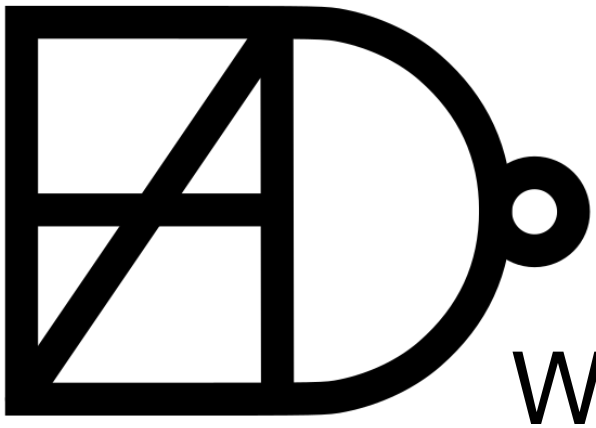


WA0EDA

STM32-DVM-MTR2K v1c-a and v2.0a

Revision: 20190824

Table Of Contents

Copyright © 2019 The Regents of the K0USY Group
This document may be reproduced and distributed provided it is not altered in any way. Alteration and redistribution or partial duplication and distribution is expressly prohibited without permission.

The WA0EDA Skunkworks may be contacted at:
5505 Plymouth Drive
Lawrence, KS 66049
n0mjs@me.com
(telephone support not provided)

Disclaimers and Conditions of Use

3rd Party Software Disclaimer

The STM32-DVM-MTR2K contains various 3rd party softwares redistributed under the GPLv3.0. WA0EDA and the Regents of the K0USY Group makes no claims or warranties regarding such software and we are not responsible for its use. Using the STM32-DVM-MTR2K requires acceptance of and compliance with the license terms of included, 3rd party software.

Use at Your Own Risk

The STM32-DVM-MTR2K, as a whole unit, is not UL (or similarly) listed, and has not been tested for FCC Part 15 (or other) compliance. By using the STM32-DVM-MTR2K, the user agrees to indemnify the WA0EDA Club, the Regents of the K0USY Group, Cortney T. Buffington, N0MJS and other affiliated entities against liability resulting in its use.

Configuration Disclaimer

WA0EDA provides configuration advice, but is not responsible or liable for providing comprehensive or authoritative information about MMDVM or the MTR20000. Configurations presented in this manual represent how we set up MMDVM and the MTR2000 with the STM32-DVM-MTR2K. This manual does not attempt to cover alternative configurations, and is not an authoritative source of information for MMDVM or the MTR2000.

Version 1c-a and 2.0a

This manual contains instructions for multiple versions of the STM32-DVM-MTR2K. Primary features of both v1c-a and v2.0a versions are functionally identical. The v2.0 board adds additional circuitry and software to:

- Duplicate specific features of the Motorola Auxiliary I/O board Part Number CLN6698
- Exposes i2c bus connectors for the NanoPi NEO and ATmega 328P microcontroller
- Adds access to NanoPi NEO /dev/ttyS2 for an MMDVM serial display
- Provide rudimentary communication between the NanoPi NEO and MTR2000 via MTR20000 GPIO

Unless specifically stated, information in this manual applies to both versions.

Overview

The STM32-DVM-MTR2K is an integrated MMDVM modem and Host designed to plug into either of the MTR2000 Option Board slots (J1, J2) in the MTR2000 card cage. The STM32-DVM-MTR2K is an all-in-one solution for adding MMDVM capabilities to the MTR2000, no additional hardware or software is necessary.

The modem section is typical of most MMDVM hardware interfaces in that it includes 3rd order 4kHz low-pass filters for the TX & RX audio paths, and signal conditioning for logic signals. The STM32-DVM-MTR2K shares a common design heritage with the Repeater Builder STM32-DVM from Scott Zimmerman, N3XCC; including an identical STM32F446RET6 microcontroller operating on the same firmware as the Repeater Builder version 3 “blue” board.

Unique to the STM32-DVM-MTR2K is the inclusion of a host SBC (single-board computer) permanently affixed to the card and no mechanical potentiometers for calibration; all level adjustments are performed in the MTR2000 RSS. The SBC provided on the STM32-DVM-MTR2K is a NanoPi NEO from FriendlyElec, and comes pre-loaded with Armbian Linux and a fully functioning, start-on-boot MMDVMHost installation.

Programming access to the STM32F446RET6 microcontroller is provided for those who wish to use alternative or updated firmware. The NanoPi NEO comes with 512MB of RAM and a 32GB micro SDHC card which may also be re-used for customers wishing to use their own builds or other 3rd party images, such as PiStar.

Physical and Electrical Specifications

Size:	160mm x 100mm (connectors excluded)
Radio Connector:	96 pin male “eurocard” (3 x 32)
Voltage:	10-18VDC*****
Current:	2A maximum, 150mA typical (180mA typical v2.0)
Network:	RJ45 10/100Mbps Ethernet, auto-sensing
Logic Outputs:	Open Collector 50VDC max, 100mA max sink
Logic Inputs:	Transistor Buffered, 50VDC max
RSSI Input:	3.3VDC maximum useable
Audio Input/Output:	AC Coupled, preconfigured for MTR2000
Host SBC:	NanoPi NEO 512MB RAM, 32GB micro SDHC
Modem processor:	STM32F446RET6
MMDVM Indicators:	Power, Activity, Heartbeat, PTT, RX Clip, COR, DMR, P25, NXDN, YSF, D*
I/O processor:	Microchip/Atmel ATmega328P (v2.0)
I/O indicators:	4 red LEDs for outputs, 4 green LEDs for inputs (v2.0)

***WARNING: NEVER APPLY POWER DIRECTLY TO THE +5VDC BUS ON THE STM32-DVM-MTR2K, INCLUDING THE MICRO USB POWER PORT ON THE NANOPI NEO. SEVERE DAMAGE MAY RESULT.**

NanoPi NEO Access

The STM32-DVM-MTR2K is shipped with a micro SDHC memory card installed in the NanoPi NEO running a customized version of Armbian Linux that includes pre-built MMDVMHost, MMDVMCal (renamed CalMMDVM) and utilities necessary for operation and maintenance. There is no host firewall configured on the pre-built version. The NanoPi NEO may be accessed over the network by secure shell (SSH). Writes to the SDHC memory card are buffered for up to 10 minutes. This saves wear and tear by flushing changes to the card much less frequently. This behavior is normally transparent to the end user, but be sure that you properly shutdown or reboot the NanoPi NEO after making changes to ensure all changes are written to the card. Making changes and immediately pulling the power will cause the changes to be lost.

As shipped, the SSH server is listening on port 32 (not the standard SSH port, 22). Direct root logins via SSH are also disabled. The as-shipped configuration is:

- SSH Port: 32
- Username: mmdvm
- Password: mmdvm
- Root Password: mmdvm

The default mdns name, as well as the hostname that will be reported to your DHCP server is “stm32-dvm-mtr2k”. Use this to find the IP address assigned by your DHCP server, or simply use mdns if supported on the system your system (i.e. “ssh -p 32 mmdvm@stm32-dvm-mtr2k.local).

