone
- Voice response informs the APRStt user of anything he needs to know
- APRStt can speak incoming messages to APRStt users
- APRStt allows DTMF origination of messages, e-mails or anything else in APRS
- APRStt can speak the position (“LEAD is 3.5 miles NW of FINISH”) of special trackers
- APRStt can speak the bearing and range to any object or even satellites that come into view...
- Users can QUERY APRStt with DTMF

about ANYTHING on the air in APRS...

• APRS/APRStt can even tie into the global Echolink system for global voice comms knowing only a call sign! See all about AVRS [R10]

To date, the only radio-only APRS signaling method has been the Kenwood D7 and D700 radios with their internal TNCs. But even so, the user interface is still only the HTs ubiquitous TTone keypad. APRStt simply moves the TT-to-APRS conversion from a TNC in the HT, to a PC on the hill so that ANY existing radio can be used instead of requiring a Processor and TNC to be built into every radio! Further, the DTMF users can receive feedback data via the text-to-speech (or CW) process built into APRStt.

The following APRSdos map of Dayton shows me in the vicinity of the AMSAT Dinner using only the DTMF entry of B47*09. Notice my call within a 1 mile ambiguity circle down where the Amber Rose restaurant is located (and near those APRS mobiles parked in its parking lot!).

The next Dayton map shows me (W4APR-11) located inside the HARA arena at the location of the APRS booth in the North Hall. This

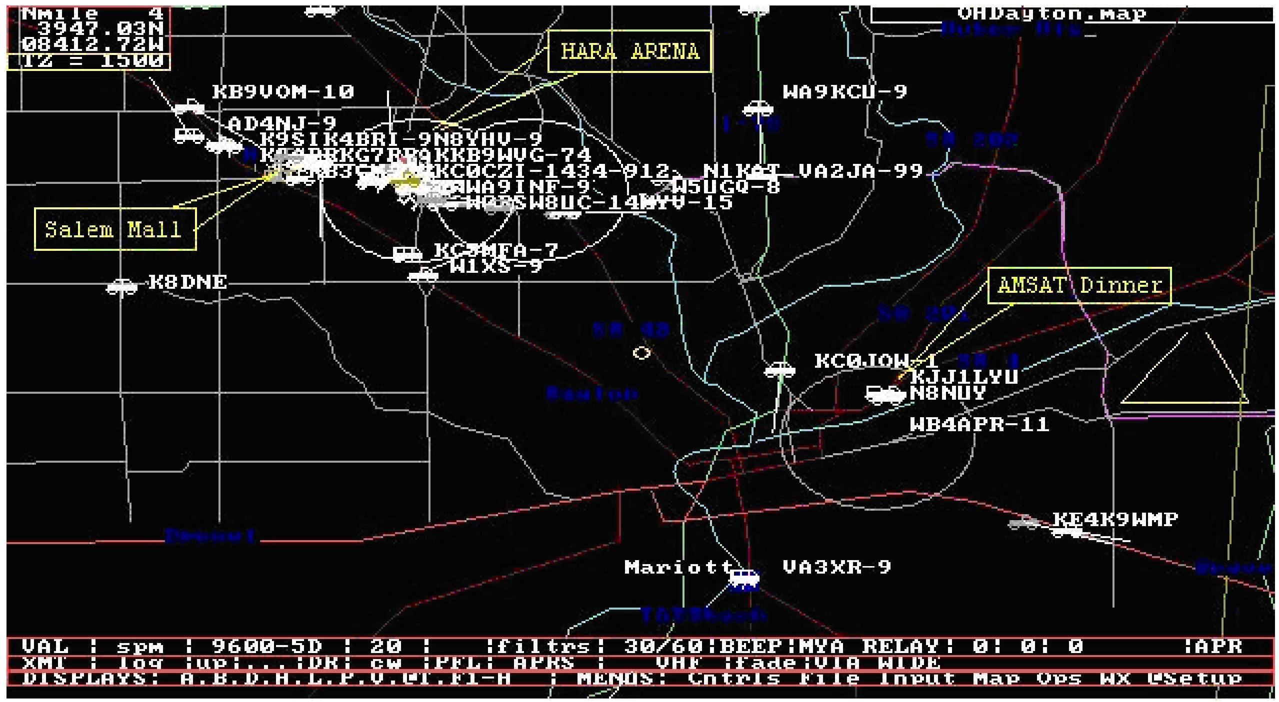


Figure 7. In this figure I was able to report my position within 1 mile vicinity of the Amber Rose restaurant by simply sending my DTMF call sign from DTMF memory and then adding a 4 digit coordinate. This 4-mile range scale APRSdos view shows my 1-mile ambiguou s position in the vicinity of the AMSAT dinner location. This was entered with DTMF using unly eight keys: B46*09D. Coincidentally, there were two other 1-mile ambiguous positions up near Hara at the time.

position is exact because I used the APRStt exact format of 8 digits B4927*1531D.

