COLORADO CONNECTION REPEATERS
"Connecting Colorado's Amateur Community"

Rocky Mountain Ham Radio
Teach ... Learn ... Operate ... Support. Be part of the team.

Voter2-2K
Users Guide

EDITED BY: WILL BEALS

CONTRIBUTORS: BEN MATTHEWS, DAVE MACIOROWSKI, DOUG SHARP, JAMES CIZEK, JOHN MAXWELL, MARK SKELTON

DRAFT VERSION: 0.00

AUGUST 18, 2025

An Open-Source Hardware and Software Project
www.colcon.org
www.rmham.org

1 INTRODUCTION

This document describes the interfaces, functions, setup, and operation of the Voter2-2K. It is the companion document to the Voter2-2K Hardware Guide which focuses on the hardware itself.

The Voter2 is designed to fully interoperate in a mixed environment with original Voters and a DIAL server. Intended as a backup only, it also includes basic stand-alone "local mode" repeater functions as well. While fully interoperable with the original Voter design, it is a completely new ground-up design effort. The reasoning for this is noted in the hardware document. The user interface and many functions operate differently, but it is 100% faithful to the original Voter protocol (well, outside bugs of course which will be addressed over time).

This guide assumes the reader is already familiar with how the Voter ecosystem works and this guide is how to operate a Voter2 in that system. I hope over time this document (or another companion document) will describe the ecosystem as well. Until then, the two documents by Jim Dixon (WB5NIL, SK) "Voter System" and "Voter Protocol" are (and will remain) the reference documents.

Moore's Law is a wonderful thing. Over the 10 years the Voter has been out, silicon capabilities have increased considerably for a given budget. The Voter2 has 25x the CPU horsepower (50x if you count the bus width doubling), 64x the RAM, and 32x the flash memory of the Voter1 for roughly the same cost as Voter1 parts cost today. The Voter2 is designed with headroom, less than a quarter of those noted resource being used with the initial feature set. The software architecture is based on an RTOS, so adding features and software maintenance should be easier as well. The feature set is expected to grow.

2 RELEASE STATE

This document revision is in sync with the Rev 0.00 (still pre-release) software. Only one major feature, simulcast support is not implemented. While I am quite confident the hardware is there to support simulcast, that feature is going to be after the "Rev 1.00" release. There is enough interest in the Voter2 without simulcast that I would like to get it released before the time and effort needed to get simulcast working.

Several known less-critical feature omissions remain as well, these include:

- Some local repeater modes to match the RTCM
- Locally implemented pre-emphasis on Tx audio (might be important)
- CTCSS detection (which may not be needed for the Voter-2K specifically)
- DIAL server failover support
- Fully compliant TFTP support and more error checking on firmware downloads

Importantly, this is all-new hardware and software compared to the Voter/RTCM. It is absolutely not as stable and vetted as the Voter/RTCM that has millions of hours of use time. While most features are implemented I am sure there are missing use cases still and it is not practical to test all combinations of all features, so there will be boundary cases that will crop up. These issues will be addressed as they are identified.

3 COMMUNICATION INTERFACES

This section describes each physical interface and then for interfaces like Ethernet and the backplane, each logical interface over that physical interface. Connections are listed from left to right looking at the front panel, then the back and finally internal connectors. Front panel space was scarce, forcing some tight constraints.

3.1 10MHz Reference

This is an optional input that may or may not be installed. It is an SMA connector for a GPS-stabilized 10MHz clock. This is an experimental input intended for better hardware support of simulcasting that may no longer be needed. See the hardware reference manual for more information. There is no software support for this feature.

3.2 USB Console

This mini-USB connector is connected to an FTDI UART chip. "Regular" serial ports are alas extinct and FTDI drivers are native to most operating systems now, so hopefully this interface should be plug and play. Set the data rate to 115,200. The Voter2 should work with default settings for both PuTTY and Tera Term for all other settings. If you are using another terminal program (or changed defaults for the above two), the other major settings are:

- 8 data bits, 1 stop bit, No parity
- No implicit CRs or LFs
- No local echo ("auto" works fine too)
- No local line editing ("auto" works fine too)
- 0x08 or 0x7f for backspace (both supported)

The only line editing supported is backspace. I hope to enhance this in the future. The USB console and the Telnet console have effectively identical functions and are mutually exclusive. If one interface is active and a connection established on the other, the console will log out of the previous interface and then allow you to log into the interface just established. The unused interface goes dormant.

The USB console does have one "special power" the Telnet interface does not have, the ability to do a factory reset. See the section on logging in.

3.3 ETHERNET

This is a 10/100 full duplex Ethernet interface with status LEDs.

- Green: Connection speed, Off for 10Mbps, on for 100Mbps
- Yellow: Activity.

The MAC address uses the Locally Administered Unicast Address format which is essentially a random address. This is assigned at board manufacturing and programmed into the OTP area of flash, so permanently set.

The interface supports fixed IP addressing, DHCP and DNS. The Voter2 can be pinged.

Listed below are the currently supported Ethernet ports:

3.3.1 DIAL Port

(UDP Client, random port) For comms with the DIAL server. For the Voter2 it is a random outgoing port number. For the host (the DIAL server) is either 667 for ASL1 or 1667 for ASL3.

3.3.2 Telnet Console Port

(TCP Host, Port 23 – fixed) Telnet interface for the console. Hopefully the Voter2 is set up to work with defaults for both PuTTY and Tera Term. If you are using another terminal program (or changed defaults for the above two), the other important settings are:

- No implicit CRs or LFs
- No local echo (negotiated using Telnet commands to disable)
- No local line editing (negotiated using Telnet commands to disable)
- 0x08 or 0x7f for backspace (both supported)

The only line editing supported is backspace. I hope to enhance this in the future. The Telnet and USB based console have effectively identical functions and are mutually exclusive. If one interface is active and a connection established on the other, the console will log out of the previous interface and then allow you to log into the interface just established. The unused interface goes dormant.