MMDVM Configuration

MMDVM configuration requirements for the STM32-DVM-MTR2K and MTR2000 are minimal. Specifically, the following “Modem” stanza from the MMDVM.ini (contained in /etc/ on our Armbian build) shows recommended values.

```
[Modem]
Port=/dev/ttyS1
Protocol=uart
TXInvert=0
RXInvert=0
PTTInvert=0
TXDelay=100
RXOffset=0
TXOffset=0
DMRDelay=165
RXLevel=50
TXLevel=50
RXDCOffset=0
TXDCOffset=0
RFLevel=0
RSSIMappingFile=/usr/local/etc/rssi.dat
Trace=0
Debug=0
```

The DMRDelay setting is absolutely critical. In early experimentation with MMDVM and the MTR2000 it was found that a setting of 162-168 which correlates to about 7ms of delay is an appropriate setting (Graziano Rossi, IZ5IGB, 2016). We empirically verified this, and in all of the MTR2000s tested, 165 +/- 2 is the midpoint of acceptable values for accurate decoding.

This setting is necessary because the MTR2000 does not have an analog FM detector. The receiver 2nd IF is digitized into PCM data, which is sent from the receiver to the SCM (system control module) where it is converted into an analog signal in CODEC #5. This creates a delay in the audio path as the ADC and DAC conversions create latency. Since DMR is TDMA, the timing between when a DMR burst is sent and when it is acknowledged is critical. The DMRDelay setting is used to compensate for this latency and time-align RX and TX bursts.

If the DMRDelay is wrong, the problem is usually easy to spot. The Modem will show a downlink activation from the subscriber unit, but because timing is wrong, the subscriber will never synchronize with the repeater. Log entries like the ones shown below (on key-up) are typical of an incorrect DMRDelay.

```
D: 2019-06-14 14:44:34.358 Downlink Activate CSBK
D: 2019-06-14 14:44:34.358 0000:  B8 00 00 00 FF FF FF 2F 9B E5 DF 60
M: 2019-06-14 14:44:34.358 Downlink Activate received from 3120101
```

The other, optional task, is RSSI. This is by no means mandatory, but remains a popular feature within MMDVM. For RSSI mapping, the MMDVM.ini requires a user supplied mapping file with measured values of the MMDVM RSSI ADC input, and the corresponding signal level in dBm. Not everyone has the means to measure this. We include a generic mapping file from a typical MTR2000, but there is some variation from station to station. If you don't have the means to make your own, or absolute accuracy isn't important, the numbers below (also contained in our supplied rssi.dat file) should work reasonably well:

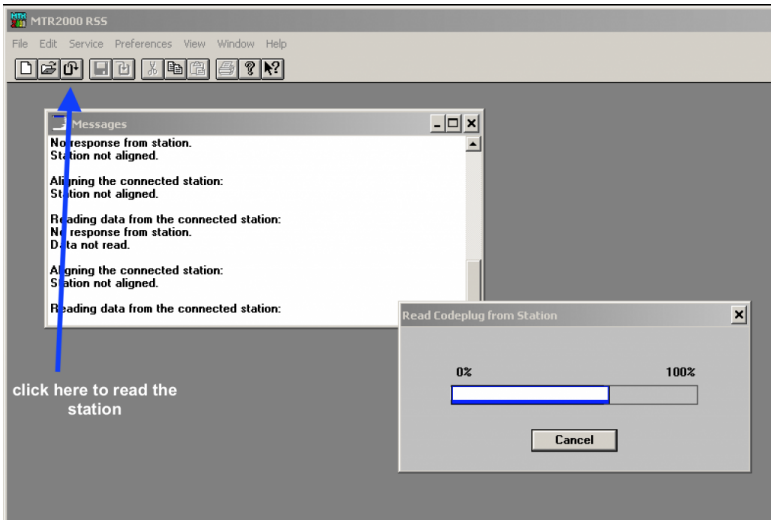
```
3226 -30
3128 -35
2992 -40
2812 -45
2657 -50
2454 -55
2320 -60
2123 -65
1986 -70
1785 -75
1650 -80
1447 -85
1313 -90
1112 -95
980 -100
778 -105
643 -110
404 -115
240 -120
94 -125
50 -130
```

The STM32-DVM-MTR2K will saturate the MMDVM ADC input at signals stronger than ~= -30dBm. We chose this as a compromise in order to maintain better ADC resolution and accuracy for weaker signals at -115dBm and lower – where it matters most.

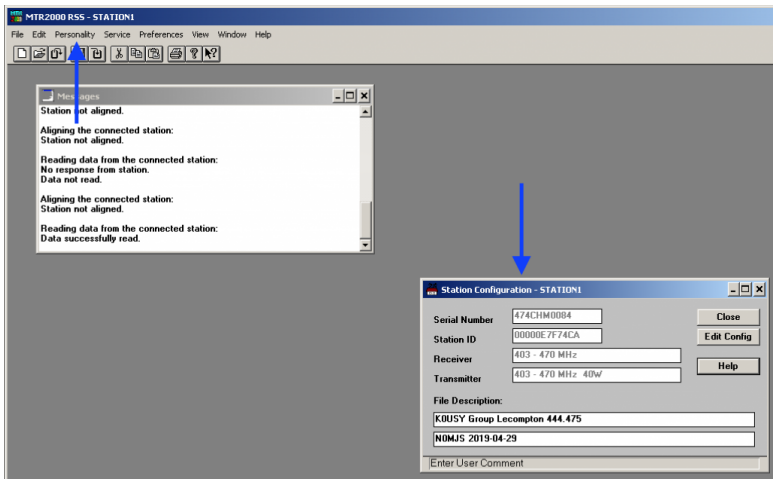
MTR2000 Configuration

Channel Settings

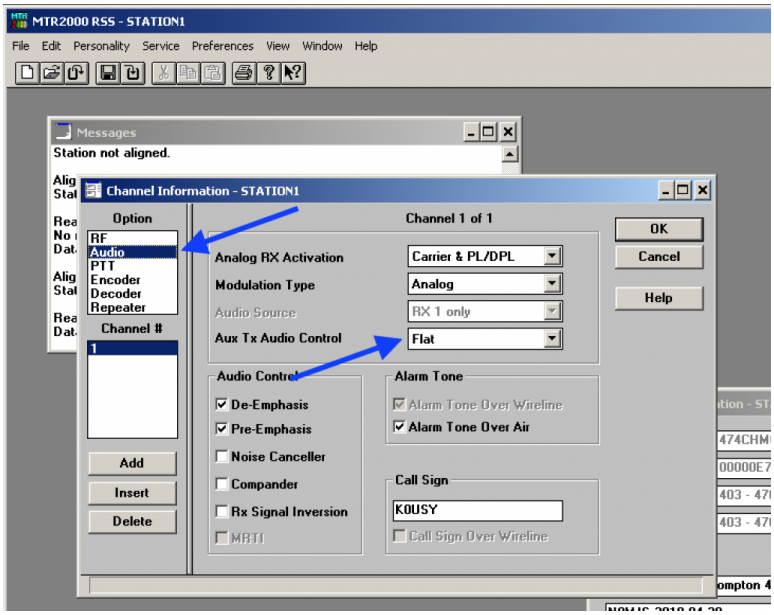
WA0EDA recommends you program the channel information into your station first, and then perform audio calibration. A number of settings shown in the included screenshots do not matter for MMDVM operation. If a particular parameter is not discussed, it is not important. Begin by reading in your station's configuration (codeplug):