HOW did I know my LAT/LONG? Easy!. Before Dayton, I copied the MAPS out of the Hamvention Program and added some simple LAT/LONG tick marks along the edges of the map. With one of these in my pocket, I could use DTMF and the 2x2 format to put myself anywhere on the map within 30 miles of the center location of the APRStt receiver. Or using the Hamvention map and the 4x4 APRStt format, I could put myself anywhere exactly. Here is the overall Dayton paper map from the brochure with the one-mile grid tick-marks.

Conclusion

Currently only about 5% of ham radio operators can be located or identified or contacted via the global APRS system, yet probably 95% of them have the capability. Using nothing but our call signs loaded into a single DTMF memory on our radios or HTs we can check-in to the global system and be found. Not only by approximate location, but also by what frequency we are monitoring.

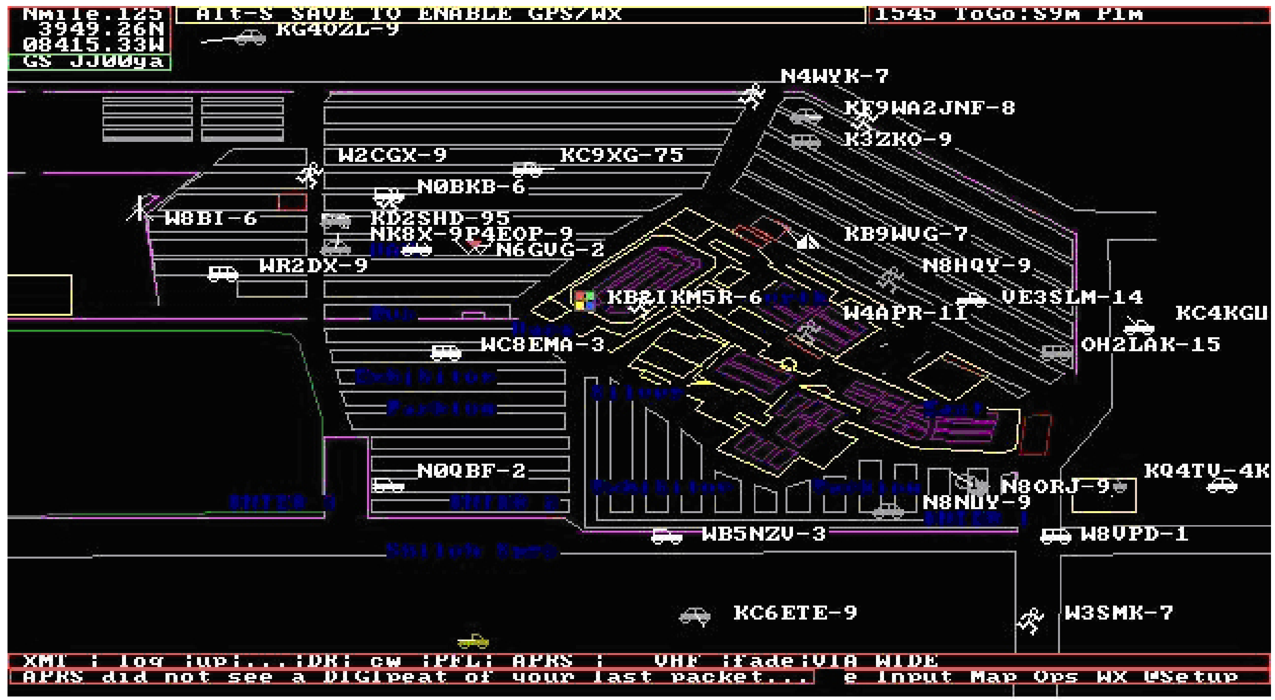


Figure 6. This figure shows me (WB4APR-11) inside the North Hall reported using only a DTMF HT and an 8 digit position coordinate. This map shows the APRSdos view of the Hamvention. Notice "W4APR-11" in the middle of the North Hall, which I entered using the format B4927*1531D using DTMF. Like all other non-message stations, the Icon (human) is shown in gray.