3.3.3 Logger Port

(TCP Host, Port 24 – fixed) The Voter2 has a tiered logging setup segmented by functional block (see the `set logging` command). The default interface on boot for this logging is the UART dedicated to the ST debugger interface. Logging into this port with a Telnet client will direct the logging data here instead of the UART, allowing remote logging. Disconnecting the Telnet client will return logging data to the UART.

Logging messages begin well before the IP stack is up and indeed even before the RTOS is active, so initial logging is always to the UART which is default on boot.

3.3.4 Ser2Net (RFC2217) Port

(TCP Host, Port 2000 – user changeable) RFC2217 is a protocol allowing a UART to be controlled remotely via a Telnet connection. Most repeaters including the MTR-2000 have a serial port for command/configuration. By using this protocol and a UART-to

RFC2217 driver on the host PC, you can run the standard Motorola RSS software (yes only on 32-bit Windows 95) and be able to communicate with the MTR-2000 remotely.

The RFC2217 implementation on Voter2 does not support the complete RFC2217 feature set. It implements just enough to support the MTR-2000 software, namely baud rate and data format (data bits, parity, stop bits). No support is included for flow control or handshaking signals. If needed for future projects with different repeaters these features can be added.

Refer to Appendix A on how to set up the PC side software to work with the Voter2.

3.3.5 TFTP Port(s)

The Voter2 supports remote firmware updates via TFTP. The Voter2 is the TFTP client.

Future feature such as status and error logging may be enabled via TFTP.

3.4 MTR UART

This is the UART interface to the MTR-2000's RSS connector. An RJ45 connector right next to the Ethernet connector isn't great, but I did want to allow the use of an off-the-shelf cable between the Voter2 and MTR-2000. A standard but very short Ethernet cable will do the job. Please do yourself a favor and mark that cable as NOT ETHERNET.

The interface is standard RS-232 voltage levels but only connecting Tx, Rx, and ground. The MTR-2000 does not support any handshaking and neither does the Voter2. All communication parameters are controlled via the RFC2217 protocol.

3.5 LEDs

There are six status LEDs on the Voter2.

- PWR (green) – Power. On if there is 3.3V, not under any software control.
- HB (green) – Heartbeat. Blinks once/second for normal operation. During initial boot, if the NVM is blank or corrupt this LED will blink five times rapidly. See the note in the setup section regarding booting with factory defaults.
- RX – Receive Status. This is a three-color LED (plus off) indicating the Rx levels. Stati are:
    - Off – Nothing recognized as a valid transmission (no carrier/RSSI stat and/or no valid CTCSS tone).
    - Yellow – Valid signal, but low audio level (less than 1/3 full scale)
    - Green – Valid signal between 1/3 and 2/3 full scale
    - Red – Valid signal greater than 2/3 full scale.

  During normal operation the LED should be bouncing between yellow and green with occasional flashes of red.

- TX (red) – Indicates the Voter2 is telling the MTR-2000 to transmit.
- DIAL (yellow) – Indicates the DIAL connection state. Off for no connection, on for connected and authorized and flashing for connected, but not authorized (likely a bad Host Password). Note that a bad Voter Challenge string or bad Voter password isn't obviously detectable, so won't cause the LED to flash.
- GPS (yellow) – Indicates the GPS link state. Off for no data, blinking for getting data but not yet 1PPS, and solid on for GPS data and valid 1PPS.

3.6 GPS(es)

The Voter2 supports two different GPS modules. Space constraints requires overlapping footprints, so only one or the other can be installed, there is no provision for installing both and selecting between them. The two GPS modules supported are:

3.6.1 Ublox M8n

This is a mezzanine board with a simple 5-pin interface. It contains the SMA connector and its own battery so completely self-contained. While the Ublox module is a good and reliable module, the mezzanine board is from suppliers that do not look like long term suppliers and offering complete boards for much less than purchasing just the Ublox module itself, so of "questionable origins". The ones we do have perform well, so this board supports them as long as we can get them.

Of note, if off for more than a day, it can take up to 30 minutes to get a 1PPS signal even though GPS position data is valid within a minute.

3.6.2 ST Teso LIV4

This solution is just the module. It is more expensive than the Ublox mezzanine board, but is a fully supported in-production module with guarantees and long term production commitments. While the Ublox module is opportunistic, this is the "10-year" solution. This requires an external SMA connector. While there is support for a battery for ephemeris data, its acquisition time including 1PPS is fast enough without it. The battery does not seem necessary.

3.7 MTR-2000 Backplane

The Voter2 can plug into either the Option 1 or Option 2 connector in the MTR-2000 card cage. The silkscreen on the bottom shows which pins have connections. If you are hand soldering, these are the only pins that need soldering. The supported signals on this connector are:

3.7.1 Rx Audio

Audio from the MTR 2000 to the Voter2. The Voter2 DC isolates it. Max signal amplitude is 2.2Vp-p. 2.2Vp-p represents full scale into the MCU ADCs.

3.7.2 Tx Audio

Audio from the Voter2 to the MTR-2000. The Voter2 DC isolates it. A full scale out of the DACs results in 1Vp-p on this line.

3.7.3 RSSI

Analog RSSI from the MTR-2000. The Voter2 digitizes this input and interprets it per the information on page 33 of the MTR-2000 field service guide. Note this is only used if analog RSSI is selected.

3.7.4 PTT

How the Voter2 tells the MTR-2000 to transmit. It is an open collector output of the Voter2.

3.7.5 CARRIER DETECT

Tells the Voter2 the MTR-2000 recognizes a valid carrier. This is only used if selected via the UI.

3.7.6 RDSTAT

Tells the Voter2 the MTR-2000 recognizes a valid CTCSS tone. This is only used if selected via the UI.

