

VOTER SYSTEM

*An Open-Source/Hardware VOIP-Based Voting Multi-Receiver and Simulcast Transmit System
Jim Dixon, WB6NIL – July 24, 2011*

INTRODUCTION

In many two-way radio applications, both for repeater systems and simplex base-stations, it is often difficult to have reliable reception when there is signal impairment due to terrain or other obstacles.

One common way to solve this problem is by use of a voting multi-receiver system. Such a system is comprised of a number of receivers, located at diverse locations. The location of these receivers is chosen so that the combined coverage of all the receivers more or less fill-in the entire desired coverage area, even though it may take a number of receivers to accomplish this. All such receivers are connected ("linked") to a central location (generally the transmitter site), and a device compares each receiver's signal strength and selects the one with the best signal. This is called voting.

Often in such a system, there is one single high-powered transmitter located at a central site that all mobile/portable stations would be able to receive.

Sometimes, one single transmitter can not be located where coverage of the entire desired area is possible. In such cases, multiple transmitters are deployed in several locations. Since all these transmitters are on the same frequency, they must be very precisely locked to the same frequency and be transmitting the exact same audio at the exact same time. Not doing so would severely degrade their reception.

Traditionally, receivers that comprise a voting system are each connected to the central site via either a UHF or microwave link. Since this requires a receiver, link transmitter and link receiver for every receive site, implementation and continued cost of such a system typically becomes quite significant and in most cases prohibitive, and requires a high complexity of implementation.

There are, however, many advantages of implementation of such a system that warrant the cost and effort required.

Recently, with the advent of VOIP -based interconnections for radio systems, it is now possible to implement a voter system using VOIP to link the receivers to the central site, and therefore reduce the system requirements and complexity and cost by eliminating the radio links between the receivers and the central site.

There are a few commercial vendors of VOIP-based voting systems that offer a functional and reliable, yet proprietary and extremely high-cost solution.

Although using VOIP-based technology significantly reduces the cost and complexity of implementation of a voting system, the cost of the commercial VOIP-based devices is still quite prohibitive for all but large commercial and/or government applications.

Therefore, it seemed appropriate to offer a completely open-source, open-hardware solution that makes the cost and complexity of a VOIP-based voting system nearly trivial and gives almost anyone the ability to implement this type of a system that wishes to do so.

DESCRIPTION

In order to successfully implement a VOIP-based solution for this purpose, it is necessary to have a precise time reference in order to be able to re-assemble (time-wise) the audio streams at the central site, since VOIP has inherent inconsistencies in time delay (jitter and such). Clearly, the best and most reasonable and economic source of such precise timing data is GPS. Therefore, a GPS receiver (with a 1 pulse per second output) must be made available at each receiver location (along with the central site).

Since precise timing is necessary, it precludes the use of any implementation which relies on a traditionally host-based operating system, such as Linux or UNIX, because of limitations of timing and scheduling precision and latency.

Therefore, a dedicated hardware device with a simple embedded micro-controller was necessary to accomplish the necessary precise timing requirements. In addition, low cost, in comparison with a host-based solution makes such a simple dedicated hardware device at each receiver location gives this solution a great advantage.

Such a dedicated hardware solution significantly increases reliability, has a far lower power consumption, and is physically smaller (in comparison with a host-based solution).

The "central site" functionality (the device which receives all of the VOIP-based audio streams from the receivers and selects which receiver's audio to use) is implemented as a channel driver (chan_voter) along with app_rpt under Asterisk PBX. The host-based system on which this runs, along with a dedicated hardware device (which gives access to the precise GPS-derived time data) is typically located with the system's transmitter (or at least one of them).

Each dedicated hardware device is also capable of simultaneously generating time-synchronized audio out to drive a transmitter (or multiple transmitters in a simulcast system). The audio out of all the dedicated hardware devices at all locations will be identical at same time.

There are facilities available for on line monitoring of the voting system (audio and signal strengths), in addition to recording (logging) facilities.

IMPLEMENTATION

A UDP-based VOIP protocol was created specifically for this technology.

The dedicated hardware device is initially implemented as an approx 160 by 100 mm circuit board utilizing thru-hole mounted components allowing user-assemble-able blank boards to be made available quite inexpensively. It uses a dsPIC33FJ128GP802 processor and a few other integrated circuits, including some operational amplifiers (which implement filters, amps, etc). The schematics, board design, artwork, and PIC firmware are available on a completely free open-source basis.

Full documentation for the above-mentioned protocol, in addition to the schematics and photographs of and artwork for the board are in other documents included in this package.

Please see <http://voipvoter.org> for full documentation and links to all the sources, files, documents and applications relating to this project.

PROVISIONING AND CONFIGURATION

GENERAL SYSTEM PROVISIONING

There needs to be:

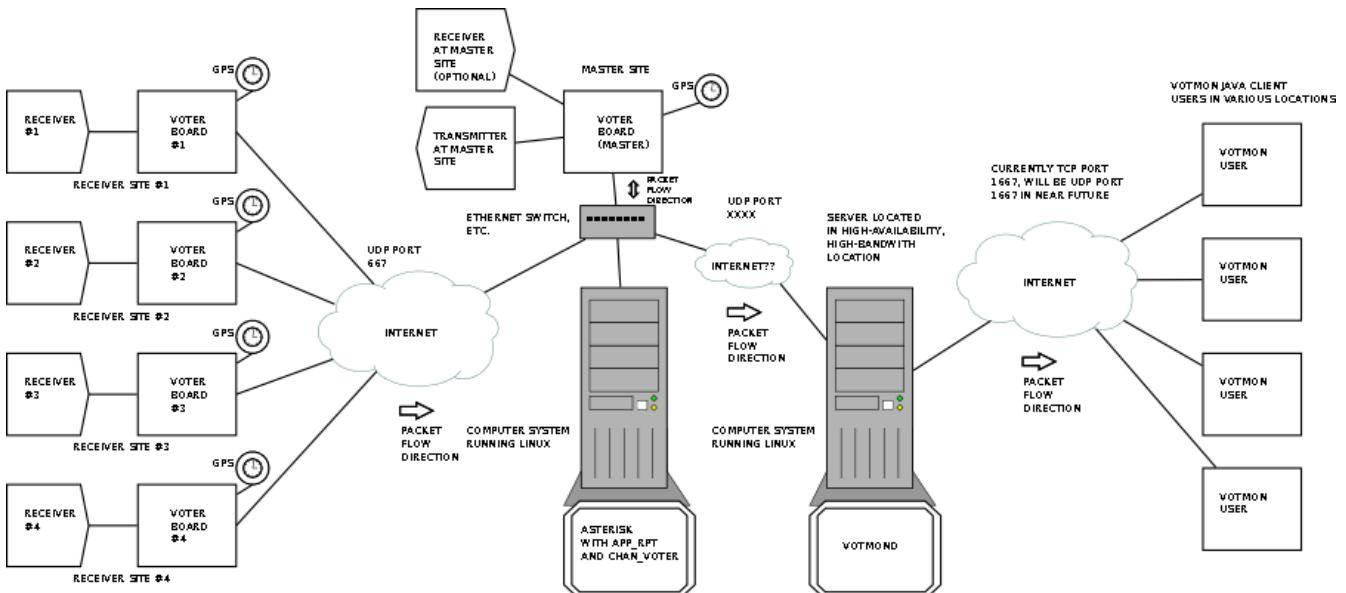
Some number (> 1) of receiver sites that contain 1 of identical model receivers (or at least ones with identical characteristics) and 1 VOTER (embedded hardware) card , 1 GPS receiver and some sort of Internet connection (can be a private LAN).

Some number of transmitter sites (typically 1, but not necessarily) that contain a transmitter, 1 VOTER (embedded hardware) card, 1 GPS receiver, and some sort of Internet connection (can be a private LAN).

(A site may have both a receiver and transmitter sharing the same VOTER card, GPS receiver, and Internet connection).

A Linux host running Asterisk with app_rpt and chan_voter, along with a VOTER (embedded hardware) card, a GPS receiver, and an Internet connection (may be a private LAN in some cases) with sufficient bandwidth to accommodate all the packet traffic from/to all the receiver and/or transmitter sites. This location may also have a receiver and/or transmitter connected to the VOTER card.

Optionally, system monitoring access (both audio and signal strength at all receive sites) may be made available through the use of a JAVA applet and a process running on a server at a location with high availability and high bandwidth. Typically, this is not at the same location as the above system, but may be if appropriate.



EXAMPLE OF TYPICAL 4 (OR 5) RECEIVER VOTER SYSTEM,
ALSO SHOWING 4 ACTIVE VOTERMON USERS

IP (INTERNET) NETWORK PLANNING AND PROVISIONING

The receiver/transmitter sites may be connected to the "master" site (the one containing the Linux system running chan_voter/app_rpt/Asterisk) via either a LAN (either physical or extended Local Area Network), or via a public Internet connection, or some combination thereof.

Regardless of the interconnection method, the following criteria must be met:

Sufficient bandwidth must be available at all times between each receiver/transmitter site and the "master" site. The latency of such connections must be within reasonable limits, and certainly must be within the limits configured in both the VOTER board devices and chan_voter.

The "master site" really should be on a static IP address. Although not 100% necessary, it could really make for an unreliable installation.

Any of the devices (including the "master" site) may be behind NAT and/or a firewall. There is no specific requirement for the VOTER board to have any sort of non-NATed access, since the actual IP address and UDP port that is public really makes no difference. All authentication is based upon CHAP-type tokens, and not upon IP address in any way. The "master" site may be behind NAT and/or a firewall, as long as its listening UDP port (generally 667) is exposed to the public (unless entire system is on an entirely private network). If the implementor wishes, telnet access may be configured for the VOTER board, and in that case, the port selected for telnet must be made publicly accessible (since the VOTER firmware support dynamic dns, it certainly may be on a dynamically-assigned public IP address when the telnet option is used, and its actual IP address really doesn't matter if telnet is not configured).