Expanding this system to 100% of ham radio operators via RFID Name Tags at special events or applications, can make ham radio into the truly responsive volunteer system it purports to be.

References

[R1] www.aprs.org/aprsevent.html - Using APRS data (not tracking) for special events

[R2] www.aprs.org/aprstt.html - APRStt web page

[R3] www.aprs.org/afrs.html - Automatic Frequency Reporting System, AFRS

[R4] www.aprs.org/localinfo.html - The APRS LOCALINFO Initiative

[R5] www.byonics.com/microtrak/ - The Byonics Micro-Track

[R6] www.argentdata.com/products/tracker2.html - Argent Data Opentrackers

[R7] www.aprs.org/aprstt.html - The call sign-only APRStt spec

[R8] www.aprs.org/aprstt/aprstt-coding.txt - the APRStt Repeater Spec

[R9] www.aprs.org/aprstt.html - The full APRStt spec

[R10] www.aprs.org/afrs.html - Automatic Frequency Reporting System, AFRS

###

Write Early and Write Often

Packet Status Register (*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 Shakespeare 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 wa1lou@tapr.org.

The deadline for the next issue of *PSR* is October 31, so write early and write often.

###

HF DV Challenge

By Mel Whitten, K0PFX, mel@melwhitten.com

WinDRM and FDMDV are now supporting only free and open codecs. The recent controversy over the use of the "M" word in these DV programs further emphasizes the need for an open source "ham radio" low bit rate codec. There has been talk about it being done, but to this date, I've not seen a plan that will ensure one day a ham codec will become available for HF applications. So what are some of the requirements to get this done? First, the occupied bandwidth available must be considered. No greater than 500Hz BW would give us freedom to move out of the SSB sub-bands where DV is constantly under attack. Narrow BW in combination with FDM could provide the robustness into the sub-zero SNR region. HF voice must provide an "ssb-like" experience (very low latency + strong robustness) to gain wide acceptance in the ham community. Nobody wants to "wait" for sync, need an S7 to avoid dropouts or be completely wiped out by an "adjacent" SSB station.

Obviously, one way to make this happen is to use a very low bit codec. It is possible to have acceptable voice quality at very low bit rates (600bps). Another way may apply a different or new technology not currently in use. Use of a phoneme coder could be a possibility. Is there anyone out there ready to take on the challenge and make this happen? I believe the HF DV community will fully support the efforts of anyone or individuals who would be willing take on this challenge.

Take a listen to the HF Digital Voice recordings here at http://tim-tom1.magix.net/ and then ask yourself, "Do I really want to continue to use SSB on HF?" Please join our DV group at http://groups.google.com/group/digitalvoice and share your ideas that may someday bring HF DV into the mainstream of Amateur Radio.

###

Digital Speech Within 100-Hz Bandwidth

By Mike Lebo, N6IEF, mike-lebo@ieee.org

Objective

To get help developing software needed to achieve extremely narrow band digital speech, which could be received by Amateur Radio operators much better then present analog speech.

Why Do This?

To communicate over a distance the receiver must hear and understand what the sender said. When using radio, it is necessary to overcome the path losses. The two traditional ways to overcome these losses are power gain and antenna gain. With the use of up-to-date computers a third way to get gain is processing. The bandwidth of voice is about 2400 Hz. If that could be reduced to 100 Hz, the gain would be 13.8 dB (24X). A power amplifier for that gain would cost over $5k. Increasing the antenna gain would cost over $20k. However processing gain by computer is free. This project receives weak signals 10 dB (10X) below noise of the radio.

Overview of Hardware

Voice enters a computer through a microphone connected to the microphone-input of the computer's sound card when the space bar of the keyboard is held down. The computer processes the voice and sends it out as a Quadrature Phased Shift Keying (QPSK) audio tone from the line-output of the sound card to the radio, just like PSK-31. (NOTE: For a laptop it is necessary to get a USB to audio line input/output device.) The radio makes a Radio Frequency (RF) QPSK signal that is sent out through the antenna. Another antenna, radio and computer at a different location receive this RF signal and convert the QPSK RF into a QPSK audio tone. Although the output of the radios speaker may sound like noise, the QPSK audio tone is still there. The signal from the radio is connected to the line-input of the sound card of the computer, where it is processed and the speech is played on the computer speaker.

Steps Needed for Computer Processing During Transmit

1. The digitized audio from the microphone can be compared to a list of 45 digitized audio clips called phonemes. A phoneme is to speech as the alphabet is to reading or writing.