3.7.7 RESET

Allows the MTR-2000 to reset the Voter2. While provisions are made to connect this signal, it is not connected.

3.7.8 OPTION ID

This is an analog voltage to the MTR-2000 from the Voter2 for presence detect as well as identifying the classification of the board.

3.7.9 MTR SPI Interface

This allows the MTR-2000 to set or query 16 control bits. There is no known use for this feature (yet), but supported in hardware. The MTR-2000 is the SPI bus master, the Voter2 is the slave. There are separate chip selects for read and write, but they share the CLK, MISO, and MOSI lines. Additionally the OPT_IRQ signal from the Voter2 to the MTR-2000 can be asserted to force a read of the read bits.

3.7.10 SPARE UART

This is an extra UART sent to the backplane intended for an S-COM board that will plug into the MTR-2000 System connector. It is currently not used by the Voter2.

3.7.11 SPARE I2C

This is an extension of the internal I2C bus sent to the backplane intended for an S-COM board that will plug into the MTR-2000 System connector. It is not directly used by the Voter2. If, however, you have some extra P3T1755 temperature sensors hooked up to these lines with unique addresses, they will be detected and temperatures displayed on request.

3.8 ST DEBUGER INTERFACES (J1 & J11)

The Voter2 supports two debugger interfaces. The 14-pin interface is a full featured interface supporting regular debugging, instruction tracing, and the UART logging. The baud rate for the UART logging is 115,200 baud. This port requires a higher end debugger dongle for these features. For a simpler and MUCH cheaper debugging interface, the 6-pin interface matches the connector on STs Nucleo modules. This connector doesn't support instruction tracing or the UART logging interface, but fully suitable for regular debugging.

3.9 REAL-TIME STATUS INTERFACE (P5)

For real-time monitoring of critical software functions such as ISRs and DSP routines this connector provides three GPIO lines that can be wiggled by software for viewing by an oscilloscope. They are labeled AISR for Audio data processing, TISR for Timer interrupt processing and DISR specifically for measuring DSP routine execution time. These are just GPIOs, so can (and are) routinely changed to measure specific pieces of critical code execution time.

The fourth output is an analog output synchronized with the audio DSP chain. Using the `set probe` CLI command, you can probe the audio signal at any of the stages of the DSP chain to verify the signal quality/integrity. It is the DSP equivalent of probing the signal at the output of the various filters if they were implemented as circuitry. See the `set probe` command for details.

3.10 TX AUDIO TP (P6)

Analog audio from the Voter2 to the MTR-2000 after DC isolation (so centered around 0V) before the 600 ohm source impedance resistor. It has a matching ground pin for better signal measurement. This is the point where full scale should be 1Vp-p.

3.11 RX AUDIO TP (P7)

Analog audio from the MTR-2000 to the Voter2 after DC isolation (so centered around 1.65V). This is the point where full scale should be 2.2Vp-p.

3.12 POWER (J7)

A place to check all four voltages, +5 digital, +5 analog, +3.3V digital, and +3.3V analog. Both analog and digital grounds are present as well, but note that the common tie point between the two is very close to this connector.

3.13 BATTERY JUMPER (J10)

If the battery is not being used, this jumper supplies +3.3V digital to the 3.3V battery bus so the battery-operated circuitry has a valid voltage whenever the board is powered. Clearly do NOT install this jumper and a battery at the same time!

4 BASIC SETUP

4.1 Factory Reset

To restore the Voter2 NVM to factory defaults requires you log into the console over the local USB interface. When you get to the login prompt, enter a username of `factory` and a password of `reset`. This will purposely corrupt the NVM so that on the next reboot it will load factory defaults instead of NVM settings.

4.2 Booting up with Factory Defaults

When the Voter2 boots up with a blank or corrupt NVM you will see the heartbeat LED blink rapidly five times before reverting to the normal once-per-second flash. When the Voter2 boots with factory defaults, it loads them into local memory only, IT DOES NOT UPDATE THE NVM! This is on purpose. This allows you to inspect the NVM if you wish first. As such, after logging in and making any changes, you must issue the `settings save` command before the next reboot or you will end up repeating the process.

4.3 Initial Login after Factory Reset

A voter2 powering up with factory defaults has the following important settings for getting started with the command line:

- USB UART: 115,200 baud
- Telnet: DHCP for IP address, port 23.

Figuring out the DHCP-assigned address is messy, so the easier way to get started is to use the USB UART. Odds are very good your OS has FTDI drivers already, if not you will need to get the drivers for the FTDI FT230XS. There should be no reason to change any other defaults for either PuTTY or Tera Term.

There are two hard-coded login accounts for the Voter2, `root` and `guest`. The factory default passwords for them are `guru` and `password` respectively. The main difference between the two accounts is that only root can view/change any of the passwords (root, guest, dial/voter passwords, and the challenge string. Some other functions such as I2C memory dump and possibly Ser2Net settings may be limited to root access. Saving settings to NVM Is not allowed in guest mode, making it harder for a guest to permanently break the Voter2. Anything the guest does can be undone by a reboot.

4.4 Debug/logging interface

A big difference from the Voter is the Voter2 uses a separate interface for debugging and logging messages than the console. This defaults to the UART on the 14-pin ST Debug connector, but you can also access it via Telnet on port 24. See the sections above regarding these two interfaces. The command to control logging is `set logging`.

5 COMMAND INTERFACE

Unlike the Voter, the Voter2 uses text based commands with a rudimentary help system to provide reference. This is more verbose than the numeric interface, but more flexible. Right now all commands and text parameters must be lower case to be recognized. Passwords like string parameters can be upper and lower case. I may change this if enough folk complain. The first useful command is `help` (shows all possible commands). Help followed by a command shows the syntax for that command. The most likely initial commands are going to be for settings, so `help set` will show you all the settings that be changed. The `show` command will show the value of any of those settings. While a bit clunky, there is also a `show all` command that lists all settings with one command, get ready to scroll back.