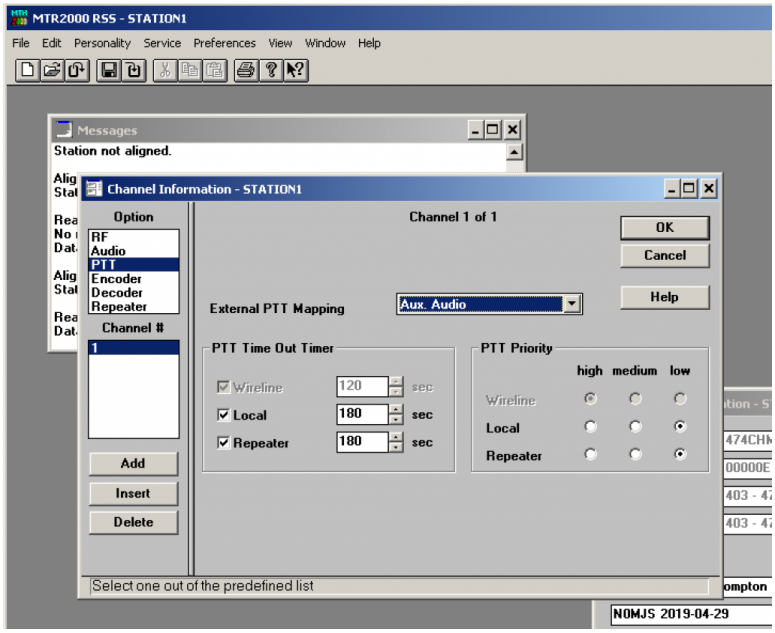
There are two items that must be set: "Aux Tx Audio Control" and "External PTT Mapping". Once the station is read, a new "Station Configuration" window will appear, and a new menu item called "Personality" will show up between "Edit" and "Service" in the RSS menu bar. The personality menu is only present when a Station Configuration window is active.



Under the “Personality” menu, select “Channel Information”. A new window will appear. In the upper left area of the window is a section box marked “Option”. From this list we’ll be working with the “Audio” and “PTT” menus. In the Audio settings, “Aux Tx Audio Control” must be set to the “Flat”. This directs the station to NOT pre-emphasize or limit the Aux Tx Audio input.



Under the “PTT” Option menu, External PTT Mapping must be set to “Aux. Audio”. This directs the station to use the Aux Audio input as the transmit audio input when the “External PTT” is engaged.

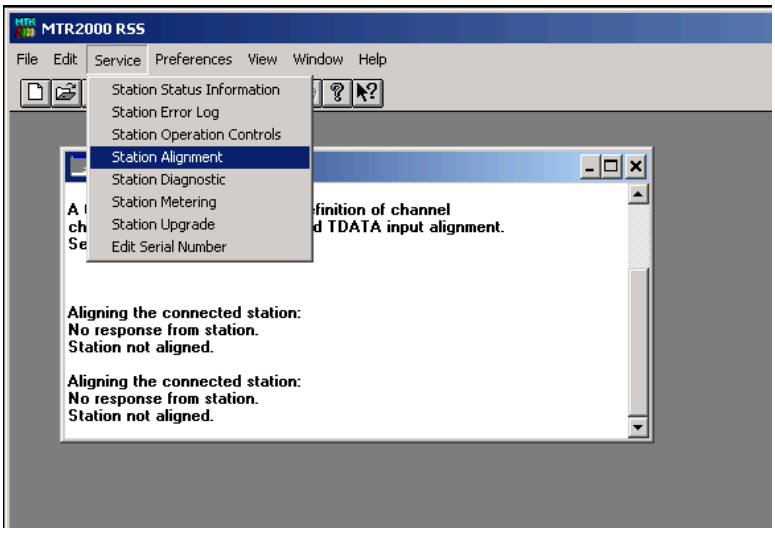


Make sure to click “ok” then write the station. None of the settings are stored until they are written back to the station. Upon completion of the write operation, the station will reset. Of course, other

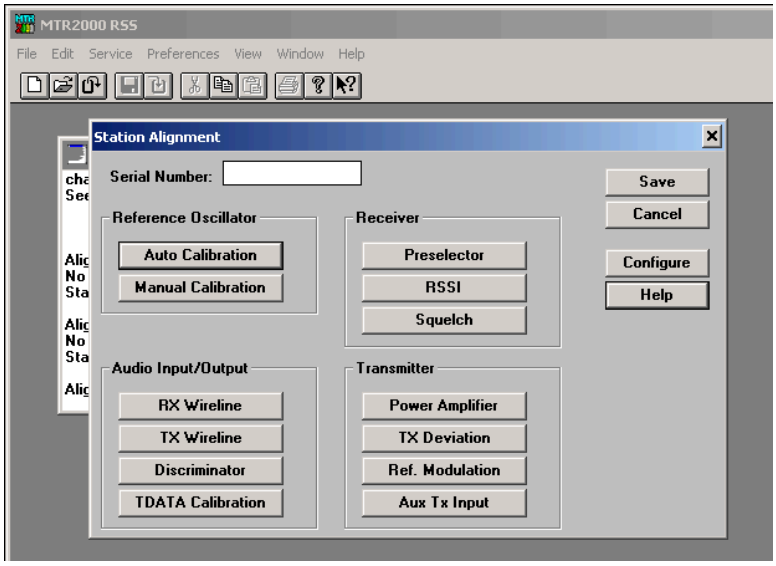
settings are required – most notably setting the TX and RX frequencies, channel bandwidth (5kHz) – but general MTR2000 programming is beyond the scope of this manual. Please refer to the MTR2000 station and RSS documentation.

Level Calibration

The STM32-DVM-MTR2K contains no trim pots. All configuration is achieved with soft pots in the MTR2000. There are two mandatory calibration items: RX audio, TX audio. To set the TX and RX audio levels levels, navigate to the “Service” menu in the RSS and select “Station Alignment”:



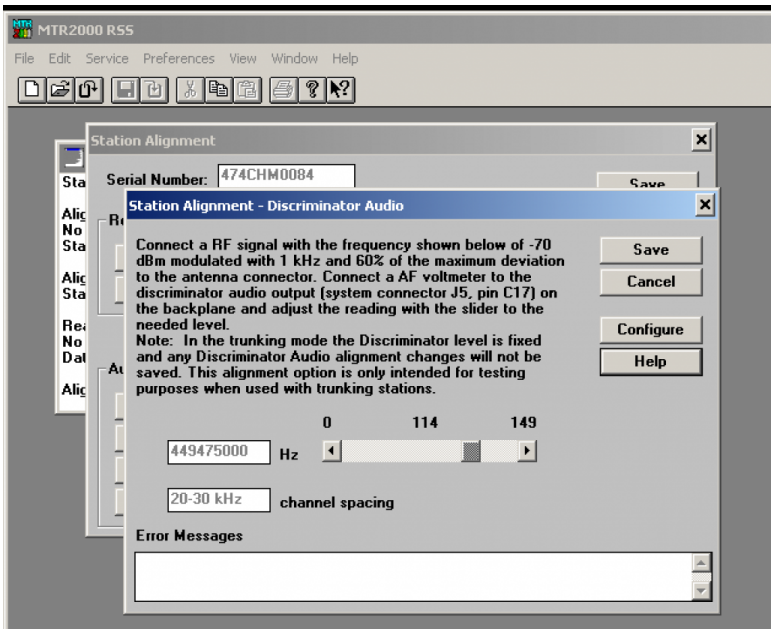
It will take a few seconds as the RSS reads alignment information from the station. To adjust RX audio, select the “Discriminator” button under “Audio Input/Output”.