2. Once a phoneme is identified, a unique coded sequence makes a QPSK audio tone, which is sent from the computer sound card.

3. If the audio level drops below a set threshold, a code of 100 is repeatedly sent.

Steps Needed for Computer Processing During Receive

1. The audio from the radio receiver is sent to the line input of the computer and is digitized through a process like PSK-31 software (www.moetronix.com/ae4jy/winpsk.htm). A waterfall is display on the computer monitor. Through a computer made digital filter, the bandwidth of the radio receiver is reduced from 2400 Hz to 100 Hz.

2. The QPSK audio tone is detected and converted into a code, which is then compared to the table of audio clips in the computer. 3. The audio clip is stretched in time to fit the code and played on the computer speaker until the next audio clip begins.

Generations of Transmit Phonemes

Since each person sounds different from another, it is clear that the computer must recognize the unique phonemes used by only that person while operating this software. The software must be able to teach itself the phonemes so that it can recognize that person's voice.

1. To set up the initial phonemes the person reads words shown on the monitor into the microphone while holding down the space bar of the keyboard.

2. Each word is repeated until the computer learns that phoneme. This repetition accomplishes two different things. First, it calibrates the automatic speech compression algorithm so that it constantly adjusts the microphone gain for fixed output level. I believe the variations of voice amplitude add very little to the context of speech and those variations are not sent or played at the receiver. Second, parts of these spoken words are used to generate the 45 digitized audio clips.

Code Used to Send and Receive QPSK Tone

The 45 phonemes are represented by a code made up of 1's and 0's. The code is similar to a court recorder typing out steno, which could be read back. All code groups start with 1 and end with two or more 0's. Since phonemes are grouped by the shape of the mouth, the codes used in one group of phonemes should be as different as possible from other groups. Example, "b" sounds like "v" and their codes should be similar. But they sound very different from "m" which should have a much different code. Some phonemes are longer than others and they should have a longer code. Others are short like the sound of the letter "t", which should have a short code. Of the 53 codes, only 45 are used with eight as spares. This code is exactly the same Varicode used in PSK-31.

100 1100 10100 11100 101100

111100	1010100	1011100
1101100	1110100	1111100
10101100	10110100	10111100
11010100	11011100	11101100
11110100	11111100	101010100
101011100	101101100	101110100
101111100	110101100	110110100
110111100	111010100	111011100
111101100	111110100	111111100
1010101100	1010110100	1010111100
1011010100	1011011100	1011101100
1011110100	1011111100	1101010100
1101011100	1101101100	1101110100
1101111100	1110101100	1110110100
1110111100	1111010100	1111011100
1111101100	1111110100	1111111100

As shown, the code is the fastest speed for each phoneme. By adding one or more extra 0's to any code, the length of that phoneme is stretched by increments of 1/100 of a second. This is very important because voice speed is constantly changing. The original 45 phonemes are expanded to many new phonemes.

100-Hz Clock

The 100-Hz clock is made exactly the same way as the 63 Hz clock of PSK-63.

Receiver Phonemes

Since each person has a unique set of phonemes, there is no way for the computer at the radio receiver to know how to make received code sound like the original person. So to make this process more fun, 12 different sets of audio clips will be available at the receiver by selecting F1 through F12 on the keyboard. These might range from the sound of a little girl to that of an old man and anything in-between.

Improvements After System is Working

One of the problems in selecting the code for the transmit phonemes is the error contributed by the background noise picked up by the microphone. To reduce this noise, two microphones should be used, one with the person's voice plus the background noise and the other with just the background noise. The computer can easily subtract the background noise. Computer sound cards are equipped for stereo, so that should be no problem.

The computer at the ham radio transmitter learns exactly what the phonemes are used by the person sending the transmission. This set of phonemes and the sending station's call sign can be e-mailed to the software at the receiver's computer. When the computer at the ham radio receiver detects the code for the phrase "This is (followed by the transmitting call sign)", it automatically switches to the e-mailed set of phonemes for that call sign. That way the sound from the speaker of the computer at the ham radio receiver sounds like the voice of the person doing the transmitting. This would be useful for three-way conversations or nets.

The eight unused phonemes are added for other languages.

Implementation of the System

Since this can be used all over the world, the software must be available on-line for free.

###

Development Plan for Digital Speech Within 100-Hz Bandwidth Software

By Mike Lebo, N6IEF, mike-lebo@ieee.org

Software Summary

1. This software makes 13.8 dB of compression gain (X 24) by reducing the SSB (Single Side Band) receiver's bandwidth from 2400 Hz to 100 Hz while keeping the same power.