If the JAVA client for monitoring is used, the server to which the clients connect must be on a fixed IP address (obviously.. A high-availability server on a dynamic IP address would be rather silly!).

Typically, the IP bandwidth usage in each direction, when a signal is present is approx 100 kbps (including UDP/IP overhead, 1 direction only if just a receive or just a transmitter site, or both directions if a receive/transmit site).

The "trunk" (or streaming) connection between the "master" site and the JAVA client distribution site (if used) uses approx 200 kbps of bandwidth when a signal is present (1 direction only).

Each JAVA client requires approx 100 kbps for monitoring.

On a public Internet connection, its a good rule-of-thumb to get a connection capable of 3 to 4 times the required bandwidth in the slowest direction (often, such connections are asymmetrical as far as their speeds are concerned).

If possible, avoid wireless connections. DSL is your friend. It may be slow, but it tends to be more reliable and consistent than other forms of broadband connections.

If a wireless connection is really required, use a high reliability solution, such as the devices from Ubiquiti, which are very high-quality, reliable, and inexpensive.

RECEIVER (AND/OR TRANSMITTER) SITE

A receiver and/or transmitter site consists of a receiver capable of providing direct access to the output of the FM discriminator (and output of CTCSS/CDCSS tone decoder, if applicable) and/or a transmitter, a VOTER (embedded hardware) board, a GPS receiver capable of providing a precise 1 pulse per second signal (such as the Garmin LVC18), and an Ethernet-based Internet connection.



VOTER board shown installed on top of a Motorola Radius Receiver with GPS receiver and associated cabling



Detail of user-assemble-able VOTER board

Since there may very well be several versions of the VOTER board hardware available, this document will not cover specific details of the shown hardware device here. The full details for each hardware device, including construction (if applicable), programming, and specific configuration options are available from the vendor of the particular device. Only generic configuration concepts and options will be covered here. The specific information on the user-assemble-able board, shown here, is included at the end of this document.

The board has connectors for the radio (receiver and/or transmitter) and power, Ethernet, GPS receiver (optionally RS-232 console) and programming adapter (also provides hardware reset access). It has several LED's potentiometers and jumpers.

Here is an example of the connectors available on the user-assemble-able board:

J1 - 6 pin MTA-100 connector for PIC programmer (PICKit3 or equiv). Also reset switch may be placed across pins 1 and 3.

J2 - Ethernet (10-Base-T).

J3 - RS-232 (console @57600 bps, DCE)

J4 - 5 pin MTA-100 connector for GPS (see below for pin-out).

P1 - Radio Connector (9 pin D-shell Male) (see below for pin-out).

Radio Connector (9 Pin D-Shell Female) Pin-out:

1 - + VIn (6.5-30 Volts DC).

2 - Transmit Audio Out

3 - Receive (discriminator) Audio In

4 - External CTCSS Input (optional)

5 - Gnd

6 - Gnd

7 - PTT Out

8 - Gnd

9 - Gnd

GPS Connector (5 Pin MTA-100 Male) Pin-out:

1 - Gnd

2 - 5 Volts DC Out (to power GPS device)

3 - GPS TXD (Gnd, actually)

4 - GPS RXD (in)

5 - GPS PPS (in)

Here is an example of the LED's, jumpers and potentiometers available on the user-assemble-able board:

LED designations (shown from right to left along front of VOTER board) ::

LED1 - (Green) System activity indicator (should toggle every 500ms)

LED5 - (Red) Transmit indicator

LED2 - (Green) Receive Signal indicator (solid green is valid Rx signal, flashing is signal without CTCSS)

LED3 - (Yellow) GPS indicator (solid is GPS received and locked, flashing is GPS received, lock in progress)

LED4 - (Yellow) Connection indicator (indicates that connection was made with VOTER host)

JUMPERS:

JP1 - Insert if low discriminator level. If squelch cannot self-calibrate with JP1 removed (too low), try with JP1 inserted.

JP2 - Insert to attenuate discriminator input level by 20db.

JP3 - Selects power source for output AMP. 1-2 is to power it from the 5VDC power supply. 2-3 is to power it directly from VIn.

JP7 - Remove for programming.

JP8 - "Initialize configuration parameters in EEPROM". If JP8 is inserted when firmware starts, the operating parameters in the EEPROM will be set to default values. The system activity LED (LED1, green) will stay off for approx 4 seconds, then stay on steady to indicate that the initialization process is complete. Afterwards, the jumper may be removed and the system will continue running as usual.

Note, if JP10 is inserted during this procedure, the "Diode Calibration" process will also occur.

JP9 - Insert to calibrate squelch. With the receiver connected and its antenna removed, Insert JP9. In the next few seconds the "Receive Signal Indicator" (LED2, green) will flash on and off, then (hopefully) on steady. This indicates that the squelch calibration has occurred successfully. If unsuccessful, the LED will flash either fast to indicate that the discriminator noise level is too high, or slowly to indicate that the discriminator noise level is too low. Note, if JP10 is inserted during this procedure, the "Diode Calibration" process will also occur.

JP10 - Insert this jumper to perform "Diode Calibration". This may only be done in conjunction with a configuration parameter initialization (see JP8, above), or a "Squelch Calibration" (see JP9, above).

JP11 - Insert this jumper to temporarily re-purpose LED3 and LED4 to allow for visual indication of RX input level

POTENTIOMETERS:

R22 - Squelch adjustment

R36 - Rx Input Level

R61 - Tx Output Level

VOTER BOARD CONFIGURATION SETTINGS

There are several categories of configuration parameters that apply to all version of the VOTER card.

IP SETTINGS – Standard IP setup parameters (IP Address, Netmask, Gateway, DHCP, etc).

VOTER SERVER – Voter server address, specified as a Fully-Qualified Domain Name (FQDN), UDP port on VOTER server, and (optionally) local UDP port (generally left at default), and client and server passwords.

The firmware in the VOTER board does a DNS resolution once per minute and resolves the IP address for the VOTER server, based upon the FQDN specified in this configuration. Therefore, the actual IP address of the VOTER server may be changed (and the its associated DNS entry), and the VOTER board firmware will continue to be aware of its current IP address (thus not requiring re-configuration of the VOTER board firmware if the IP address of the VOTER server changes).

The “client password” is the password, as configured in the server, that identifies the VOTER board (client), and the “server password” is the common server password as configured in the server (which allows verification of the server's identity by the VOTER board firmware).

TX BUFFER PARAMETER – Transmit buffer length. This allows for simultaneous time-consistent audio output at all VOTER boards in a system. Buffer length is set to allow for maximum network latency between “master site” and receive/transmit sites. .

GPS RECEIVER PARAMETERS – Data Protocol (NMEA or TSIP), Baud Rate, Data Polarity and PPS Signal Polarity (and time offset, don't ask, just leave it at 0, hopefully) for GPS receiver.

TELNET PARAMETERS – Port Number, User name, and Password for Console Telnet Access.

DYNDNS PARAMETERS – Setup parameters necessary to make dynamic DNS work (if desired).

EXTERNAL TONE DECODE – Sets mode for external tone decode input (enable/polarity).

The console gives access to a menu allowing setting of the above-mentioned parameters in addition to various system-related functions, reboot, save parameters, status, etc.

In addition, the console displays important system status messages as various events occur (such as changes in GPS status, changes in DNS resolution, etc).

VOTER BOARD SETUP AND INSTALLATION

Appropriate connections need to be made between the receiver's discriminator and optionally its tone decode output and/or the transmitter and its line level (or MIC input, attenuated) and PTT signals.

The GPS receiver needs to be connected, and the board needs to be connected via Ethernet to an Internet connection, and, of course, power needs to be applied to the board.

With no antenna connected to the receiver, the boards needs to have its “squelch calibration” procedure performed (see the documentation for the specific hardware).

Then the receive level needs to be set, either by putting the board in “level setting” mode, or with use of the console (see the documentation for the specific hardware).

Once the above procedures are performed, the board is ready to have its operating parameters configured and then it will be ready for use.

Set the IP Setting, TX Buffer Parameters, GPS Parameters (if necessary), Telnet Parameters, DYNDNS Parameters, and External Tone Decode appropriately for the installation.

The VOTER board needs to be on an Ethernet segment (typically a router or switch or wireless bridging device of some sort) that has connectivity to the “master” site, either via public Internet or via private LAN (extended, perhaps). It may be on a network that allocates IP addresses (LAN and/or public IP address) dynamically. The VOTER protocol and the VOTER board firmware are designed specifically to allow for not needing the IP address of any VOTER board to be long-term consistent, or even important, for that matter. All authentication and identification is based upon tokens, **not** IP address or port.

If telnet access is desired, the VOTER board needs to at least have its telnet port publicly accessible (through a NAT/Firewall if used), otherwise it is not necessary. Even if telnet access is used, a dynamically-assigned public IP address will work, because dynamic DNS (dyndns.org) is supported in the VOTER board firmware.

This document assumes sufficient familiarity with IP and general networking concepts, and therefore will not go into specific details of network provisioning.

The VOTER Server Settings on the VOTER board will be covered in the next section.

OVERALL SYSTEM / MAIN SITE CONFIGURATION

app_rpt/Asterisk uses a channel driver “*chan_voter*”, which implements the “master site” functionality, providing an Asterisk channel which app_rpt may open as a radio interface. This driver receives (and/or sends) UDP packets (typically on UDP port 667) from/to all associated VOTER boards at receive and/or transmit sites.