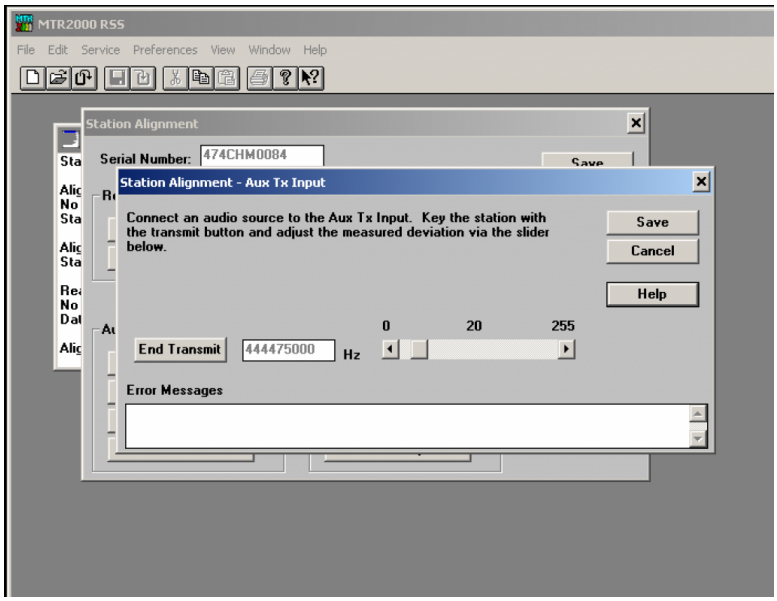
STM32-DVM-MTR2K analog conditioning circuitry is optimized for the MTR2000, which makes alignment very simple. With no input signal present, the MTR2000 Discriminator output level should

be increased until the red CLIP LED on the STM32-DVM-MTR2K begins to light. Then back the level down until the CLIP LED extinguishes. This is the only RX calibration required.

On all of the MTR2000s we've set up for the STM32-DVM-MTR2K, the levels have been between 109 and 114. Starting somewhere in this area will get you to the right level faster.



The TX audio adjustment is very similar and is accomplished by selecting the "Aux TX Input" adjustment in the "Transmitter" alignment section.



Note: When adjusting Aux TX Input, the slider will not appear until you click the “Transmit” button. The slider appears when the station is keyed. The final result will be around 26, making that a good place to start.

Use MMDVMCal (renamed CalMMDVM in our Armbian image to make tab-completion more effective) and either set TX deviation with the DMR test tone as described with the MMDVM documentation, or use a spectrum analyzer (25kHz sweep works well) to set modulation as close to Bessel zero (nulling the signal at the carrier center frequency) as possible. You will not achieve a perfect “null” using the Bessel zero technique as the soft pot’s steps are too coarse to do this. But if you note on a deviation meter how much change there is, between a perfect null and something close to it, you’ll realize that the perfect null isn’t necessary.

Make sure to use the “Save” buttons both on the individual alignment windows and then on the main station alignment window as well. Calibration values are not permanently written into the station until the main Station Alignment “Save” button is clicked. You will know that you’ve successfully written the station when you see it reboot:

RSSI is fixed and requires no calibration. WA0EDA has supplied a mapping file with our software load, but for absolute accuracy, we recommend creating one specifically for your station. You may note that signals about -30dB will saturate the RSSI ADC input – this is intentional and intended to provide greater accuracy in RSSI measurement for weak signals of -115dB and lower.

Modem Firmware

STM32-DVM-MTR2K boards are shipped with a mature version of MMDVM. Generally, it should not require firmware updates, but at some point, features may be added and users may wish to update firmware. The STM32-DVM-MTR2K is equipped with the means to update firmware in the field (i.e. remotely), or with manual intervention by the operator. In either case, the on-board NanoPI NEO is used to upload firmware into the STM32F446RET6 microcontroller. For those with any reservations about performing a firmware update, we strongly recommend using the manual method, at least the first time, as this method is somewhat simpler.

As the STM32-DVM-MTR2K shares a common design heritage with the Repeater Builder STM32-DVM, the procedure is almost identical to that listed on the Repeater-Builder FAQ for the Version 3 PiHat (blue board). All utilities needed to perform these procedures are included on the micro SDHC card supplied with your STM32-DVM-MTR2K, and all procedures can and should be performed from the mmdvm user’s home directory. If you have changed the software, a more

detailed article, including information on the necessary utilities, for uploading firmware is available at the KS-DMR website:

http://ks-dmr.net/2019/05/28/stm32-dvm-mtr2k-deep-dive-updating-firmware

Binary firmware files are maintained by Steve, N4IRS, and Mike, N4IRR, at DVSwitch. Their current version may be downloaded at under the “Version_3_Firmware” link at:

http://dvswitch.org/files/HAM/MMDVM

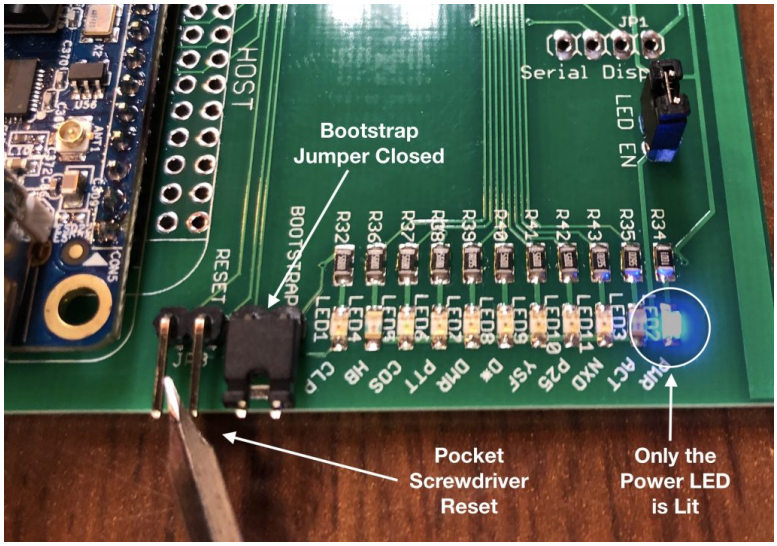
It is fastest to download this directly into the NanoPi NEO. Log in as the mmdvm user and in the home directory (where you are after logging in) use wget to download the firmware:

```
wget http://dvswitch.org/files/HAM/MMDVM/Version_3_Firmware/mmdvm_f4.hex
```