2. Voice through the computer's microphone is converted into numbers, amplified to a constant level, converted into 16 bands of frequency, cut into three parallel 30 ms sections of time, compared in a two-stage process to a library of 45 phonemes that have been made by the operator of the radio, converted to a digital code, stretched to fit the operator's real speech, and sent to the radio in a way similar to PSK-31 (Phase Shift Keying with 31 Hz bandwidth) or even more similar to QPSK-63 (Quadrature Phase Shift Keying with 63 Hz bandwidth) to be transmitted.

3. The other radio receives a digital signal similar to QPSK-63.

4. The audio from the other radio's speaker goes into the computer where the software shows the waterfall and spectrum response on the computer's monitor just like PSK-31.

5. The code is detected and compared to one of twelve person's phoneme sounds selected by the operator and played as speech on the computer's speaker.

Modification of WinPSK Program

This software is modified from the QPSK-63 software. I have chosen to use Visual C++6.0 as the programming language because it is easier to modify software than to create it. Moe Wheatley, ae4jy, has done an outstanding job on his open source WinPSK program and his documentation of the software. Please read the PSKCore.DLL (Dynamic-Link Library) Software Specification and Technical Guide at www.moetronix.com/ae4jy/winpsk.htm. Before installing this QPSK-100 (Quadrature Phase Shift Keying with 100 Hz bandwidth) software, make sure your radio, interface and computer are working by testing the WinPSK program with PSK-31 over-the-air.

The modifications to the WinPSK program are the following:

1 The data to the modified WinPSK program is a series of ones and zeros at a 100 Hz clock rate from the transmit sequence in place of the typing from the keyboard.

2 The data from the modified WinPSK program is a series of ones and zeros at a 100 Hz clock rate to the receive sequence and played as speech over the computer speaker in place of the text showed on the computer monitor.

3 Holding down or releasing the space bar on the computer keyboard is the same as tapping the TX/RX icon or F12 key of the WinPSK program.

4 All received signals start out as BPSK (Bi-Phase Shift Keying) and are automatically changed to QPSK once BPSK ends and the squelch is released.

5 The transmit text and the receive text displays on the monitor of the WinPSK program are deleted, but the spectrum and waterfall are expanded and are always displayed.

6 Most of the icons are removed from the display of the WinPSK program, except for the icons from RX Freq to Net and the Spectrum icons.

Transmit Sequence

The transmit sequence starts with the pressing of the space bar on the computer keyboard and continues until the space bar is released. The computer speakers D/A (Digital to Analog) converter is force to zero. The AGC (Automatic Gain Control) is unfrozen.

The 400 ms synchronizing alternating series of ones and zeros is sent to the transmit section of the WinPSK program. This 100 Hz BPSK code is used by the other computer's receiver section of the WinPSK program to re-synchronize the 100 Hz clock, which insures that the other computers receiver section of the WinPSK program is sampled in the middle of each code digit and is not sampling during the transitions.

The sampling 66,000 Hz clock starts the A/D (Analog to Digital) converter from the microphone input of the computer. Each clock cycle makes the A/D output a 16-digit signed number. Each number goes to the AGC (Automatic Gain Control) array and the AGC level adjustor.

The AGC is used to amplify the weak signal from the microphone to about 90% of the maximum value for the 16-digit signed number. This is done by TBD (To Be Determined) method. It will use the normal fast attack and slow decay, but it will be frozen when the space bar is not pressed.

Some of the numbers from the AGC level adjustor go to 32 FIR (Finite Impulse Response) low-pass filters. An FIR low-pass filter has a frequency F and a number of taps N and a sampling rate. The problem with filters is the time difference, DPD (Differential Propagation Delay), between the outputs of high frequency filters and the outputs of low frequency filters with the same input to both. The 17 F frequencies for the FIR filters are 8000 Hz, 6083 Hz, 4625 Hz, 3517 Hz, 2674 Hz, 2033 Hz, 1546 Hz, 1176 Hz, 894 Hz, 680 Hz, 517 Hz, 393 Hz, 299 Hz, 227 Hz, 173 Hz, 131 Hz, and 100 Hz.