If you make any changes to settings (or came up with factory defaults), these are only stored in local RAM, not NVM. There will be an asterisk before the > of your command prompt. That asterisk is a reminder that you have settings in local memory that have not been saved to NVM and will be lost if you reboot. This allows a very limited "undo" by simply rebooting the Voter2.

I do hope some day to have at least a last command recall with simple editing.

Next is the reference list for the current commands on the Voter2. Commands are in alphabetical order, but for settings they are grouped and listed as they are in the help listing.

5.1 cls

Send the ANSI/VT100 FF (Formfeed, 0x0c) byte to the console. If you have almost any kind of terminal emulation this should clear the screen.

5.2 help

This command by itself lists all commands.
`help` followed by a command lists the syntax for that command.

5.3 logout

Logs out of the console session. You are returned to the login prompt. The connection (UART or Telnet) is not closed. If done, just close your terminal program afterwards.

5.4 mtrspi

This is to test the MTR-2000 SPI interface. There is no known use for this interface on the Voter2, it is for the S-COM project. This interface can affect the functions of the MTR-2000, so don't mess with it!

5.5 rtc

This feature and commands are only useful if a battery has been installed. For the Voter2 where a super-precise time is available via GPS and/or a DIAL server, it is likely not useful. If the Voter2 is operating as a stand-alone device in local mode only, then this can be the time reference. While settable and viewable in this command line interface, it is currently not used anywhere else in Voter2 code. If there is no battery and the sync command hasn't been issued, the RTC value is essentially the "time since power-up". Three RTC commands are available:

`sync`: If you do have valid GPS time, this command will program the RTC with the GPS time. This will be UTC time.

`set yy mm dd hh mm ss`: Manually set the RTC time.

`show`: Show the current RTC time.

5.6 reboot

Reboots the Voter2.

5.7 settings

This command is to manage saving and restoring settings as a whole. If you have modified any individual setting or came up in factory default mode, then your settings in local memory are not in sync with what is in NVM. If that is the case, you will see an asterisk before your ">" command prompt. If you reboot or enter the `settings reload` command those local settings will revert to the ones in NVM. Note though that settings are not reloaded if just logging out and back in again.

The settings commands are:
`save`: Save current settings in local memory to NVM. The total number of bytes written is noted, get worried if it is getting close to 8192. This command is only available in root mode. This allows a guest to try settings, but can't make any permanent changes.

`reload`: Revert current settings (if changed) back to what is in NVM. The total number of bytes read is noted.

`dump`: This is a debugging command to dump the contents of a range of NVM. If no additional parameters are provided, byte addresses 0-255 will be shown. Valid start and end addresses for the 24LC64 are 0-8191. This command is only available in root mode.

5.8 set

This command and its companion `show` command allow you to set and show the operating parameters of the Voter2. Settings changed with the `set` command are only changed in local volatile memory. They are not automatically saved in NVM. If you want the changes to be permanent (across reboots), the settings should be saved to NVM using the `settings save` command. If a setting requires a reboot to take effect, this will be noted in command feedback.

Commands listed below are by functional grouping instead of alphabetical and are in the same order as the `help settings` string.

For parameters that are a list of items, to get the list of possible parameters, enter set, the setting and then something junky and help will spit out the possible parameters. For example, enter `set myipmode harry` and you will get back the possible `myipmode` settings. I promise I will make this better.

5.8.1 NETWORK SETTINGS

5.8.1.1 mystaticip, mystaticsn, mystaticgw, mystaticdns

If `myipmode` is set to `static` these will be the Voter2's IP, subnet, Gateway, and DNS server addresses respectively. The parameter is the standard xxx.xxx.xxx.xxx format where xxx is 0-255. Ignored if `myipmode` is set to `dhcp`.

5.8.1.2 staticdialip

If `dialipmode` is set to `static` this is the DIAL servers IP address. Ignored if `dialipmode` is set to `dhcp`.

5.8.1.3 myactualip

The Voter2's actual IP address regardless of the source (show only).

5.8.1.4 tftpaddr

The IP address of the TFTP server for software downloads and (some day) NVM backup/restore and logging.

5.8.1.5 dialactualip

The DIAL servers actual IP address regardless of the source (show only).

5.8.1.6 mymac

The Voter2's MAC address. This is in OTP, so show only.

5.8.1.7 myserialnum

The Voter2's serial number. This is in OTP, so show only. If the Voter2's OTP is not yet programmed, then this is the command to program all OTP values, the serial number, hardware revision, and MAC address. In this case, there are three parameters, the serial number (32-bits, so huge), the hardware major revision 0-255, and the hardware minor revision (0-99). The MAC address is not a parameter, it is randomly generated, locally administered, unicast MAC address (that's an official thing!). As these are OTP values, the command to set them can only be executed once and will normally be executed as part of board manufacturing. After that, this command is show only.

5.8.1.8 Myhwrev

The hardware revision of the board. This is in OTP, so show only.

5.8.1.9 myipmode

- Voter IP address acquisition mode. It can be `static` or `dhcp`.

5.8.1.10 guestpwd

This is the password for the Voter2 guest login. It can be up to 16 characters.

5.8.1.11 rootpwd

This is the password for the Voter2 root login. It can be up to 16 characters.

5.8.1.12 s2nport

Ser2Net is the protocol for remote UART connections managed over Ethernet using the RFQ2217 standard. This is the listening port for that service. The default is 2000, but this command can change it. It is any valid port number, 0-63335.

5.8.1.13 s2nptime

This is the polling time for serial data from the MTR-2000. The default of 100ms works with the Motorola software. If you are using different software that may be more sensitive to timing issues, this value can be decreased at the expense of more CPU time. If you tweak it, do try for the longest possible time that does not generate errors.

5.8.2 DIAL SERVER SETTINGS

5.8.2.1 unauthrate

This is the rate (in ms) to send authorization packets to the server when in the unauthorized state. The default is 1000ms. It would be strange to change this, but it can be if needed.