In addition, the channel driver (optionally) may be configured to send a stream of UDP packets to a predetermined IP address and UDP port. This stream is meant to be redistributed by the *votmond* program running on a high-availability, high-bandwidth Linux Server, and give users access to monitoring of the VOTER audio and signal strength information in real time.

Chan_voter uses a configuration file, */etc/asterisk/voter.conf*, which contains typically several stanzas.

Here is an example of a *voter.conf* file:

```
[general]
port = 667
buflen = 500
password = BLAH

[1234]
MAD6 = madcow6
MAD5 = madcow5
MAD4 = madcow4
MAD3 = madcow3
MAD2 = madcow2
MAD1 = madcow1,master
streams = 67.215.233.178:1667
plfilter = y
txctcss = 100.0
txctcsslevel = 100
txtoctype = none
thresholds = 255,110=5
linger=6
```

The “*general*” stanza specifies the UDP port that the VOTER boards are expected to send packets to (the “*port*” parameter), the receive buffer size in milliseconds (the “*buflen*” parameter), and the SERVER password (the “*password*” parameter).

Each additional stanza is named in an appropriate manner (generally, the associated node number in the *rpt.conf* file). This stanza name references the channel name (in this case “*Voter/1234*”) that is used when specifying the channel within the Asterisk system.

Within each named stanza (not the “*general*” stanza), the VOTER board clients are defined (by specifying *DISPLAY_NAME* = *password[,options]* for each client), and operating parameters, such as “*streams*”, which specifies a list of *IP_ADDR:UDP_PORT* values indicating where to send the votmond redistribution stream associated with the specified system, “*plfilter*”, which, if set to true, enables a 300Hz low pass filter, used when a CTCSS/CDCSS tone is higher in frequency than 120 hertz, “*txctcss*”, “*txctcsslevel*”, and “*txtoctype*” (if transmit CTCSS is desired), and “*thresholds*” and “*linger*” which define the behavior of the voting selection algorithm.

The “options” for each specified *DISPLAY_NAME* are as follows:

- master* – This specifies that this client is the Master Timing source (the VOTER board that is on the same LAN as the Asterisk server. There can **only** be 1 Master Timing source per entire Asterisk server.
- transmit* – This specifies that this client is intended to have transmit audio sent to it and will have a transmitter connected to it.
- adpcm* – This specifies that this client is to be sent audio in ADPCM format, rather than Mu law.

If no options are specified for a client, then none of the options specified above are enabled for the particular client, and will be treated as a standard (non-master), receive-only, Mu law audio client.

On each VOTER board, the “*Client Password*” must be set to the associated value for the desired client as specified in the stanza of the *voter.conf* file. Using the above example, the “MAD2” VOTER board would be configured to have a “*Client Password*” of “*madcow2*”. The “*MAD2*” designation is only used and referenced by *chan_voter*, and is not specified or known in any way by the VOTER board.

On each VOTER board, the “*Server Password*” must be set to the value specified in the “*password*” parameter in the “*general*” stanza of the *voter.conf* file.

The “*buflen*” parameter in the “*general*” stanza specifies the receiver buffer length in milliseconds. This parameter should be set to the maximum expected network latency, plus a little padding (100 milliseconds of padding is a good amount). The default is 500 milliseconds, and is what the value will be set to if not specified.

The Asterisk server at the “main site” (in addition to all the ports that are canonically made public to operate a standard *app_rpt* installation) **must** have the port specified in the “*port*” parameter in the “*general*” stanza in the *voter.conf* file made publicly available, or at least available to all possible IP addresses that its associated VOTER clients could possibly be o

The “*Tx Buffer Length*” parameter on each VOTER client, is similar to the “*buflen*” parameter for the receive side. It's set the transmit buffer length in samples (8000 samples per second, so 8 samples = 1 millisecond). Like the “*buflen*” parameter, this should be set to the maximum expected network latency plus a little padding (800 samples is a good amount of padding, so setting the parameter to 3000. is probably a good place to start).

In both the receive and transmit cases, the buffer length is a fixed value. This will hard-set the delay time between actual reception and presentation time to the Asterisk channel (in the case of receive), and presentation from the Asterisk channel and actual transmission (in the case of transmit). Since the overall system time delay (the amount of time between reception and re-transmission of the signal) is determined as the sum of the receiver buffer length, the (very minimal) latency within app_rpt/Asterisk and the Transmit buffer length, it is very necessary to keep these settings to a reasonable minimum (without sacrificing packet consistency/integrity by making it too short based upon network latency/jitter).

If transmit CTCSS is desired, the “*txctcss*” and “*txctcsslevel* ” (and optionally the “*txtotype*”) parameters must be specified. In order to facilitate generation and transmission of CTCSS along with the normal system audio, the VOTER board (and host) generate “flat” (pre-emphasized/limited) audio, which is intended to directly modulate a transmitter (to be applied at a direct modulation point, NOT a line level or microphone input).The “*txctcss*” parameter specifies the transmit CTCSS frequency in hertz.

The “*txctcsslevel*” specifies the level of the CTCSS tone (0-250). Optionally, the “*txtotype*” parameter may be specified, which determines the “turn-off” (when transmitter un-keys) style for the CTCSS. Setting this parameter to “*none*” (which is default) means that the transmitter immediately stops transmitting when the system “un-keys”. Setting this parameter to “*phase*” means that the CTCSS tone will briefly be transmitted 120 degrees out of phase when the system “un-keys” (also known as “reverse-burst”, etc). Setting this parameter to “*notone*” means the transmitter will continue transmitting for a brief period with no CTCSS after the system “un-keys” (also known as “chicken-burst”).

If transmit CTCSS is **not** desired, **DO NOT** specify **ANY** of the “*txctcss*”, “*txctcsslevel* ” or “*txtotype*” parameters. The audio that is produced by the VOTER board (and the host) will be normal line-level (intended for a line or microphone input) and will NOT be pre-emphasized.

Since token-based authentication and identification is utilized, having the UDP port open to the public should not be a significant security risk.

There **must** be a VOTER board on the **same LAN** (very low latency) as the Asterisk server implementing the “master site”, which acts as the Master Timing source. This allows chan_voter to have a consistent, reliable, accurate timing source with which the timing information from all other inbound packets are compared and appropriately processed, and from which to generate accurate timing information for time-consistent transmission purposes.

VOTING SELECTION ALGORITHM AND ASSOCIATED PARAMETERS

Voting selection (the choice of client from which to take the audio stream) is based upon the RSSI (relative signal strength/quality) value associated with each particular client's received signal.. The parameters associated with voting selection are the “*thresholds*” and “*linger*” parameters specified for each VOTER client.

If **neither** of these parameters are specified, the voting selection algorithm will default to choosing the client with the highest RSSI number (and the last-most one listed in the *voter.conf* file, if a tie exists) each and every received audio frame (every 20 milliseconds). Although this is a functional option, **this is probably NOT what you really want**. Doing so will cause the received signal to have an audible “squelch-tail” at the end of a transmission if there is a receiver receiving the signal strongly in addition to a receiver receiving the signal at a low, noisy level. This, however might be appropriate in very rare instances where receivers are placed in locations that require very agile following of the strongest signal. It is very unlikely, however, that this is the case.

It is far more likely that it is appropriate to set up a set of “*thresholds*” and optionally a “*linger*” time that is different than the default (which is 6 frame periods of 20 milliseconds).

The “*linger*” parameter allows specification of a default “*linger*” value (in 20 millisecond frames) other than the system default value (6).

The “*thresholds*” parameter is a comma-separated list of RSSI threshold values each specified as follows:

```
MIN_RSSI[=REASSESS_FRAMES[:LINGER_FRAMES]]
```

MIN_RSSI is the minimum RSSI value for this threshold (1-255)

REASSESS_FRAMES is the number of 20ms frames for the client to remain selected at this threshold level before being re-assessed (may be specified as 0, meaning “re-assess at next frame”).

LINGER_FRAMES is the number of 20ms frames for the client to remain selected after no longer being at this threshold (or any other) (may be specified as 0, meaning “do not linger”)

If MIN_RSSI is specified without the other values, its REASSESS_FRAMES is considered to be infinite (it will not get re-assessed if its RSSI value remains at this threshold), and LINGER_FRAMES will be its default value (6).

If MIN_RSSI and REASSESS_FRAMES is specified, LINGER_FRAMES will be at its default value(6).

A specification for REASSESS_FRAMES **must** be present if there is a specification for LINGER_FRAMES.

For example:

thresholds = 255,110=5

Meaning:

If the RSSI level is 255, stay selected on that client as long as it still has an RSSI of 255, else if the RSSI level is at least 110, stay selected there for no more than 5 frames. If the RSSI drops below 110, re-assess every frame (since no value below 110 was specified).

Also, since there were threshold values specified and the “linger” value remained default, any client that meets any specified threshold, stays selected for 6 frames when no clients meet any thresholds any longer.

If, instead the example was:

thresholds = 255,110=5:10

Then:

The same would be true except that if a client meets the “110” threshold (and not the “255” threshold), and then no more clients meet any more thresholds, the client that met the “110” threshold would stay selected for 10 frames rather than the default 6.

This whole methodology is rather strange, but it seems to be appropriate and effective. In the future, it may be improved or modified in some way, but for now it is more than sufficient.

VOTERMON (JAVA MONITOR CLIENT) AND VOTMOND SERVER CONFIGURATION

A JAVA Client (applet) known as “*votermon*” is available, allowing real-time monitoring of the VOTER audio stream and relative signal strengths of all receivers in a system, as well as the result of the “voting” selection.