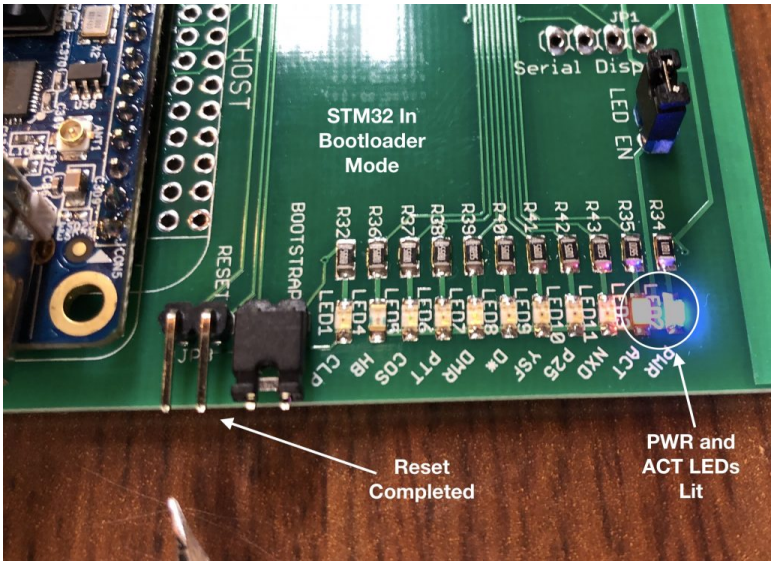
Before firmware can be uploaded, the STM32F446RET6 microcontroller must be placed in bootloader mode. There are two ways to do this: Physically, with jumpers, and via software with the GPIO pins of the NanoPi NEO. Before proceeding with either method, please ensure the MMDVMHost process is terminated. If MMDVMHost is running during the firmware update process, the update will fail, and the bootloader in the microcontroller may be damaged.

Manual Method

Place the STM32F446RET6 into bootstrap mode by placing a jumper on JP2 “BOOTSTRAP” while momentarily shorting the pins of JP3 “RESET”.



Note that while the board is being reset only the Power LED should be illuminated. Once the board has successfully been reset and is in bootstrap mode, the Power and Activity lights will remain lit constantly. Actually, the DMR and COR LEDs are also lit very dimly; it may not be possible to see them lit at all in a well lighted room.



After the STM32F446RET6 enters bootloader mode, send the following command at the NaoPi NEO's shell prompt:

```
stm32flash -v -w ./mmdvm_f4.hex -R /dev/ttyS1
```

The process takes some time to complete. Specifically while the stm32flash utility is erasing the flash memory on the STM32F446RET6, there is no feedback to the user. That's ok; if you've made it this far, the processing is working correctly. After erase, stm32flash will program and verify the onboard flash memory then reset the STM32F446RET6. Output should be similar to the following:

```
stm32flash 0.5

http://stm32flash.sourceforge.net/

Using Parser : Intel HEX
Interface serial_posix: 57600 8E1
Version      : 0x31
Option 1     : 0x00
Option 2     : 0x00
Device ID    : 0x0421 (STM32F446xx)
- RAM        : 128KiB  (12288b reserved by bootloader)
- Flash      : 512KiB (size first sector: 1x16384)
- Option RAM : 16b
- System RAM : 30KiB
Write to memory
Erasing memory
Wrote and verified address 0x0800fea0 (100.00%) Done.

Resetting device... done.
```

After upload is completed you should see the normal power-up sequence on the STM32-DVM-MTR2K modem status LEDs. Firmware update is now complete and you may restart MMDVMHost.

The most common problem is a failure of the stm32flash utility to properly initialize the STM32F446RET6 microcontroller. By far, the most common causes are that the microcontroller is either not in bootloader mode, or MMDVMHost was either running or started while the microcontroller was in bootloader mode. An initialization failure will produce an error such as this:

```
stm32flash 0.5

http://stm32flash.sourceforge.net/

Using Parser : Intel HEX
Interface serial_posix: 57600 8E1
Failed to init device.
```

In-Field (Remote) Method

The STM32-DVM-MTR2K includes necessary connections between GPIO (General Purpose Input/Output) lines on the NanoPi NEO and the STM32F446RET6 microcontroller for placing the micro controller into bootstrap mode, without the use of jumpers. A script, “update_modem.sh” has been provided that will automate the procedure. The script expects to be run from the same directory as the firmware file, but does need you to tell it the firmware file’s name. Assuming the firmware file name is “mmdvm_f4.hex”, the following command will update firmware in the modem:

```
./update_modem.sh mmdvm_f4.hex
```

The script works using the “gpio” command line utility from the WiringPi utility package to manipulate GPIO (placing the microcontroller into bootloader mode) before running the stm32flash utility. The sequence is as follows:

- Set GPIO pins to output:
  - gpio mode 7 out
  - gpio mode 1 out
- Set the BOOTLOADER line high (pin 7), toggle the RESET line (1), Set BOOTLOADER low
  - gpio write 7 1
  - gpio write 1 0
  - gpio write 1 1
  - gpio write 7 0
- Set GPIO pins to input:
  - gpio mode 7 in
  - gpio mode 1 in

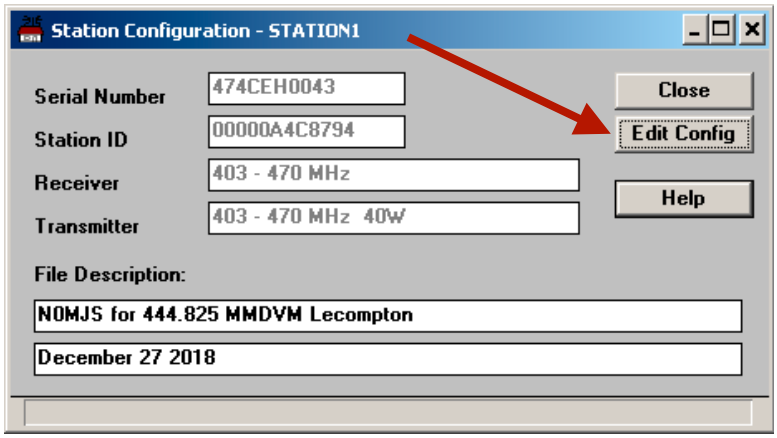
If completed successfully, the script will provide output similar to this:

```
Setting Up GPIO Pins
Sending STM32 Device Into Bootloader Mode
Resetting GPIO Pins
Attempting to program STM32 device
The following output is from stm32flash:

stm32flash 0.5

http://stm32flash.sourceforge.net/

Using Parser : Intel HEX
Interface serial_posix: 57600 8E1
Version      : 0x31
Option 1     : 0x00
Option 2     : 0x00
Device ID    : 0x0421 (STM32F446xx)
- RAM        : 128KiB  (12288b reserved by bootloader)
- Flash      : 512KiB (size first sector: 1x16384)
- Option RAM : 16b
- System RAM : 30KiB
Write to memory
Erasing memory
Wrote and verified address 0x0800fea0 (100.00%) Done.

Resetting device... done.

stm32flash completed
```