5.8.2.2 txbuf

This is the amount of transmit audio data to buffer up in ms before asserting PTT and transmitting audio. This value is highly dependent on your network setup and in general you are trying to make it as small as possible without buffer underruns.

NOTE: While this parameter is in ms, the Voter2 "rounds" this up to the nearest packet length of 20ms. A length of 1-20ms will result in a 20ms delay, a value of 21-40ms a 40ms delay, etc. This coarseness should be OK until we start playing with simulcasting.

Acceptable values for this parameter are 60-800ms. You do want this value as low as practical as this time adds directly to system delay.

NOTE: This parameter also applies to local mode.

5.8.2.3 compression

The Voter protocol supports two Audio compression standards, uLaw and IMA ADPCM (Intel/DVI block format). The DIAL server tells the Voter2 which format to use so this command doesn't do anything! (needs work).

5.8.2.4 packettiming

Tells the Voter2 what packet timing mode to use. The official options are `gp` for General Purpose mode also known as mix mode or non-GPS mode, or `gps` for GPS-timed mode. This must be `gps` for voting operations. A third mode, `fakegps`, is for test purposes where the Voter2 gets the time from the DIAL server and then fakes gps packet timing using that time. This latter mode should not be used outside bring-up testing.

5.8.2.5 ipmode

The DIAL server IP acquisition mode. It can be `static` or `dns`.

5.8.2.6 fqdn

If the IP acquisition mode for the dial server is DNS, this is the FQDN (path) for the DNS server to look up. It can be up to 40 characters.

5.8.2.7 port

The UDP port number for DIAL communications. This should be 667 for DIAL servers up until ASL3, then 1667 for ASL3 (and probably later).

5.8.2.8 voterpwd

For Voter/DIAL authentication, this is the voter password. It can be up to 16 characters.

5.8.2.9 hostpwd

For Voter/DIAL authentication, this is the host password. It can be up to 16 characters.

5.8.2.10 voterchal

For Voter/DIAL authentication, this is the challenge string. It can be up to 10 characters.

5.8.2.11 restart

This will reset the dial connection state and stop all DIAL comms for about 5 seconds for things to reset, then allow dial comms to restart. No other IP ports are affected.

5.8.3 GPS AND MISC SETTINGS

5.8.3.1 Local

If communications with the DIAL server are down, the Voter2 reverts to Local mode for stand-alone operation. This is a minimal repeater controller operation. As soon as comms with the DIAL server are restored, the Voter2 goes back to Voter mode. The settings below apply to implementing the stand-alone local mode.

5.8.3.1.1 mode

The operational mode while in Local Mode. I don't use this parameter, need to see if I'm missing something!

5.8.3.1.2 cwspd

The speed at which to send CW identifiers, the station ID and proceed ID. In wpm.

5.8.3.1.3 plfreq

The desired transmit CTCSS Frequency when in local mode. If you do not want a transmit CTCSS frequency, set `pllevel` to 0 and you can ignore this setting. The standard CTCSS frequencies are supported: `67.0 71.9 74.4 77.0 79.7 82.5 85.4 88.5 91.5 94.8 97.4 100.0 103.5 107.2 110.9 114.8 118.8 123.0 127.3 131.8 136.5 141.3 146.2 151.4 156.7 162.2 167.9 173.8 179.9 186.2 192.8 203.5 210.7 218.1 225.7 233.6 241.8` and `250.3`.

5.8.3.1.4 deemph

Enable/disable De-Emphasis in local mode. Setting can be `on` or `off`.

5.8.3.1.5 precw

The delay between the end of any repeated audio and the next ID which could be the proceed ID or station ID. In ms.

5.8.3.1.6 postcw

The delay between the proceed ID and station ID if both are sent. In ms,

5.8.3.1.7 idtime

The time between station ID transmissions if anything was repeated. FCC suggests 10 minutes, so unless you are doing something weird the default to 600 seconds is normal.

5.8.3.1.8 hangtime

The delay from whatever was last to be transmitted (audio, proceed tones, ID, etc) and releasing PTT. In ms.

5.8.3.1.9 pllevel

Volume level of the local CTCSS tone. Value is 0 (none) and 1-10 where 1 is 10% and 10 is 100% (all CTCSS, no audio!). The audio level is decreased by the CTCSS level to maintain the overall level. Only values 0-3 make much sense.

5.8.3.1.10 id

The local callsign ID string. This can be the characters A-Z, 0-9, - or /. The max length is 10 characters.

5.8.3.1.11 proc

The local proceed ID string (sent after every transmission) in local mode. This can be the characters A-Z, 0-9, - or /. The max length is 10 characters.

5.8.3.2 gpsdev

This is a convenience shortcut command to set up the GPS receiver parameters by device, saving you the effort of looking them up. It assumes you haven't changed the defaults of the GPS! GPS modules currently supported are `m8n` and `liv4f` for the Ublox M8N and ST Teso Liv4f respectively. The settings changed are `gpsbaud` and `gpsedge`.

5.8.3.3 gpsbaud

Sets the baud rate of the GPS receiver. Parameter can be between 2400 and 115200.

5.8.3.4 gpsedge

Selects the active edge of the GPS interfaces 1PPS signal, the edge representing the time in the NMEA message. Parameter can be `rising` or `falling`.

5.8.3.5 rxmode

This selects what signals and conditions constitute a valid received signal (that is, means the audio gets sent to the DIAL server or rebroadcast if in local mode. Parameters are:

- `off`: Never receive. For a TX-only site.
- `cor`: Rely on the MTR-2000 COR signal only (which is carrier detect)

- `ctcss`: Rely on the MTR-2000 RDSTAT signal only (which is valid CTCSS).
- `corctcss`: Rely on both COR and CTCSS being valid.
- `on`: Transmit always (!!!). For testing purposes. Note that this mode overrides the timeout timer in local mode.