It is a JAVA applet to be embedded into a web page served by an HTTP/HTTPS Web server. Based upon parameters specified on the web page in which it is embedded, it will connect to a server running the “*votmond*” program.

This program takes a UDP stream from a running instance of chan_voter (as specified in its stanza's “streams” parameter), and make it available for users of the JAVA applet to connect.

The server should be at a high-availability, high-bandwidth location, and have a fixed IP address. It must be configured to allow public access to whatever TCP port the “*votmond*” program is set to listen to client connections on, and minimally, access from the IP address of the “main site” system on which the instance of chan_voter is running.

SUMMARY

This technology/project is a great asset to the two-way radio community in general, particularly Amateur Radio and other public-service related radio services, allowing inexpensive, general, and open access to what previously would have been impossible or otherwise unattainable.

Systems of this type can greatly improve ability to provide efficient and reliable communications, not to mention promoting usage of frequencies and modes that otherwise may have been underutilized or ignored.

This document and information contained herein (including the use of it, and that which is derived from the use of it), and all other intellectual properties contained herein, and all intellectual property rights have been and shall continue to be expressly for the benefit of all mankind, and are perpetually placed in the public domain, and may be used, copied, and/or modified by anyone, in any manner, for any legal purpose, without restriction.

VOTER Protocol – Voice Observing Time Extension (for) Radio

Jim Dixon, WB6NIL

Version 0.7 – Jul. 24, 2011

VOTER is a completely connectionless UDP-based protocol (using IANA-assigned well-known port 667, originally assigned to me for a project I was doing for the United States Federal Election Commission, which, of course, deals mainly with an entirely different kind of voting). It provides for transmission of audio signals from remote radio receivers, along with ultra-precise (GPS-based) timing information and signal quality, allowing synchronization of multiple signals at the "head" end (hereinafter referred to as the "host") of the system and selection of which signal to use based upon the signal quality information and the consistency thereof. Audio and timing information may also optionally be sent back the remote location (hereinafter referred to as the "client") by the host to facilitate a multiple-site simulcast transmission system.

Security is provided by a Challenge/Response authentication technique using CRC-32 based digest information. Although it is by no means a terribly secure method (it is not even close to cryptographically-strong), it at least it provides somewhat of a deterrent for those attempting to circumvent the security.

Both the client and the host use the same exact messages as follows:

The VOTER protocol packet consists of a 24 octet header and optionally a payload, which currently can either be 160 samples (20ms) of Mu-law encoded audio, 320 samples (40ms) of ADPCM encoded audio, or GPS location and elevation information (sent once per second).

The VOTER protocol packet header consists of 24 octets as follows:

All date and time information is in GMT.

All multi-byte numeric fields are sent standard network-order (most significant byte first).

Octets 0-3: Current accurate time in whole seconds, 32 bits (network order).
Octets 4-7: Current accurate fractional time in nanoseconds, 32 bits (network order).
Octets 8-17: Authentication challenge, ASCII string, null-terminated (up to 9 chars long + null).
Octets 18-21: Authentication response (digest), 32 bits (network order)..
Octets 22-23: Payload type, 16 bits (network order).

Payload types are as follows (header octets 22-23):

Payload type 0 – Authentication plus flags only if sent by host. Client does not send payload.

Octet 24: Flag bits, as follows (specified in bit value):

1 – Flat Audio (NOT de-emphasized) if flag set, Processed audio (de-emphasized) if not set

- 2 - Send audio always, regardless of whether or not signal is detected if set, do not send audio unless signal is detected..if not set.
- 4 – Do Not filter Sub-Audible tones from audio stream, if set. Filter Sub-Audible (<300Hz) tones from audio stream if set.
- 8 – Master Timing Source Mode. If not set, payload 1,2 and 3 packets are delayed (approx 6 ms) to guarantee that the packets from the device designated as "Master Timing Source" are received by the host previous to packets from any other device. If set, no delay is performed, and the packets are sent immediately. This must be set on only one device in the network.
- 16 – Send audio in IMA ADPCM (Intel/DVI block format) rather than Mu-law.
(32-128) – RFU

Payload type 1 – RSSI information + Mu-law audio. Octet 24 is RSSI value 0-255 and Octets 25-184 are 160 samples of Mu-law encoded audio (20ms @ 8K samples/s). Packets of this payload type are sent only when the receiver is receiving a valid signal (or the host is indication that a signal is to be transmitted).

Payload type 2 – GPS Information as follows:

Octets 24-32: Longitude value, in the format XXXX.DDY (eg 4807.038N), ASCII string, null-terminated.

Octets 33-42: Latitude value, in the format XXXXX.DDY (eg 01131.000E), ASCII string, null-terminated.

Octets 43-49: Elevation value in Meters (eg 545.4), ASCII string, null-terminated.

Payload type 3 – RSSI information + IMA ADPCM audio in Intel/DVI block format.

. Octet 24 is RSSI value 0-255 and Octets 25-187 are 163 octets (320 samples) of IMA ADPCM (Intel/DVI block format) encoded audio (40ms @ 8K samples/s). Packets of this payload type are sent only when the receiver is receiving a valid signal (or the host is indication that a signal is to be transmitted).

By definition, a digest value of 0 is an indication that the entity generating it has not received a validly recognizable digest from its peer. Therefore, a challenge must not be generated that would result in a digest value of 0. Also, a "Master Timing Source" must not use ADPCM audio.

The intended methodology for the use of the protocol is as follows:

Then the client starts operating, it starts periodically sending VOTER packets to the host containing a payload value of 0, the client's challenge string and an authentication response (digest) of 0 (to indicate that it has not as of yet gotten a valid response from the host). When such a packet is received by the host, it responds with a packet containing a payload value of 0, the host's challenge string, an authentication response (digest) based upon the calculated CRC-32 value of the challenge string it received from the client and the host's password., and option flags. If the client approves of the host's response, it may then start sending packets with a payload type of 1 or 3, the client's challenge string, and an authentication response (digest) based upon the calculated CRC-32 value of the challenge string it received from the host and the client's password, along with the 1 byte of RSSI information and 160 bytes of mu-law audio (and continues doing this every 20ms) as long as there is signal being received by the receiver.. If the host approves of the client's response, it then uses the audio and/or GPS information provided by the client (and the flow continues).

At any point, if either side does not approve of its peers authentication response, it must reply to its peer with a packet with a payload type of 0 (indicating that authentication need to take place).

The connection-less nature and the flexibility of this protocol allow the client to continue operating even of the host stops or changes its challenge string in mid-stream.

Basically, when a client starts up, it just starts sending out packets, whether or not the host is there, or responding or responding positively. Once it receives valid authentication information from the host, it then is to send periodic GPS packets (once every second), and audio packets (once every 20/40ms) when there is audio to be sent (a signal is present on the receiver, or there is a signal to transmit). If, at any point the client gets responses from the host that do no contain valid authentication, it merely goes back to sending authentication packets once again, and the whole process repeats. Similarly, any time the host receives a packet from the client that does not contain valid authentication, it sends an authentication packet to the client.

Therefore, the protocol is completely tolerant of interruptions in network connectivity, restarts of either or both sides of the connection, and accomplishes this quite easily and with very little complication and overhead.

In summary, there are 5 cases of packets:

- 24 octets, payload 0, digest 0 -- This is only sent by a client
- 24 octets, payload 0, digest (calculated) – This is only sent by a client
- 25 octets, payload 0, digest (calculated) ,flags – This is only sent by a host
- 185 octets, payload 1, digest (calculated)
- 50 octets, payload 2, digest (calculated) – This is only sent by client
- 188 octets, payload 3, digest (calculated)

A host will not send a packet to a client until the client has send a packet for the host. Therefore the "24 octets, payload 0, digest 0" packets will only be sent by a client (since a digest of 0 indicates not yet hearing from a peer). A client may send as many payload 0 packets as is appropriate (there really is no limit). A host must respond to each of these packets.

Presence of payload 1 and/or 3 packets indicates valid signal received or to be transmitted.

For a "receive-only" implementation, the host may only send payload 0 packets in response to packets from the client. It need not send any payload 1 or 3 packets if doing so is inappropriate. A client need not keep intrinsic track of date information (such as a real-time clock, etc). The client may, if no other means is available, determine current date (not including time of day) from the time information in the authentication packet last received from the host. Therefore, the host must send time information that has an accurate date (accurate time is not required).

This document and information contained herein (including the use of it, and that which is derived from the use of it), and all other intellectual properties contained herein, and all intellectual property rights have been and shall continue to be expressly for the benefit of all mankind, and are perpetually placed in the public domain, and may be used, copied, and/or modified by anyone, in any manner, for any legal purpose, without restriction.

APPENDIX B

Installation & Utilization information for VOTER board 07/06/2011 User-Assemble-Able VOTER BOARD

Installation of firmware on board:

You will need to have a Windows PC loaded with the MICROCHIP MPLAB IDE software
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodId=1406&dDocName=en019469
and download the "MPLAB IDE v8.66" (at the end of that page).

Also, you will need the MICROCHIP MPLAB C30 Compiler
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodId=1406&dDocName=en010065&part=SW006012
and download the "MPLAB C Compiler for PIC24 and dsPIC v3.25 in LITE mode" (at the end of the page).

You will need the MICROCHIP PICkit3 programmer device (PG164130).

Download the "voter-bootloader" files at <http://svn.ohnosec.org/viewvc/projects/allstar/voter/voter-bootloader/?root=svn>

Open the voter-bootloader project in MPLAB IDE.