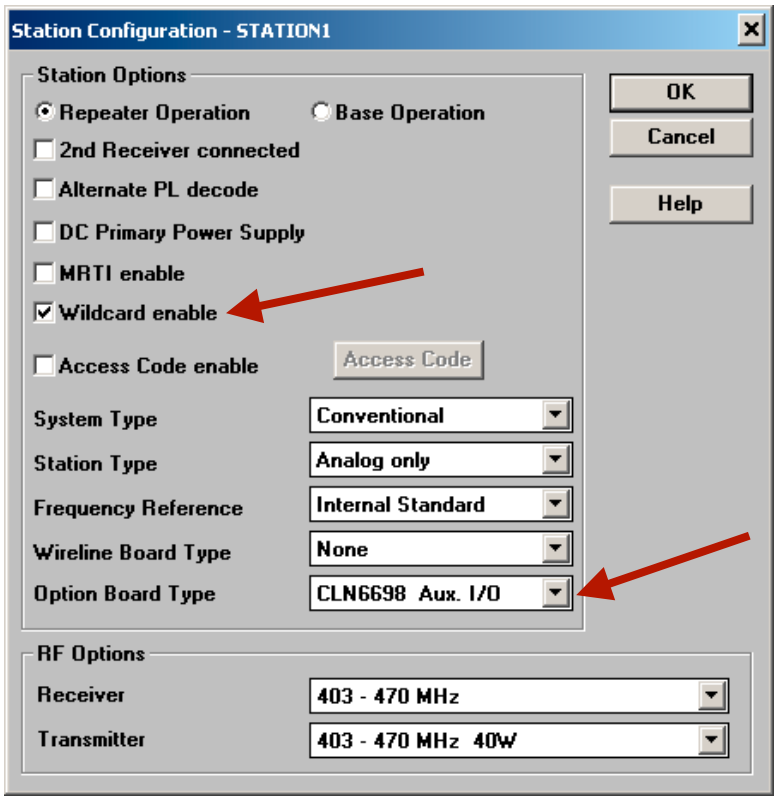
Errors with the stm32flash utility will resemble those from the manual method section.

Auxiliary I/O
(VERSION 2.0 ONLY)

The STM32-DVM-MTR2K V2 features the ability to replicate many features of the Motorola Auxiliary I/O board CLN6698. The V2 board will identify to the MTR2000 SCM (System Control Module) as a CLN6698, which will allow wildcard features to be enabled on the station (done through the MTR2000 RSS). In addition, buffers have been added to provide 4 general purpose inputs (GPIs) and 4 general purpose outputs (GPOs). To configure the station to to use the Auxiliary I/O features, read the station into RSS and select “Edit Config” from the Station Configuration window:



Two configuration items are required; “Wildcard enable” must be checked and “Option Board Type” must be set to CLN6698 Aux. I/O”:



The MTR2000 SCM communicates with station modules (including the CLN6698 Auxiliary I/O board) via a Serial Peripheral Interface (SPI) synchronous serial bus operating in SPI mode 3 with 16 bit datagrams. The original CLN6698 exposed two separate SPI “slaves” to the bus, one for the inputs and one for the outputs. In each case, two 8-bit shift registers connected in series forms the hardware interface to convert parallel I/O (GPIO lines) to serial data on the SPI bus.

On the STM32-DVM-MTR2K v2.0, GPIO processing is implemented in an Microchip/Atmel ATmega328P microcontroller. This approach has several advantages:

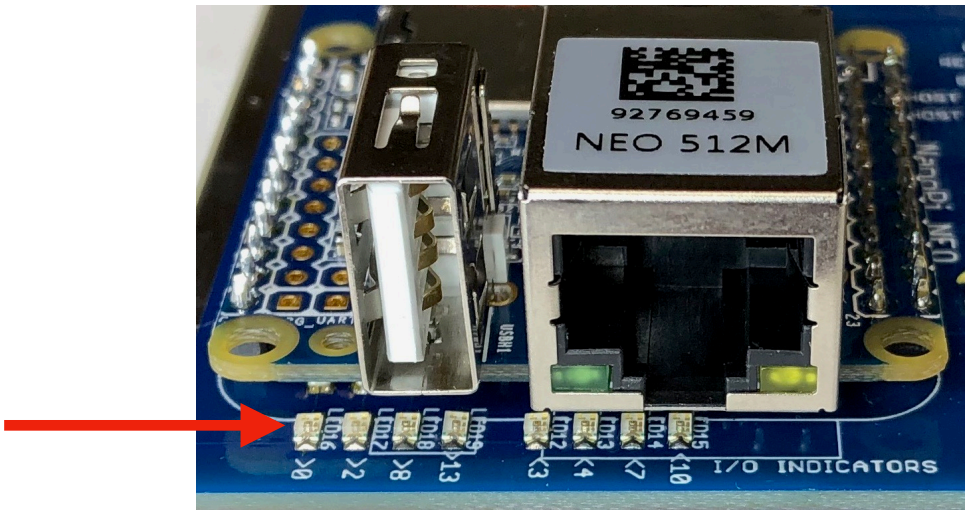
- Future upgrades can be made with a software release
- End users may write their own software
- More advanced interfaces can be achieved (such as GPIO communication with the NanoPi NEO in addition to the physical GPIO pins)

The STM32-DVM-MTR2K V2 hardware provides physical GPIO for:

INPUTS	OUTPUTS
GPI_3	GPO_0
GPI_4	GPO_2
GPI_7	GPO_8
GPI_10	GPO_13

These I/O lines are presented as open collector outputs and transistor buffered inputs. I/O actions are defined by creating wildcard tables in the MTR2000 RSS.

LED indicators for the GPIO lines have been provided under the front end of the NanoPI NEO. Left to right, red output LEDs labelled: >0, >2, >8, >13 and green input LEDs labelled: <3, <4, <7, <10 illuminate when active:



GPO_2 and GPI_3 lines may be configured to perform a special function. When enabled (by closing jumpers JP5 & JP6 in v1; jumper JP8 in v2) and providing a appropriate wildcard table configuration, the MTR2000's internal controller and MMDVM can gracefully co-exist for mixed analog and digital operation. When disabled (no jumpers installed), GPO_ and GPI_3 function as any other GPIO pins. A complete tutorial on analog + digital integration with the internal MTR20000 controller may be found on the STM32-DVM-MTR2K information page at:

http://ks-dmr.net/stm32-dvm-mtr2k-information/

Note: Inputs are NOT debounced. This feature may be added in a future software release. Care must be taken if connected to mechanical contacts to avoid the inputs triggering multiple times.

Finally, a serial connection is provided between the NanoPi NEO (/dev/ttyS2) and the digital I/O processor (ATMega328P) that may be used for programming the ATmega328P (it is shipped with the Arduino UNO serial boot loader installed), but may also be used for communication between the two devices. This is intended to provide a rudimentary communication bridge between the NanoPi NEO and the MTR2000 System Control Module (SCM) via the MTR2000 GPIO pins. Future software updates will allow the NanoPi NEO to fully interact with the GPIO function.

Bus Expansion

(VERSION 2.0 ONLY)

To create a more extensible system, two communication connections are provided on the STM32-DVM-MTR2K v2:

- NanoPi NEO i2c bus (3.3v, pull-ups installed)
- ATmega328P i2c bus (5.0v, pull-ups installed)

The i2C busses on the NanoPi NEO and ATmega328P are not used by either sub-system, and are completely available for end-user additions/customizations. They have been connected to unused GPIO pins on the MTR2000 System Connector available on the back of the station.