When I get the out-of-band RSSI function working this will get a bit more complicated.

5.8.3.6 rssimode

This setting selects what source to use for determining the RSSI value to send to the DIAL sever. The two options are `analog` and `hpf`. `analog` selects the MTR-2000 RSSI voltage, `hpf` selects the out-of-band high pass filter noise to be used for RSSI calculations. The latter is what the Voter1 uses. Using the analog RSSI value is experimental!

5.8.3.7 logging

The Voter2 uses a separate interface for posting logging information rather than sharing it with the console. See the sections on the Ethernet logger port and ST Debug ports for descriptions of where logging information is sent.

The Voter2 has a more granular approach to logging than the Voter1. There are only four logging levels, but logging can be set uniquely for each functional block, allowing more or less detail by block, not for everything. In general the levels mean:

- `none`: No messages at all from this block.
- `major`: Only major (startup and important error) messages.
- `support`: More granular information about good data too.
- `io`: Lowest level of detailed information (lots of data!)

The defaults for all blocks are major except for boot which is support. Logging levels are not saved across reboots.

The logger support tries to gracefully manage large impulses of data. It has a ring buffer currently set to 1024 bytes and if data comes in faster than the UART can empty it and the buffer fills, logging messages get dropped. While this is possible too with using the Ethernet port for logging, that should be tough to overflow.

The current functional blocks that have logging support are:

- `boot`: The boot process
- `mainapp`: This shows usage statistics mostly, once a second in support mode
- `ethapp`: Voter2 Ethernet support code
- `ethlwip`: LWIP (ethernet stack) code
- `ksz8081`: Ethernet PHY code

- `console`: Internal console parsing and character handling.
- `i2c`: I^2C comms debugging
- `dial`: DIAL server interface
- `gps`: GPS data and 1PPS interface
- `logtel`: Logger (logger, log thyself!)
- `local`: Local mode
- `ser2net`: Ser2Net interface
- `spinor`: SPI NOR interface
- `spimtr`: MTR-2000 SPI interface (not used by Voter2)

5.8.3.8 probe

While DSP is great for flexibility, it's really hard to probe the intermediate signals between the various DSP blocks in real time since they are just function calls inside a program. This command helps with that. At each stage in the Rx and Tx DSP chains there is a monitor function that if selected will show what the signal looks like at that stage and drive it to a second DAC so you can look at the signal at that stage on your oscilloscope. Pretty cool, eh? This is the command that selects where to probe. Parameters are:

- `none`: Probing disabled. This is the power-up default.
- `rxin`: Rx signal as digitized by the A/D (no DSPing yet)
- `rxdem`: Rx signal after de-emphasis
- `rxvoice`: Rx signal after the voice passband filter (300-2400Hz)
- `txsrc`: Tx signal as it came from the DIAL server post decompression. Or the locally repeated audio if in local mode (identical to `rxvoice` in this case)
- `txpl`: Tx signal after tone processing (PL and/or test tone)
- `txout`: Tx signal as it's going to the DACs. Well, not really, in making the code efficient, I don't have a dedicated buffer for just the DAC samples so this is the same as `txpl`.
- `rsssig`: The Rx signal after the high pass filter for RSSI determination
- `rsslvl`: The derived RSSI value from the RSSI energy detect. The "0-255" level is translated to 0-3.3V on the DAC for the entire packet.

5.8.3.9 flags

This sets a value for internal debugging flags. Unless you are a code maintainer or in close contact with one, best not to tinker with this command! :) The first parameter is the flag number and second parameter the flag value. While easily changed, there are currently 8 flags (0-7) and each flag can have a value of 0-255. The default values on power-up and reboot is zero for all flags and values are not preserved on reboot.

Their use (if at all) is purely for transitory internal debugging and likely will change with each code release. Debugging code may also change the value of a flag to something different than was set.

5.8.3.10 myname

This sets a name to be echoed at the command prompt, making it easier to know which Voter2 you are dealing with in a network with many of them. Good idea Dave!

5.8.3.11 errlog

This command manages Voter2 error logs, mostly for long term Ethernet traffic error tracking. It is for low level debugging, not for general users, so somewhat cryptic. It is an array of error counters for various internal functions. If any errors occur, a counter for that error gets incremented. Showing this parameter lists any non-zero counter values with just an error counter ID and count. "Setting" this parameter (no data) resets all the counters to zero. Note that these counter values are preserved across resets, so likely have garbage values on initial power-up. To be useful the counter values must first be manually reset to zero. Additionally, if the Voter2 has a battery installed, then these count values will even be preserved across power cycles! If you are curious, the index of IDs are in `ErrLog.h`. Locate the enum entry for a counter that is incrementing and then search for that enum in the source directory.

5.8.3.12 credits

Just what it says, the Voter2 sponsors and development team.

5.8.3.13 all

A quick and easy way to print out all the settings to capture and save. (show only)

5.9 show

The companion to the `set` command. See it for all the parameters.

5.10 status

This command is to show the status of different systems and interfaces of the Voter2. Nothing is settable. The items to show are:

- `14v`: Shows the status of the 14 Supply. It is just a comparator at 11.9V, so not very useful.
- `dial`: Shows the status of the connection with the DIAL server including all of the status byte settings.
- `gps`: Shows the status of GPS receiver including 1PPS.
- `level`: Shows a live portrayal of the incoming audio level. It is scaled to a 0-255 number with 255 being a full scale of the DAC input. The bar "tick" marks are in

units of 50. The display keeps on updating every 200ms until a key is received via the console.