The default (hard-coded) IP address for the bootloader firmware is: 192.168.1.11.
If this doesn't work for you, please refer to the following directions to change it:

Select View->Program Memory (from the top menu bar).

Hit Control-F (to "find") and search for the digits "00A8C0".

These should be found at memory address "03018".

The "A8C0" at 03018 represents the hex digits C0 (192) and A8 (168)
which are the first two octets of the IP address. The six digits to
enter are 00 then the SECOND octet of the IP address in hex then the
FIRST octet of the IP address in hex.

The "0B01" at 0301A represents the hex digits 0B (11) and 01 (1)
which are the second two octets of the IP address. The six digits to
enter are 00 then the FOURTH octet of the IP address in hex then the
THIRD octet of the IP address in hex.

Remove JP7 on the VOTER Board. This is necessary to allow programming by the PICKit3 device.

Attach the PICKit3 device to J1 on the VOTER board. Note that Pin 1 is closest to the power supply
modules (as indicated on the board).

If you have not already selected a programming device,
Select Programmer->Select Device and choose PICKit3

Then Select Programmer->Program. This will program the bootloder firmware into the PIC device on
the board.

Replace JP7.

Download the "EBLEX C30 Programmer" files from
<http://svn.ohnosec.org/viewvc/projects/allstar/voter/EBLEX%20C30%20Programmer/?root=svn>
and the "Voter.cry" file from
<http://svn.ohnosec.org/viewvc/projects/allstar/voter/board-firmware/Voter.cry?root=svn&view=co>

Put the "EBLEX C30 Programmer" in some reasonable directory, as this is the location from which the program will be executed. If the IP address of the VOTER board bootloader is not the default address (192.168.1.11), you will need to edit the "ENC_Loader.INI" file to reflect the correct IP address.

Run the "ENC_loader". Open the "Voter.cry" file. Reset the VOTER board. QUICKLY (within a couple of seconds), press the "Capture Target" button on the ""EBLEX C30 Programmer" program. Once captured, press the "Program" button. Once programmed, press the "Reset LIA" button. The VOTER board should restart, and after about 5 seconds, the green activity light should start flashing, etc.

Once you have verified that the firmware is running, place a jumper shunt on JP8 and JP10. Be sure to do this at a reasonable human room temperature (like 60-80 deg. F). This is necessary because this procedure initializes the operating parameters in the EEPROM and also does a temperature-compensation diode calibration which needs to happen at room temperature. Once the jumper shunts have been installed, power-cycle the board and wait for the system activity light to come on steady (about 4 seconds or so after the 5 second initial wait time). Then remove the jumper shunts.

Usage of VOTER board:

Connectors:

J1 - 6 pin MTA-100 connector for PIC programmer (PICKit3 or equiv). Also reset switch may be placed across pins 1 and 3.

J2 - Ethernet (10-Base-T).

J3 - RS-232 (console @57600 bps, DCE)

J4 - 5 pin MTA-100 connector for GPS (see below for pinout).

P1 - Radio Connector (9 pin D-shell Male) (see below for pinout).

Radio Connector (9 Pin D-Shell Female) Pinout:

1 - + VIn (6.5-30 Volts DC).

2 - Transmit Audio Out

3 - Receive (discriminator) Audio In

4 - External CTCSS Input (optional)

5 - Gnd

6 - Gnd

7 - PTT Out

8 - Gnd

9 - Gnd

GPS Connector (5 Pin MTA-100 Male) Pinout:

- 1 - Gnd
- 2 - 5 Volts DC Out (to power GPS device)
- 3 - GPS TXD (Gnd, actually)
- 4 - GPS RXD (in)
- 5 - GPS PPS (in)

LED designations (shown from right to left along front of VOTER board) :

- LED1 - (Green) System activity indicator (should toggle every 500ms)
- LED5 - (Red) Transmit indicator
- LED2 - (Green) Receive Signal indicator (solid green is valid Rx signal, flashing is signal without CTCSS)
- LED3 - (Yellow) GPS indicator (solid is GPS received and locked, flashing is GPS received, lock in progress)
- LED4 - (Yellow) Connection indicator (indicates that connection was made with VOTER host)

JUMPERS

JP1 - Insert if low discriminator level. If squelch cannot self-calibrate with JP1 removed (too low), try with JP1 inserted.

JP2 - Insert to attenuate discriminator input level by 20db.

JP3 - Selects power source for output AMP. 1-2 is to power it from the 5VDC power supply. 2-3 is to power it directly from Vin. DO *NOT* CHANGE THIS JUMPER WITH POWER ON. DAMAGE MAY OCCUR IF YOU DO!!

JP7 - Remove for programming.

JP8 - "Initialize configuration parameters in EEPROM". If JP8 is inserted when firmware starts, the operating parameters in the EEPROM will be set to default values. The system activity LED (LED1, green) will stay off for approx. 4 seconds, then stay on steady to indicate that the initialization process is complete. Afterwards, the jumper may be removed and the system will continue running as usual. Note, if JP10 is inserted during this procedure, the "Diode Calibration" process will also occur.

JP9 - Insert to calibrate squelch. With the receiver connected and its antenna removed, Insert JP9. In the next few seconds the "Receive Signal Indicator" (LED2, green) will flash on and off, then (hopefully) on steady. This indicates that the squelch calibration has occurred successfully. If unsuccessful, the LED will flash either fast to indicate that the discriminator noise level is too high, or slowly to indicate that the discriminator noise level is too low. Note, if JP10 is inserted during this procedure, the "Diode Calibration" process will also occur.

JP10 - Insert this jumper to perform "Diode Calibration". This may only be done in conjunction with a configuration parameter initialization (see JP8, above), or a "Squelch Calibration" (see JP9, above).

JP11 - Insert this jumper to temporarily re-purpose LED3 and LED4 to allow for visual indication of RX input level (see "Rx Level Calibration", below).

Potentiometers:

R22 - Squelch adjustment

R36 - Rx Input Level

R61 - Tx Output Level

Squelch Setting:

Once the "Squelch Calibration" procedure has been performed (see JP9, above), the squelch adjustment (R22) needs to be properly set. Make sure that the "External CTCSS" is set to "ignore" (value: 0). Adjust R22 until the Receive Signal Indicator (LED2) is lit. Then advance R22 clockwise until LED2 is no longer lit. That is the minimum squelch setting. You probably will want to crank it up at least another turn clockwise, because if you leave it there it will open on a REALLY low level signal. After proper setting has been achieved, return the "External CTCSS" configuration to its original setting.

Rx Input Level Calibration:

Place a full-quieting saturated signal on the receiver modulated by 1000 Hertz sine wave at 3KHz deviation. With JP11 inserted, LED3 will indicate (by brightness) if the RX level is too low, and LED4 will indicate (by brightness) if the RX level is too high. So the idea is to tune R36 so that there is minimal brightness on both LED3 and LED4 (like a null, more or less). Alternatively, the "97" command from the console menu gives a more graphical method of setting the Rx input level.

VOTER board console:

Serial Console access is provided by an RS-232 serial port (J3) at 57600 baud, 8 bits, No parity, 1 Stop Bit. It is also made available remotely (off-board) via Telnet. When telnet-ing the default password is "admin" and the password is "radios". When done with the console session, please use the 'q' command to disconnect. The local serial port and the Telnet (remote) user share the same (and only) console session.

Console Menu (commands):

The Console Menu (configuration and administration) commands are separated into the the following categories:

Configuration:

- Serial Number (MAC Address)
- IP/DNS Configuration
- VOTER Server Address
- UDP Ports(for VOTER communications)
- VOTER authentication info
- Tx Buffer Parameters
- GPS Device Settings
- Telnet authentication info
- Telnet port setting
- DYNDNS Settings
- External CTCSS Configuration

Administration:

- Set Debug Level
- RX Level Display
- System Status Info
- Quit Remote Telnet Session
- Reset (reboot) System

Discussion Of Various Parameters/Menu Functions:

The "Serial Number" (menu item 1), is an unsigned 16 bit number (0-65535) which defines the last 2 bytes of the MAC address of the ethernet interface. Be sure that the MAC address of the device is unique on the LAN to which it is connected.

The device may operate with DHCP-assigned IP information (menu item 7 set to 1), or Static IP information (menu item 7 set to 0). For Static configurations, menu items 2-6 allow specification of IP configuration information.

The "Voter Server Address" (menu item 8) is an ASCII string containing the FQDN of the VOTER server to which the board should communicate.

The "Voter Server Port" (menu item 9) allows specification of the UDP port on which the VOTER server operates. It is usually 667.

The "Local Port Override" (menu item 10) allows the specification of a UDP port number for the device to use locally. If specified as 0, it uses the same port as the "Voter Server Port" (see above).

The "Client Password" (menu item 11) allows the specification of the ASCII password for the client, as configured in the server (for that particular client).

The "Host Password" (menu item 12) allows the specification for the ASCII password the client expects to be receiving from the host.

The "Tx Buffer Length" (menu item 13) is the length, in samples (at 8000 per second), plus overhead, of the Transmit Delay buffer. This (plus the "Tx Buffer Delay", below) specifies the delay between the head end and all the clients on the transmit side. Must be set long enough to allow for worst-case delay and jitter in the Internet path between the client and the head end. Minimum setting is 800, maximum setting is 6400.

The GPS parameters (menu items 14-17) specify configuration for the GPS device. Trimble Thunderbolt devices (TSIP) are supported.

The "Telnet Port" (menu item 19) specifies the TCP port on which Telnet (remote console) server listens. This is generally 23. Its a good idea to change it to something more unusual.

The Telnet authentication parameters (menu items 20 and 21) allow setting of the remote console (Telnet) user name and password.