A first order attempt to solve the DPD problem is to use different sampling frequencies for each FIR filter. The numbers from the A/D are at a 66,000 Hz rate. If every fourth number is used, the new sampling rate is 16,500 Hz, or 66,000 Hz divided by 4 is 16,500 Hz. The 16 divide-by numbers are 4, 5, 7, 9, 12, 16, 21, 28, 36, 48, 63, 82, 110, 145, 190, and 251.

The divided-by-4 sampling rate is used by the two highest frequency FIR low-pass filters. Both FIR low-pass filters need to have the same number of taps N to insure that their output numbers are available at the same time, or zero DPD. By subtracting the two FIR low-pass filter outputs, numbers are created at the sampling rate of the FIR low-pass filters. These numbers are approximately the instantaneous amplitude of the sound between the two frequencies. In the same way the other 15 frequency bands of numbers are made, with associated sampling rates. Each set of two FIR low-pass filters has the same sampling rate, and taps N, and their DPD is zero, so their output numbers can be subtracted.

The DPD between frequency bands is not zero, but this doesn't matter because the numbers between frequency bands are never used together.

A phoneme is to speech as the alphabet is to reading or writing. Some people say that there are 44 phonemes and one extra for no sound. Dividing the A/D sample clock rate of 66,000 Hz by 1980 makes the phoneme sample interval. This interval is 30 ms. After the start of the phoneme sample interval, the next available 14 numbers from each of the 16 frequency bands are examined for the largest or smallest value. If the smallest number is selected, it is made positive by multiplying by negative one. This is called the peak search process. Just before the end of the interval, say at count 1979 of 1980, the 16 peak numbers are put into the phoneme sample array. The phoneme sample forms a 16 by 1 array. This process re-synchronizes the DPD problem to the original 66,000 Hz sample clock of the microphone input D/A.

In order not to miss a phoneme, this procedure is repeated in parallel, two other times by starting at counts 660 and 1320 from the original 1 to 1980. This insures a new phoneme sample array every 10 ms. The 30 ms time interval is used to detect each of the 45 phonemes, even if the phoneme lasts longer. To reduce the chances of receiving part of one phoneme and part of another phoneme, a new set of 16 peak numbers is started every 660 numbers or 10 ms. Overlapping numbers insure that a phoneme is not missed.

One of three phoneme comparators takes its phoneme sample array and compares it to one of 45 arrays from the phoneme library by subtracting one array from the other array. One way to visualize the phoneme sample array is to think of it as the silhouette (sill-a-wet) of a city or in the case of no-sound, the silhouette of a flat lake. Then each of the differences is multiplied by itself to make them all positive numbers. Then all 16 numbers are added together to make the single number error value. In the same way the next array from the phoneme library is used until all 45 arrays from the phoneme library are done. The phoneme code for the smallest, the second smallest and the third smallest error number of the 45 possible error numbers is sent to the guesser along with their error number and code size. Although this process takes some time, the output rate should be the same as the input rate of 30 ms. Since there are three peak detectors with three comparators staggered 10 ms apart, a phoneme code with its error number and code size is sent into the guesser each 10 ms. The code size is a number from three to ten, which is the number of ones and zeros in that phoneme code.

The guesser is used to determine what code should be sent to the output Q. The guesser is like a Q with three levels. Three phoneme codes and their error numbers enter the back of the guesser and work their way down to the front of the guesser. So there are always nine phoneme codes in the guesser. Whenever three codes are entered, three other codes are removed. If there are three of the same phoneme codes in the guesser, the error number of that phoneme code in the front of the guesser is divided by three. If there are two of the same phoneme codes in the guesser, the error number of that phoneme code in the front of the guesser is divided by two. After the divides, the phoneme code and the code size of the smallest error number of the three in the front of the guesser is sent to the output Q. This happens ever 10 ms.

The output Q is used to fix problems that happen when one phoneme transitions to another phoneme in our speech. The output Q is used to sort the phoneme codes into groups, like sorting cards into suits. If the phoneme code sent to the back of the output Q is the same as any of the two previous phoneme codes in the output Q, the new phoneme code is moved forward to that same phoneme code group.

One phoneme code is removed from the front of the output Q as each digit of the phoneme code is sent to the transmit part of the WinPSK program. But before a new phoneme code group is sent to the transmit part of the WinPSK program, the number of phoneme codes in that group is checked to see if they are less than the minimum number for that code size. If they are less than the minimum number, the group is deleted from the output Q.

An extra zero is sent to the transmit part of the WinPSK program as each extra phoneme code beyond the phoneme code size is removed from the output Q. No extra zeros are sent to the special phoneme code of 100, but the code could repeat if needed.