- `power`: Shows the state of the MTR-2000s AC_FAIL signal. If the Voter is running and AC_FAIL is active, that means the repeater and Voter2 are running from battery power.
- `rssi`: Shows both the analog and high-pass computed RSSI values. The bar "tick" marks are in units of 50. The display keeps on updating every 200ms until a key is received via the console.
- `sync`: Checks the synchronization between the Voter2 and the DIAL server's time. This is not an accurate test and intended for a rough check only. Healthy numbers range from 500 to 1500us on a local network.
- `tempr`: Shows the internal temperature of the MCU as well as any external I2C temperature sensors. There is one on the board away from the MCU and others may be added off-board.

5.11 ttone

This controls the test tone generator. A sinewave with the specified parameters will be generated. When enabled, PTT is asserted overriding any Rx or DIAL audio and any timeout timers. Parameters are:

- `Off`: Turns off the tone generator if turned on.
- `Level`: The amplitude of the tone. If the tone generator is off, setting a level turns it on—even if set to 0 (silence). The value can be in direct DAC steps, 0-4096 or can be specified as a percentage of full scale if the number is followed by a % character.
- `Freq`: Frequency of the tone in Hz. This can be 1-4000.

If you enter ttone with no parameters, the current tone settings will be displayed.

5.12 update

This is the command to remotely update the Voter2 firmware. Refer to Appendix B for how to set up a TFTP server to support firmware downloads. The Voter2 currently needs some special tweaks to the TFTP server setup to work, so please read fully. The Voter2 uses a staging flash to hold the new image before programming the executable flash, that way it is not relying on anything other than stable power for the update process. Updating firmware is therefore a multi-step process: First is to check for the latest version on the server, second to download a new version into the local update flash, then finally to update the executable flash from the download flash. Update commands are as follows: (in the order you would run them)

- `check`: Check the update server to see if there is a newer version. The current version running on the Voter2 and on the server will be listed so you can compare them.

- `download`: Download the version noted in check command into local download flash. Note that the check command must be run first to get the version to download. This does not update the Voter2, it just stages the image into a staging flash to ensure we have a complete and verified image in local memory before doing the update.
- `verify`: Verify the code in the download flash matches what is available on the server. As with `download`, the check command must have been performed before this command. In theory this step isn't needed as the download command is also checking, but this is still handy to run. If you ever get a verify error, please let me know!
- `program`: Program the Voter2's executable flash with the version in the download flash. This will involve two reboots, one to initiate the update, then a second to boot up with the new version. Note that to be able to program, a download must have been performed since the last boot—and not erased! The entire process should take less than 90 seconds. If your console connection is Telnet, the connection will be lost due to the reboots.
- `erase`: Erase the download flash. No real reason to do this normally, it was mainly to assist in code development.
- `Dump [addr] [len]`: Dump a range of download flash memory. The address and length are both hex parameters, up to 8 hex characters. There is no real address checking. Also note that for this command the address is the native flash address, not where the address is mapped to in STM32 address space, so 0-4M for the Voter2, 0-32M for the S-Com.

5.13 xuart

The External UART is not currently being used by the Voter2, it is intended for the S-COM project. This command just runs a loop-back check on this UART. To pass, the external UARTs Tx and Rx lines should be connected together.

6 APPENDIX A – SER2NET SETUP

Editors Note 1: The instructions below are *a* method that works, but I'm quite confident it is far from the best way of making things work. There are many other RFC2217 UART-to-Telnet drivers out there and there may be some that are easier to set up and maybe even stay resident instead of the method below. My initial goal was getting *something* to work in order to verify my RFC2217 implementation with the real Motorola RSS software and the process below does that. I'm happy to receive and include better implementations.

Editors Note 2: I took notes, but not the best notes while iteratively figuring out the installation and test process. Now that the software is installed and running, I fear that the instructions may be missing some steps. If you are doing a fresh install and find out something is missing, please let me know.

The hardest part of course is finding a computer old enough to still have a 32-bit processor as that is the only CPU the Motorola RSS software will run on. It will most likely be running Windows XP. Luckily, a computer that old likely still had a serial port and you've been running the Motorola software using the standard UART interface.

The port redirection software recommended to me is com0com and hub4com. A place to download these is on sourceforge at:
https://sourceforge.net/projects/com0com/files/

Select hub4com, the 2.1.0.0 folder and download
hub4com-2.1.0.0-386.zip
Open this ZIP file and save the contents to a directory of your choosing.

Back at sourceforge, go back to the top then select com0com, the 3.0.0.0 folder and download the signed image:
com0com-3.0.0.0-i386-and-x64-signed.zip

That ZIP file contains two images, the x86 and x64 images.
Extract and run the x86 image.
Accept the defaults. Note the install process takes a while!
You will get a "Found New Hardware" popup

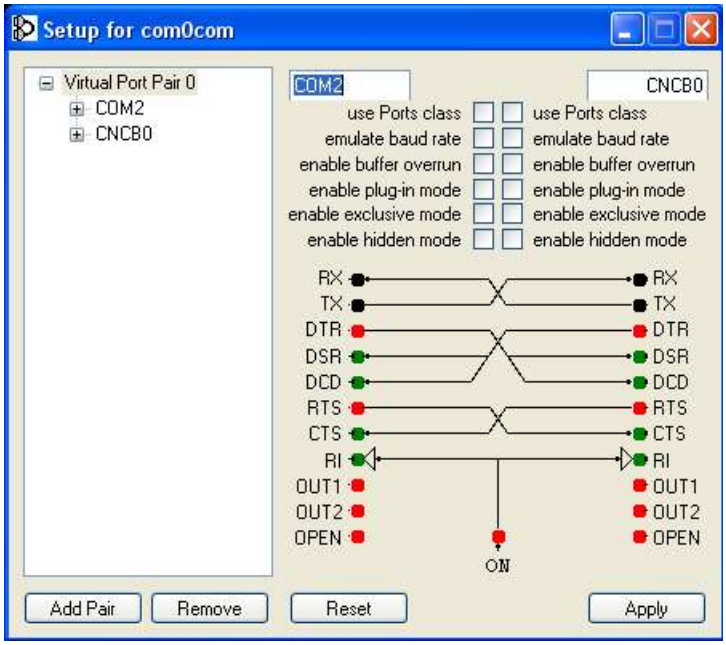
- Say "let window do the update automatically" (it works!)
- This happens four times.