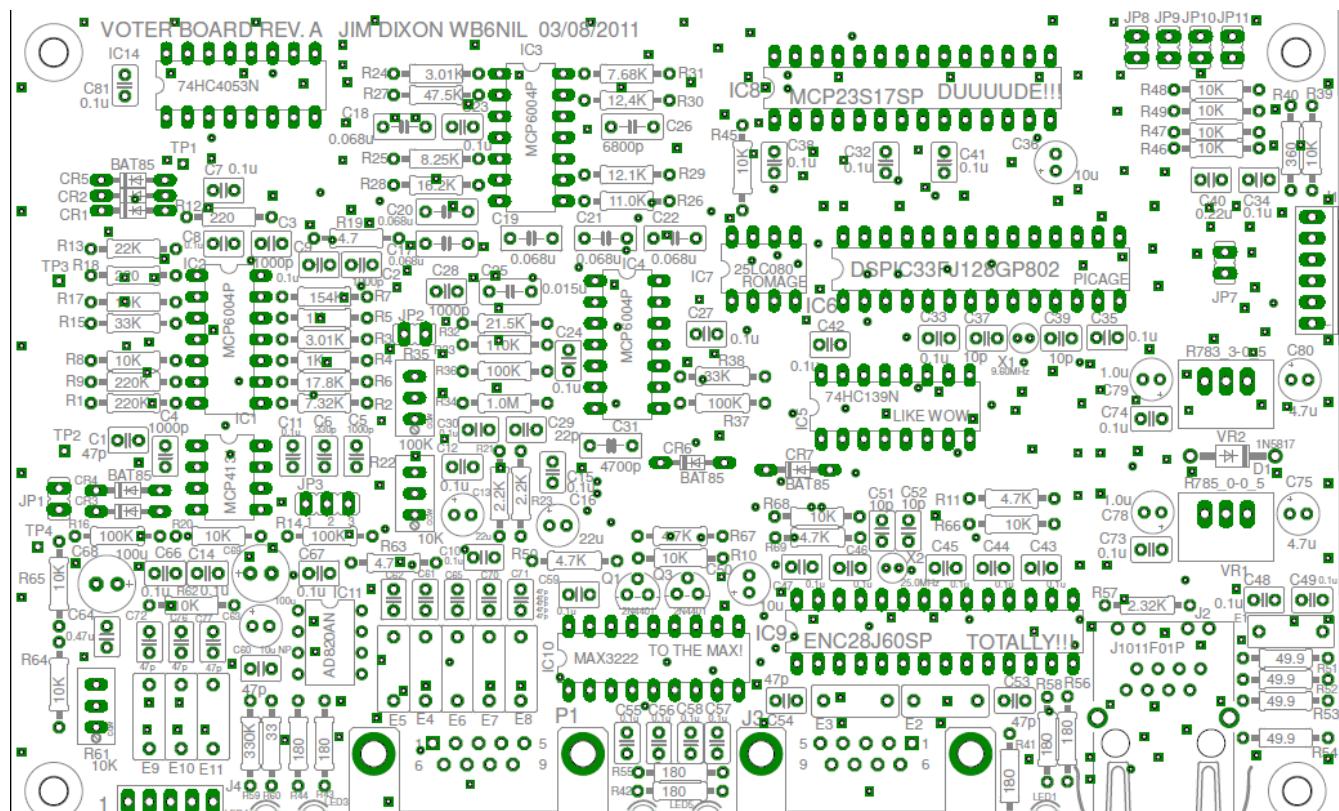
The DYNDNS parameters (menu items 22-25) allow configuration of the DYNDNS service, if desired.

The "External CTCSS" (menu item 26) allows specification of the configuration of the External CTCSS input. It may be ignored (value: 0), non-inverted (value: 1), or inverted value:2).

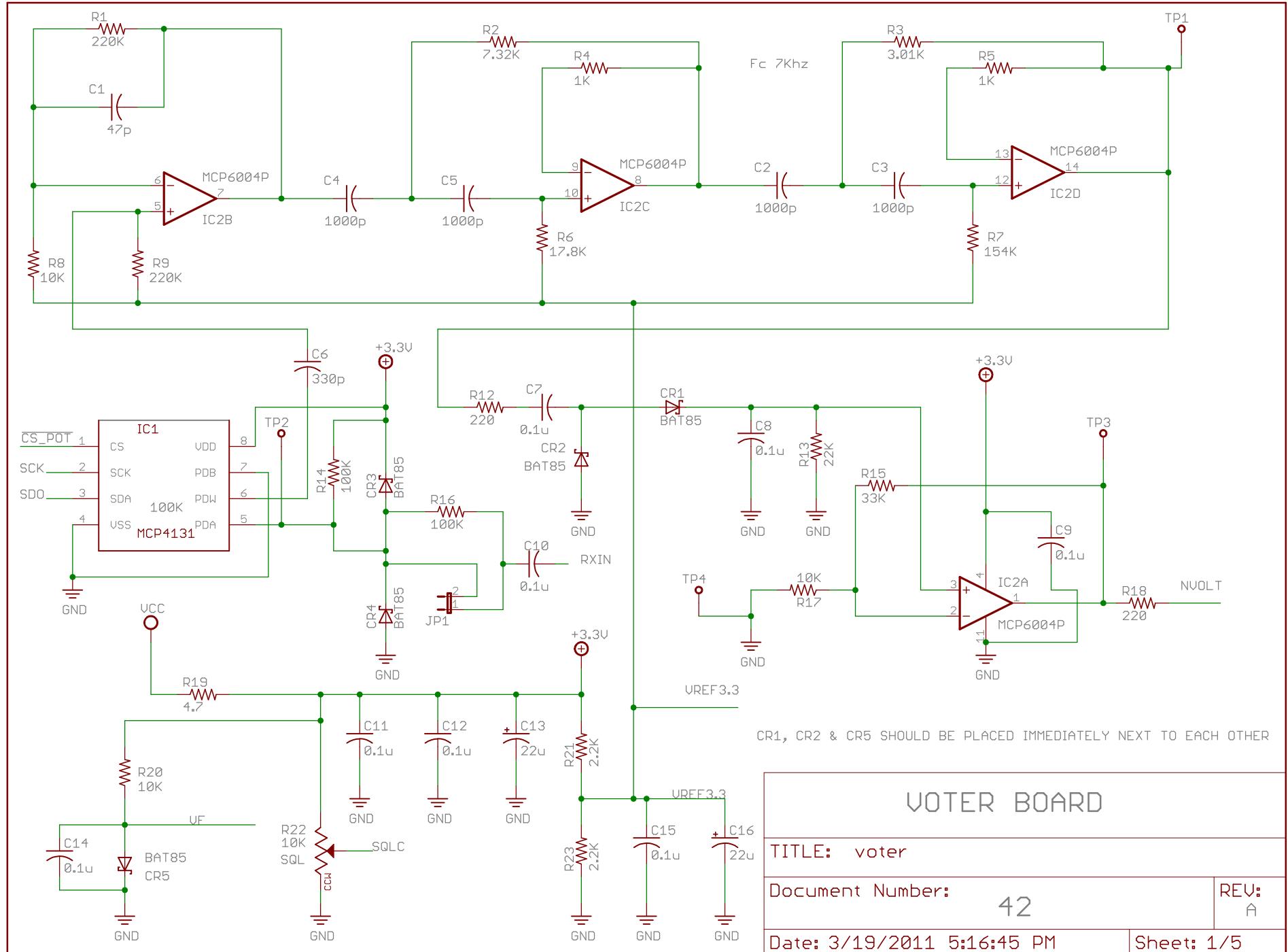
"Rx Level" (menu item 97) allows quasi-graphical display of the Rx input level.

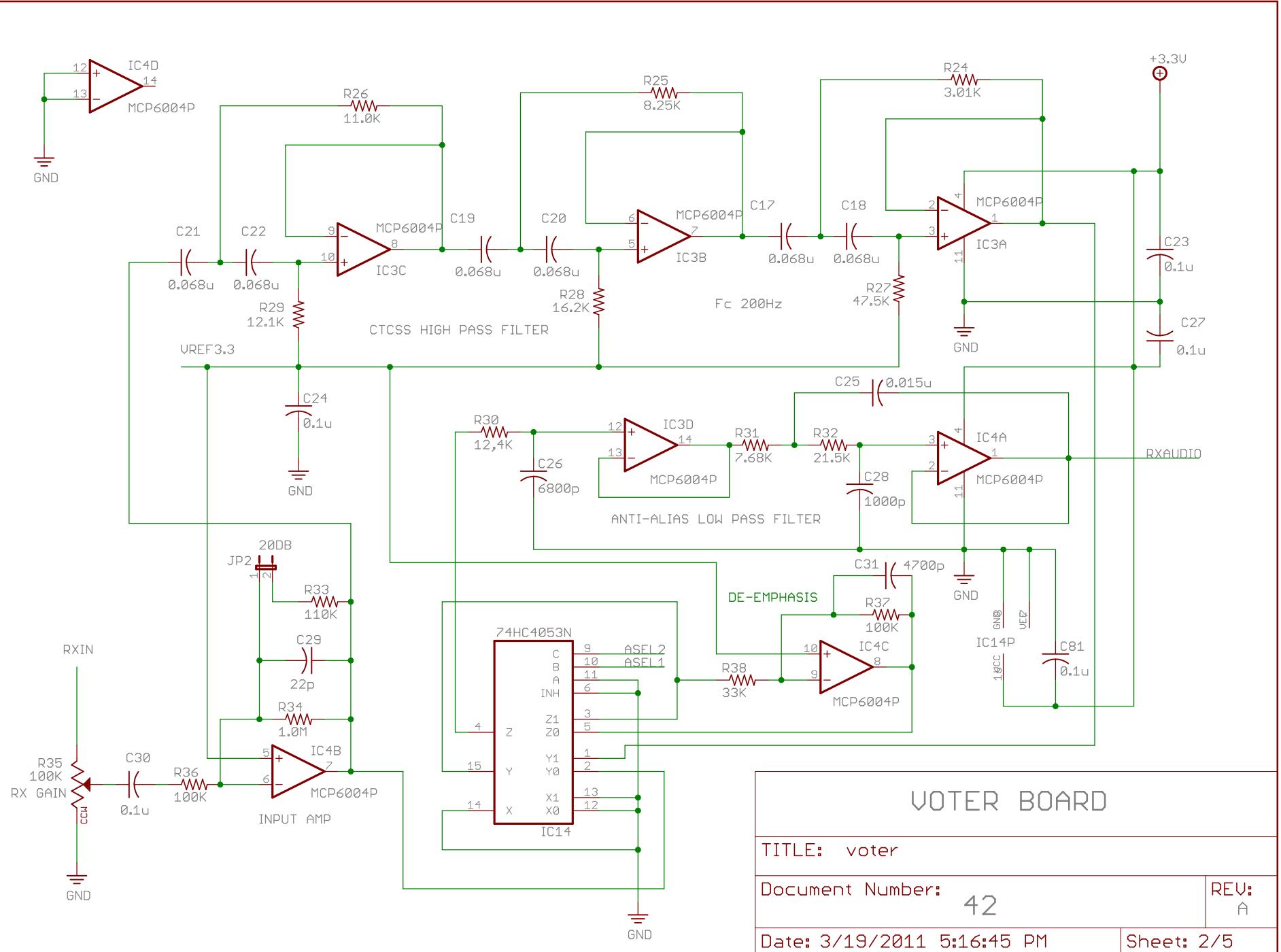
"Status" (menu item 98) shows various operating parameters.