See the information in section V2.0a Jumpers and Connections for pin assignments.

For More Information

Additional information, updates, detailed articles, etc. regarding the STM32-DVM-MTR2K can be found in the "Shop Talk" section of the KS-DMR website (http://www.ks-dmr.net).

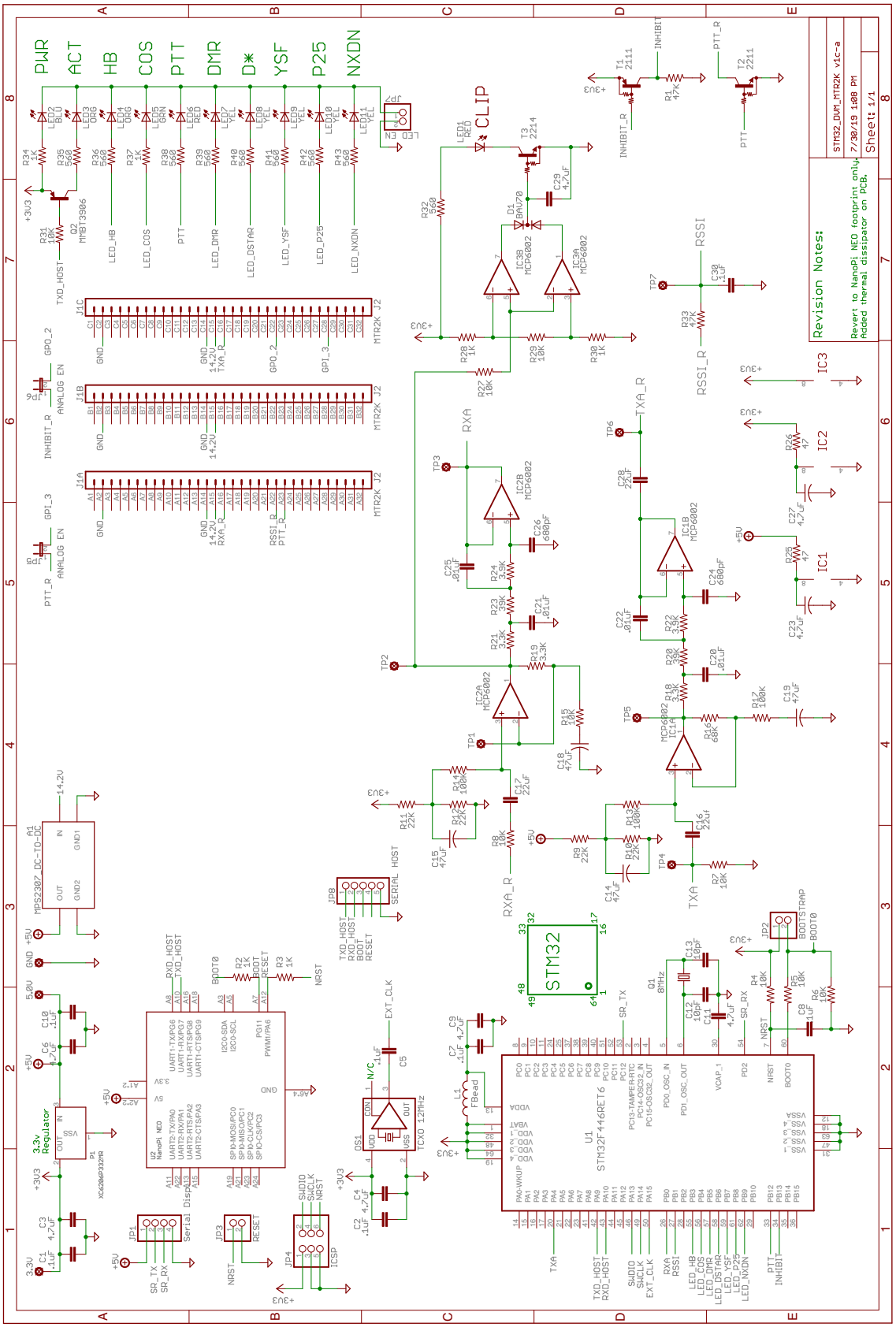
WA0EDA Skunkworks is the R&D subsidiary of the K0USY Group (http://www.k0usy.org).

The K0USY Group is the author of popular DMR networking software packages HBlink, DMRlink and dmr_utils (http://github.com/n0mjs710), operates many networked DMR repeaters in Northeast and Northcentral Kansas, provides hosted DMR networking services, and is a founding member of KS-DMR.