If the Q does not contain enough of the phoneme codes, each digit of the code is still sent to the transmit part of the WinPSK program, but the Q does not move to the next phoneme code until all the digits of that code are sent.

Code Size	Minimum Number
3	2
4	2
5	3
6	4
7	4
8	5
9	6
10	7

At the start of each transmission sequence the guesser and output buffer are filled with a quantity of the 100 special codes for no-sound, because the computer takes some time for the numbers from the microphone A/D to be processed. These leading 100 special codes are deleted from the output Q and the ones and zeros of the rest of the real phoneme codes are sent to the transmit part of the WinPSK program.

Each digit of the phoneme code is sent serially at a 10 ms rate. This is the same rate which the error numbers enter the guesser and the same rate which the radio outputs each QPSK RF (Radio Frequency) audio tone.

At the end of each transmission, the space bar on the computer keyboard is released, all 100 special phoneme codes on the back of the output Q are deleted and the special phoneme code of ten ones in a row is sent to the output Q and then to the transmit part of the WinPSK program. This sets the squelch of the other computers receiver section of the WinPSK program.

With today's computers having 4 GHz clocks and quad processors, twelve billion operations can be done every second. (Speech recognition software in 2004 did not have this computer power and did not work very well.) In the event the guesser makes a mistake, the occasional anomalous sound from the receiver is dealt with by our brains. Words may sound mispronounced, but we should know what they mean.

This transmit sequence may look like speech recognition software, but it has two differences. First, speech-to-text software requires the ability to handle spelling and meaning. An example would be the homonyms "to," "two," and "too." Most of the speech recognition software would not be used. Second, speech recognition software has no time limit from sound to print. The transmit sequence of this software requires a minimum fixed time delay.

So let's review. The time interval for the no-sound code is from 30 ms to the end of transmission in 30 ms steps. The time interval for the largest phoneme code is 100 ms to the end of transmission in 10 ms steps. The fastest average phoneme rate is about 15 phonemes per second. This may sound like a lot, but the word "at" has three phonemes. There must be a no-sound phoneme between the beginning and the ending of the word "at". Adding zeros expands the original 45 phonemes to a very large number of phonemes needed for speech spoken at any speed. The amplitude changes are not transmitted. The error correcting coding of the modified WinPSK program reduces received errors without adding to the bandwidth or reducing the sending speed. The transmit power is 100%, so don't burn up your final. The bandwidth is 100 Hz. There is a half-second delay for re-synchronization on each transmission, so don't expect an instant reply to your message.

Receiver Sequence

The receiver sequence starts with the release of the space bar on the computer keyboard and continues until the space bar is pressed. The microphone A/D is forced to zero. The guesser is not allowed to send more codes to the output buffer.

After the 400 ms BPSK signal re-synchronized the 100 Hz clock and releases the squelch, the ones and zeros coming from the receive part of the WinPSK program are sent to the phoneme comparator. The first one after two consecutive zeros starts a new phoneme code. The first code of ones and zeros assumes a 100 special code for no-sound has been detected. Since the phoneme code is sent serially, each digit goes to the phoneme code library one at a time where half of the library is eliminated with each digit after the first one. When the next digit is received, half of the half of the library is eliminated and so on until two consecutive zeros are detected. That is when the phoneme code is found. Then four phoneme arrays (audio clips) are found from the phoneme library. The first phoneme array is called the main array. It is ((the code size – 2) X 10 ms) long and has ((the code size – 2) X 660) numbers. The next phoneme array is called the zero array. It is 10 ms long and has 660 numbers. The next phoneme array is called the third array. It is the same as the zero array, but each of the numbers has been divided by three. The last phoneme array is called the two-third array. It is the same as the third array, but each of the numbers has been multiplied by two.

Normally a .wav file would be used for an audio clip, but that won't work for 10 ms to 80 ms sound clips with 660 to 5280 numbers in each array. A new way to send the numbers to the speaker D/A will be made by TBD a method.

When the first two consecutive zeros of the present phoneme code are detected, each of the numbers in the present third array and each of the numbers in the previous two-third array are added in the first blender array. Then each of the numbers in the present two-third array and each of the numbers in the previous third array are added in the second blender array. Then the first blender array is sent to the sound card D/A buffer of the computer, followed by second blender, followed by the main array of the present phoneme code. If another zero is detected after the first two zeros of the phoneme code, the zero array of the present phoneme code is sent to the sound card D/A buffer for each extra zero.