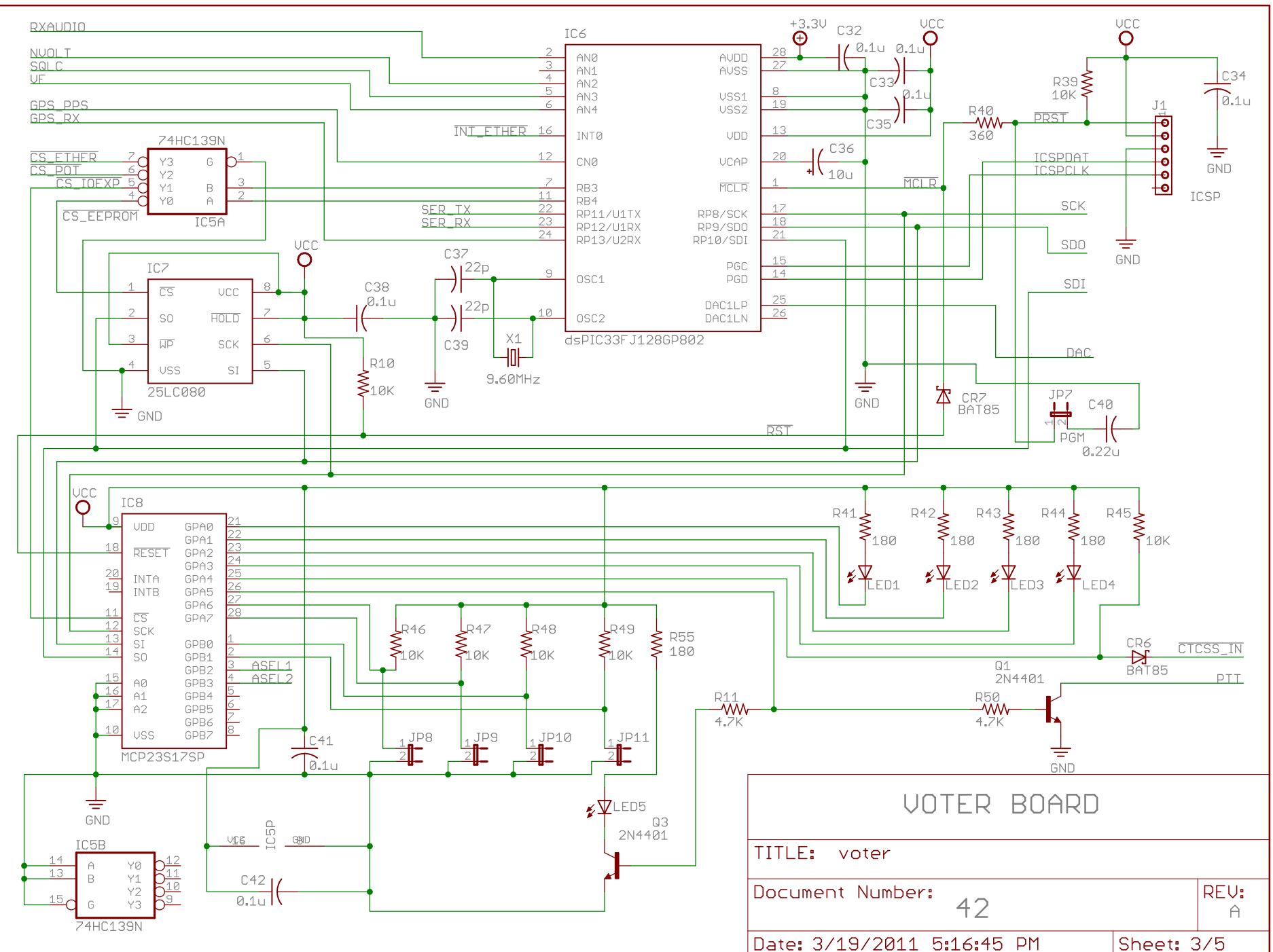
NOTE: The Ethernet must be connected for the firmware to start properly.