Next, run com0com setup. This can be automatic at the end of the install or you can run it manually afterwards, it shows up as an application in the Windows start|all programs|com0com menu.

This brings up the graphical configuration screen for com0com. The default setup has two Virtual Port pairs. You only need one, so OK to delete Virtual Port pair 1 and work with Virtual Port pair 0.

For Virtual Port Pair 0, change the left port to an unused COM port on your computer. Note that the Motorola software only supports COM ports 1-4, so it must be one of those

Leave the right port as CNBC0. Leave all of the check boxes un-checked. In the end, the setup screen should look like this:



Click Apply and close.

Make sure your MTR-2000 and Voter2 are powered up and connected to the same network as the laptop. Note the Voter2's IP address (`show myactualip`).

Go to the folder you installed the hub4com files in to and open a command console in that directory. There should be a batch file called com2tcp-rfc2217.bat.
Enter the DOS command similar to below

```
com2tcp-rfc2217 \\.\CNCB0 192.168.2.163 2000
```

where 192.168.2.163 is instead your Voter2's IP address and if you changed the Voter2's Ser2Net port to something other than 2000, that new value.
If all goes well, a bunch of logging text will scroll by and will pause after a few seconds without any errors, *not* returning to the command prompt. The text is logging information regarding RFC2218 commands.

Leave that window alone and fire up the Motorola RSS software.
Select the COM port you chose during the com0com setup, (say a short prayer) and click on the codeplug download button. Hopefully it works!

After you have finished and closed the MTR-2000 software, just close the command box you ran the com2tcp batch file in.

Whenever you want to run the Motorola RSS software again, you don't need to re-run the com0com setup, but do need to pick up where you open the command console to the hub4com directory and re-run the com2tcp-rfc2217 batch file.

When I used similar software on my modern computer, I did not need to open the command window and run the com2tcp batch file every time, the settings stayed resident. I still hope this is possible with Windows XP as that would mean just firing up the computer and running the Motorola RSS software. I haven't figured this out yet.

7 APPENDIX B – TFTP UPDATE SERVER SETUP

TFTP support for the Voter2 is not fully standard—yet. A special tweak is necessary to the TFTP server setup to work with the Voter2. Why is noted later in this appendix, but the practical requirement is the "randomly" assigned download port called out in the TFTP RFP needs to be a fixed port for the Voter2.

For development purposes, the TFTP application chosen was OpenTFTPServer for Windows:
https://sourceforge.net/projects/tftp-server/
For that project I used the 64-bit multi-threaded version, but fairly confident the other versions will work too. If you get a different TFTP server working, particularly for a different OS, please let me know and I will add it to this appendix.

Installation is pretty straightforward other than the install directory is in c:\. I opted for setting up the TFTP server to load automatically on boot, you may not want to do so. By default, the home directory for downloads is also c:\.

The only parameter in the `OpenTFTPServerMT.ini` file that needs changing from defaults is `port-range`. Remove the tick in front of it and change it to:
`port-range=30000-30000`
This fixes the download port range to just that port number. Note that a tick mark is also a comment, so all the parameters preceded by a tick are not actually parameters!

That is all that is needed to set up the TFTP server itself, next is to set up the files that the Voter2 is expecting to see.

The first file is the download information file. For the Voter2-2K it is `Voter2-2K.txt`. As other hardware variations of the Voter2 get released (such as the Voter2-SA for the standalone version), they will have a config file name matching the product name. For reference, this is name is noted in the source `config.h` file and reported when you log into the console.

The information file contains a single line with 4 space-separated parameters as in this example:
`0.00C 2025-06-22 Voter2B2-0.00C.srec 4D73`
The first parameter is the firmware version. This should match the version in the `config.h` file for the build. By convention, the letter at the end will be C for candidate releases for that version and then R for the final release of that version. After the version is the date. Again this is noted in the `config.h` file and noted just for reference. The third parameter is the actual filename of the download. Finally (and yet to be implemented) will be the CRC32 of the download file.

This setup should allow this single TFTP server to support multiple Voter2 versions and always present the latest version to the Voter2's `update check` command.

REASON FOR THE NON-STANDARD TFTP IMPLEMENTAION:

I hope this "fixed port" work-around is temporary, but the cause is a fairly fundamental disconnect between the TFTP RFP and the way the LwIP drivers work. The issue, to me is the TFTP RFP, it seems very non-standard!

The issue is that while the client (Voter2 in this case) contacts the server at a standard port from its own "high numbered" port, when the server responds, it does not respond from its standard port, but instead from its own random "high numbered" port. Pretty fundamental to IP traffic is that the server responds back to the client using its standard port it got the request from. That way the client knows the response that came back was in response to its request. This is absolutely required for TCP traffic, but evidently optional for UDP traffic. The LwIP netconn drivers (the ones I am using) make that inherent assumption the reply is from the same port. When the TFTP server responds on a different port than the request was sent to, it drops the response assuming it was for some other connection.

The work-around is to fix the TFTP servers "random" address to a specific one, then I hack into the LwIP context structure to change the servers standard port number to 30000 so it will accept the return packet. Not great, but it works.

The assumption on preserving the sever port number is built into the LwIP netconn drivers. The only well documented fix is to use the raw drivers instead. The characteristic of the raw drivers is the LWIP stack just gives all incoming packets to the application and lets the application sort them out. This would then apply to all IP traffic for the Voter2, not just the TFTP traffic. I am using the regular netconn drivers for all the other IP traffic on the Voter2, so not a great option. There is a netconn connection option type called `netconn_raw` noted in the LwIP APIs, but I cannot find any documentation on it. I hope this will be the long term solution.