The two 10 ms blender arrays are used to ease the transition from one phoneme to the next phoneme when played on the computer's speaker.

Then the next detected phoneme code is sent to the sound card D/A buffer and so forth. The sampling rate for the D/A is 66,000 Hz because 66,000 Hz was used to make the original phoneme code arrays in the look-up library. Although this example uses one set of phoneme code arrays for each phoneme code, the computer contains 11 other sets of phoneme code arrays, which can be selected by the operator pressing one of the F1 through F12 keys on the computer keyboard.

So let's review. The 100 Hz clock is re-synchronized. The receivers bandwidth is 100 Hz, which is 13.8 dB (24X) better then the 2400 Hz for SSB voice. The code is used to find the sound clips in the library. When played at the computer, the sound clips are blended together.

Making the Operator's Phonemes Sequence

Before doing the transmit sequence the phoneme library arrays must be known. This is a once in a lifetime event, which must be done before the computer is connected to the radio. The best way to do this is TBD, but the following method can be used. The operator says words into the microphone that are displayed on the computer monitor, while holding down the space bar on the keyboard. These words are chosen for a particular phoneme. When the sound is examined with an audio spectrum analyzer, the phoneme can be seen. Since we don't have this device, another method is needed to get the numbers for the phoneme. One method would be to use the audio spectrum analyzer to look at the total time it takes to say the word and assign a start-and-stop percentage of that word for the phoneme of interest. It is easy to look at the numbers in the array for the spoken word and find the start-and-stop time. This is when the amplitude goes above or below a minimum value. No matter who says the word, the percentage of the word that is the phoneme remains the same.

The same microphone and A/D converter from the transmit section are used to make the numbers of the phoneme, which are then applied to the same FIR filters. After the start of the phoneme sample interval, the next available 14 numbers from each of the 16 frequency bands are examined for the largest or smallest value. If the smallest number is selected, it is made positive by multiplying by negative one. This is the same peak search process as in the transmit section. Just before the end of the interval, say at count 1979 of 1980, the 16 peak numbers are put into the phoneme sample array. The phoneme sample array becomes the library value for that phoneme. But this library value might be wrong. So the word should be repeated and averaged. When the change in the average is small, then there is enough information to use the array. This needs to be done for all 44 phonemes. The no-sound phoneme is the only exception. No testing is required. Any DPD problems are exactly the same in both the transmit sequence and the making operator's phonemes sequence, which negate each other.

Making the Library Sequence at the Distribution

One more thing is needed before distribution of the software. The main, zero, third and two-third arrays used in the library of the receive section need to be made. Twelve different people should record the 44 phonemes. This will be done in the lab with audio spectrum analyzers and high tech computers. Each of the numbers in an array must start and end at zero crossing and have a positive slope at each start and a negative slope at each end. This is to prevent discontinuities when any two sets of numbers are connected then played into the computer speaker. After the main phoneme arrays are made, the zero arrays are made. This could be done in the lab by changing individual numbers in the zero array for best sound when connected and played on the computer's speaker. The third array and the two-third array are easy to do.

Software Milestones

1. Change QPSK-63 in the WinPSK program

2. to QPSK-100 and test this over-the-air.

3. Change codes to make the special code of 100 a no-op rather than sending the alternating series of ones and zeros between typed characters of PSK-31 and test this over-the-air.

4. TBD

###

Packet Status Register

#106 Summer 2008, ISSN: 1052-3626

Published by

TAPR
phone 972-671-TAPR (8277)
fax: 972-671-8716
e-mail tapr@tapr.org
URL www.tapr.org
TAPR Office Hours
Monday – Friday, 9 AM – 5 PM Central Time

Opinions expressed are those of the authors and not necessarily those of TAPR, the TAPR Board of Directors, TAPR Officers, or the Editor. Acceptance of advertising does not constitute endorsement by TAPR, of the products advertised.

Postmaster: Send address changes to TAPR, P. O. Box 852754, Richardson, TX 75085-2754. *Packet Status Register* is published quarterly by TAPR. Membership in TAPR, which supports the electronic publication of the *Packet Status Register*, is $20.00 per year payable in US funds.

TAPR is a community that provides leadership and resources to radio amateurs for the purpose of advancing the radio art.

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 (wa1lou@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 (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* version 5 and distributed electronically at www.tapr.org

PSR Packet Status Register Editor:

Stan Horzepa, WA1LOU
One Glen Avenue, Wolcott, CT 06716-1442 USA
phone 203-879-1348
e-mail wa1lou@tapr.org

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.