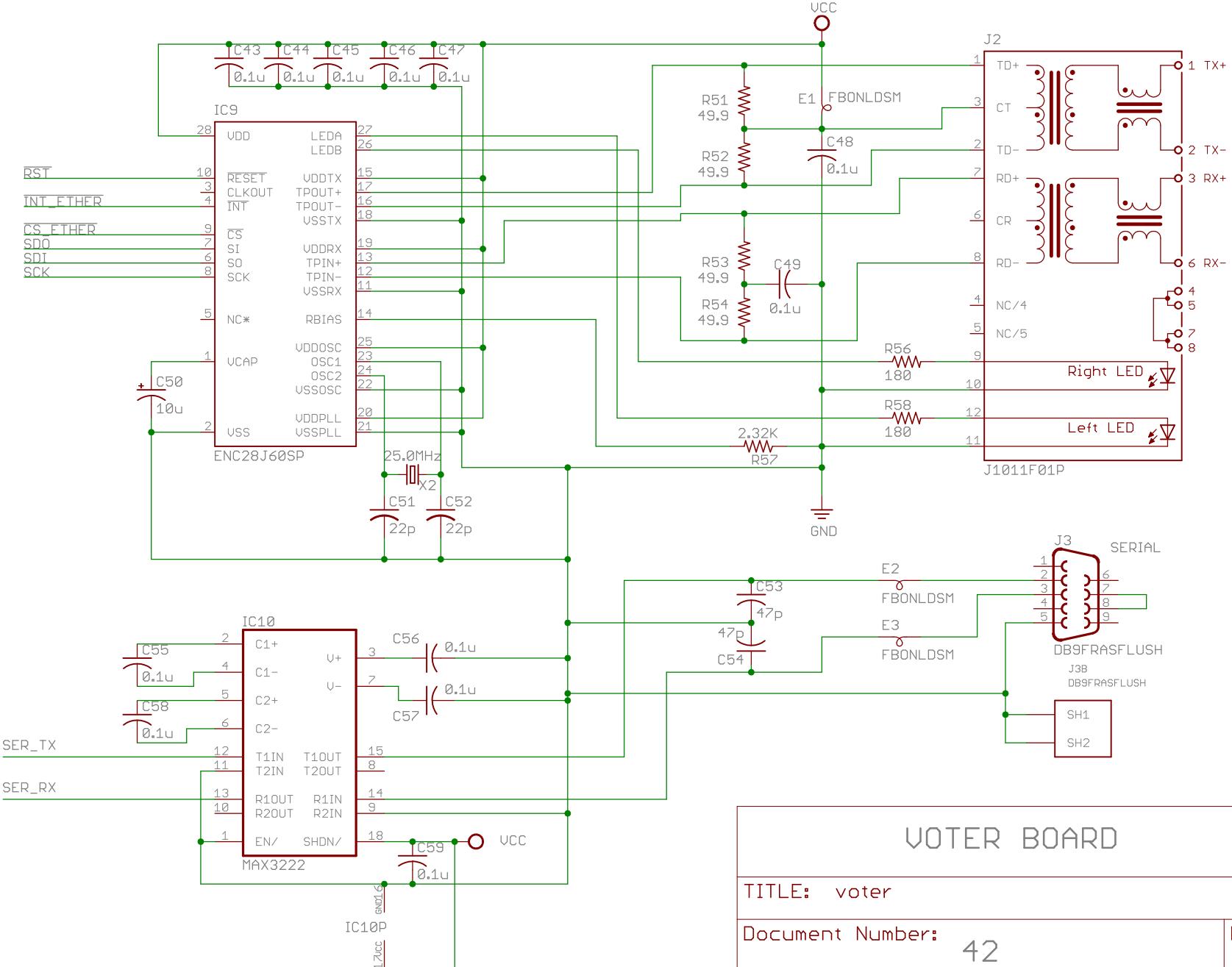


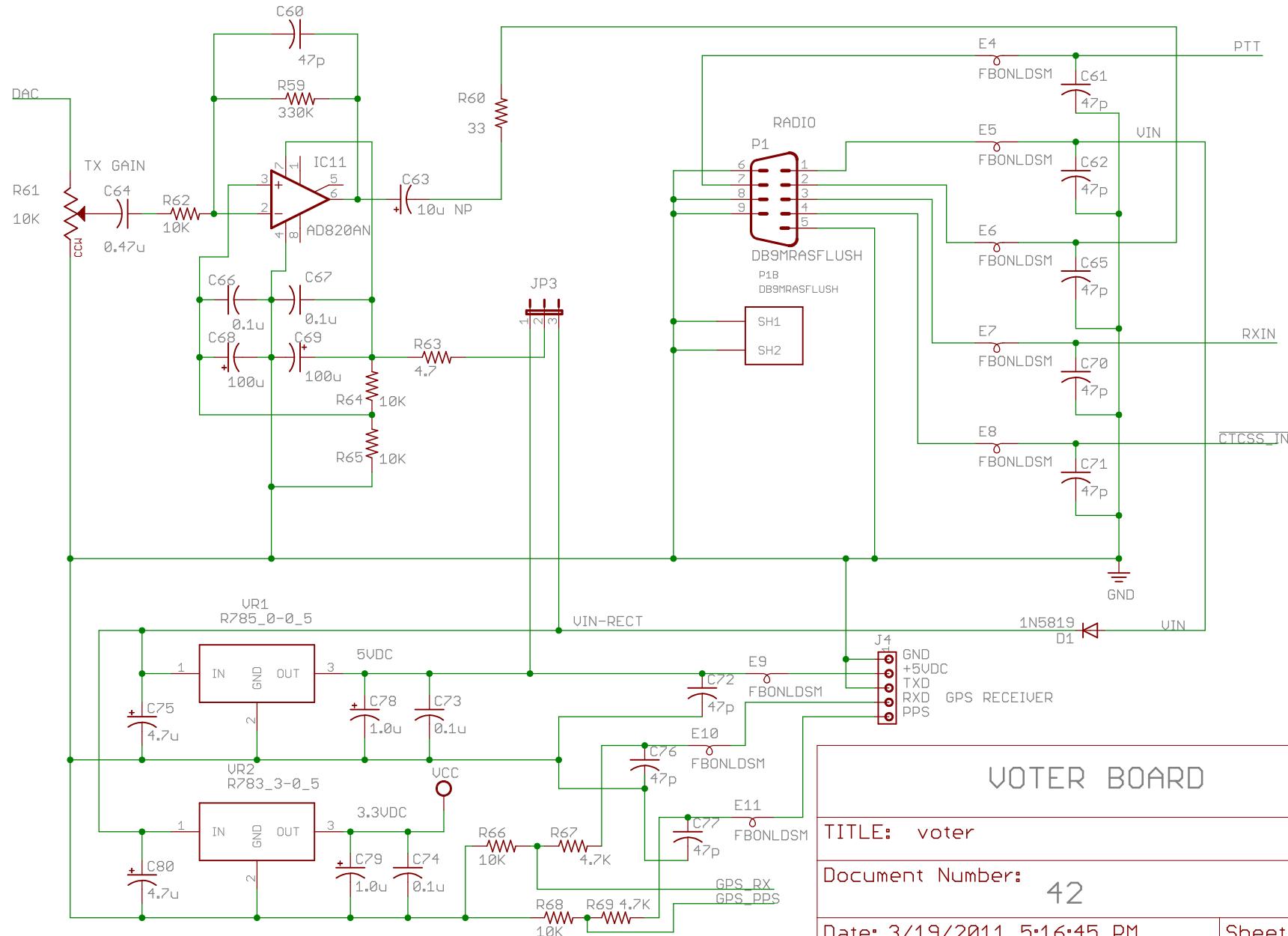
This document and information contained herein (including the use of it, and that which is derived from the use of it), and all other intellectual properties contained herein, and all intellectual property rights have been and shall continue to be expressly for the benefit of all mankind, and are perpetually placed in the public domain, and may be used, copied, and/or modified by anyone, in any manner, for any legal purpose, without restriction.











VOTER REV. A BOM

QTY	DIGI-KEY PART NUMBER	VALUE	DESIGNATION
12	399-4181-ND	47p	C1, C53, C54, C60, C61, C62, C65, C70, C71, C72, C76, C77
2	P5149-ND	22u	C13, C16
6	3012PH-ND	0.068u	C17, C18, C19, C20, C21, C22
5	399-4143-ND	1000p	C2, C3, C4, C5, C28
1	P4515-ND	0.015u	C25
1	3006PH-ND	6800p	C26
1	3005PH-ND	4700p	C31
2	P5148-ND	10u	C36, C50
5	399-4162-ND	22p	C29, C37, C39, C51, C52
1	399-4288-ND	0.22u	C40
1	399-4173-ND	330p	C6
1	P1187-ND	10u NP	C63
1	399-4309-ND	0.47u	C64
2	P5551-ND	100u	C68, C69
36	399-4151-ND	0.1u	C7, C8, C9, C10, C11, C12, C14, C15, C23, C24, C27, C30, C32, C33, C34, C35, C38, C43, C44, C45, C46, C47, C48, C49, C55, C56, C57, C58, C59, C66, C67, C73, C74,
2	P5566-ND	4.7u	C75, C80
2	P13460-ND	1.0u	C78, C79
7	568-1617-1-ND	BAT85	CR1, CR2, CR3, CR4, CR5, CR6, CR7
1	1N5819FSCT-ND	1N5819	D1
11	P9820BK-ND	FBONLDSM	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11
1	MCP4131-104E/P-ND	MCP4131	IC1
1	MAX3222CPN+-ND	MAX3222	IC10
1	AD820ANZ-ND	AD820AN	IC11
1	TC74HC4053AP-ND	74HC4053N	IC14
3	MCP6004-I/P-ND	MCP6004P	IC2, IC3, IC4
1	TC74HC139APF-ND	74HC139N	IC5
1	DSPIC33FJ128GP802-I/SP-ND	DSPIC33FJ128GP802	IC6
1	25LC080D-I/P-ND	25LC080	IC7
1	MCP23S17-E/SP-ND	MCP23S17SP	IC8
1	ENC28J60-I/SP-ND	ENC28J60SP	IC9
1	A19433-ND	CONN-MTA100-6	J1
1	553-1350-ND	J1011F01P	J2
1	A35108-ND	DB9FRASFLUSH	J3
1	A19432-ND	CONN-MTA100-5	J4
7		.100 JUMPER 2 POS	JP1, JP2, JP7, JP8, JP9, JP10, JP11
1		.100 JUMPER 3 POS	JP3
2	516-1792-1-ND	3 MM LED GREEN	LED1, LED2
2	516-1292-ND	3 MM LED YELLOW	LED3, LED4
1	516-1291-ND	3 MM LED RED	LED5

VOTER REV. A BOM

1	A35105-ND	DB9MRASFLUSH	P1
2	2N4401-APCT-ND	2N4401	Q1, Q3
2	220KQBK-ND	220K 5%	R1, R9
4	4.7KQBK-ND	4.7K 5%	R11, R50, R67, R69
2	220QBK-ND	220 5%	R12, R18
1	22KQBK-ND	22K 5%	R13
4	100KQBK-ND	100K 5%	R14, R16, R36, R37
2	33KQBK-ND	33K 5%	R15, R38
2	4.7QBK-ND	4.7 5%	R19, R63
1	7.32KXBK-ND	7.32K 1%	R2
2	2.2KQBK-ND	2.2K 5%	R21, R23
2	3296W-103LF-ND	10K 20 TURN	R22, R61
1	8.25KXBK-ND	8.25K 1%	R25
1	11.0KXBK-ND	11.0K 1%	R26
1	47.5KXBK-ND	47.5K 1%	R27
1	16.2KXBK-ND	16.2K 1%	R28
1	12.1KXBK-ND	12.1K 1%	R29
2	3.01KXBK-ND	3.01K 1%	R3, R24
1	12.4KXBK-ND	12.4K 1%	R30
1	7.68KXBK-ND	7.68K 1%	R31
1	21.5KXBK-ND	21.5K 1%	R32
1	110KQBK-ND	110K 5%	R33
1	1.0MQBK-ND	1.0M 5%	R34
1	3296W-104LF-ND	100K 20 TURN	R35
2	1.0KQBK-ND	1K 5%	R4, R5
1	360QBK-ND	360 5%	R40
7	180QBK-ND	180 5%	R41, R42, R43, R44, R55, R56, R58
4	49.9XBK-ND	49.9 1%	R51, R52, R53, R54
1	2.32KXBK-ND	2.32K 1%	R57
1	330KQBK-ND	330K 5%	R59
1	17.8KXBK-ND	17.8K 1%	R6
1	33QBK-ND	33 5%	R60
1	154KXBK-ND	154K 1%	R7
15	10KQBK-ND	10K 5%	R8, R10, R17, R20, R39, R45, R46, R47, R48, R49, R62, R64, R65, R66, R68
1	945-1037-ND	R785_0-0_5	VR1
1	945-1035-ND	R783_3-0_5	VR2
1	SER3418-ND	9.60MHz 18pf 30ppm	X1
1	SER3441-ND	25.0MHz 18pf 30ppm	X2

C37, C39, C51, C52 are marked on silkscreen as 10pf, but really are 22pf

D1 is marked on silkscreen as 1N5817, but really is 1N5819