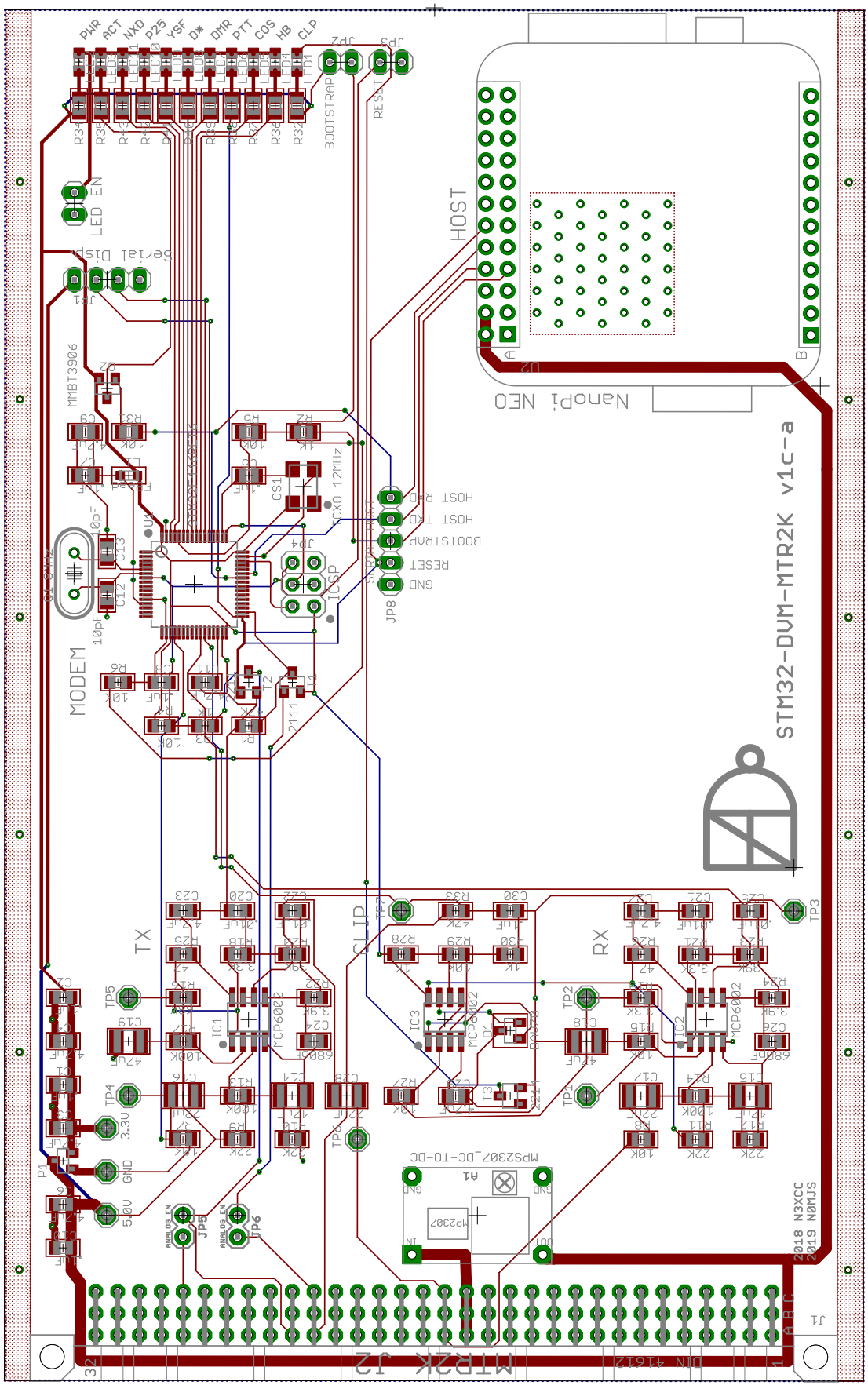
V1c-a Jumpers and Connections

ID	Name	Purpose	Type	Description
J1	MTR2K J2	MTR2000	96pin Euro	Connects STM32-DVM-MTR2K to the MTR20000
JP1	Ser Disp.	Peripheral	1x4 .1” pin header	Serial “repeater” connection via modem
JP2	BOOTSTRAP	Programming	1x2 .1” pin header	Used to place the modem into bootloader mode
JP3	RESET	Programming	1x2 .1” pin header	Used to reset the modem, used with bootloader mode
JP4	ICSP	Programming	2x3 .1” pin header	In circuit serial programmer for the modem
JP5	ANALOG EN	Analog handshaking	1x2 .1” pin header	Connects PTT_R to GPI_3 - used with wildcard to knockdown in-cabinet repeat during digital transmission
JP6	ANALOG EN	Analog handshaking	1x2 .1” pin header	Connects INHIBIT_R to GPO_2 - used with wildcard inhibit modem during analog transmission
JP7	LED EN	Enable LED Display	1x2 .1” pin header	Enables front panel LEDs
JP8	SERIAL HOST	Testpoints/ Programming	1x5 .1” pin header	Exposes connection between the modem (STM32F446RET6) and HOST (NanoPi NEO)
TP1	RX buffer feedback	Testpoint	Solderpad	RX Buffer feedback loop test point
TP2	RX buffer output	Testpoint	Solderpad	Output of RX audio buffer used for calibration/testing
TP3	RXA	Testpoint	Solderpad	This is RX audio as presented to the modem ADC
TP4	TXA	Testpoint		This is the TX audio output from the modem DAC
TP5	TX buffer output	Testpoint	Solderpad	DAC output after buffer amplifier
TP6	TXA_R	Testpoint	Solderpad	TX audio output to exciter
TP7	RSSI	Testpoint	Solderpad	RSSI from receiver, as supplied to the modem
3.3V	3.3VDC	Testpoint	Solerpad	3.3V supply rail
5.0V	5.0V	Testpoint	Solderpad	5.0V supply rail
GND	GND	Testpoint	Solderpad	Ground bus

V1c-a Schematic Diagram



V1c-a PBC Layout Diagram



V1c-a Parts List

NOTE: Parts not listed are not populated on this version. Some unnecessary parts were removed, in some cases jumpers added, prior to this build version. Parts not listed and not populated include: R7, R9, R10, R13, R17, C14, C19. Parts jumpered include: R16, R31, C16

STM32_DVM_MTR2K v1c-a Build b

Value	Device	Package	Parts
ANALOG EN	.1” Pin Header 1x2	1X02	JP5, JP6
BOOTSTRAP	.1” Pin Header 1X2	1X02	JP2
RESET	.1” Pin Header 1X2	1X02	JP3
Serial Header	.1” Pin Header 1X4	1X04	JP1
SERIAL HOST	.1” Pin Header 1X5	1X05	JP8
ICSP	.1” Pin Header 2X3	2X03	JP4
XC6206P332MR	3.3V Linear Regulator	XC6206_SOT-23	P1
MTR2K J2	96 Pin (3x32) Eurocard	DIN41612	J1
BLU	Blue LED	CHIPLED_0805	LED2
.01uF	Ceramic Capacitor	C0805	C20, C21, C22, C25
.1uF	Ceramic Capacitor	C0805	C1, C2, C5, C7, C8, C10, C30
10pF	Ceramic Capacitor	C0805	C12, C13
22uF	Ceramic Capacitor	C1210	C17, C28
4.7uF	Ceramic Capacitor	C0805	C3, C4, C6, C9, C11, C29, C23, C27
47uF	Ceramic Capacitor	C1210	C15, C18
680pF	Ceramic Capacitor	C0805	C24, C26
100K	Chip Resistor	R0805	R14
10K	Chip Resistor	R0805	R4, R5, R6, R8, R15, R27, R29
1K	Chip Resistor	R0805	R2, R3, R28, R30, R34, R37
22K	Chip Resistor	R0805	R11, R12
3.3K	Chip Resistor	R0805	R18, R19, R21
3.9K	Chip Resistor	R0805	R22, R24
39K	Chip Resistor	R0805	R20, R23
47	Chip Resistor	R0805	R25, R26
47K	Chip Resistor	R0805	R1, R33
560	Chip Resistor	R0805	R32, R35, R36, R38, R39, R40, R41, R42, R43
8MHz	Crystal	HC49/S	Q1

MP2307	DC-DC Converter	Module	A1
BAV70	Dual Diode Array	SOT23C	D1
MCP6002	Dual Op-Amp	SO08	IC1, IC2, IC3
GRN	Green LED	CHIPLED_0805	LED5
FBead	L-USL2012C	L2012C	L1
2111	MUN2111LT1-PNP Prebias	SC59-BEC	T1, Q2
2211	MUN2211T1-NPN Prebias	SC59-BEC	T2
2214	MUN2211T1-NPN Prebias	SC59-BEC	T3
NanoPi NEO	NANOPI NEO	NANOPI-NEO	U2
JUMPER	Not Installed	C1210, R0805	C16, R16, R31
ORG	Orange LED	CHIPLED_0805	LED3, LED4
TCXO 12MHz	OSC-OE-CFPS-9	5.2X3.4-4-PAD	OS1
LED EN	Pin Header 1X2	1X02	JP7
RED	Red LED	CHIPLED_0805	LED1, LED6
Solder Pad	Solder Pad	Solder Pad	3.3V, 5.0V, GND, TP1, TP2, TP3, TP4, TP5, TP6, TP7
STM32F446RET6	STM32F446RET6	QFP64N	U1
YEL	Yellow LED	CHIPLED_0805	LED7, LED8, LED9, LED10, LED11

V2.0a Jumpers and Connections

STM32-DVM-MTR2K Connections:

ID	Name	Purpose	Type	Description
J1	MTR2K J2	MTR2000	96 Pin (3x32) Eurocard	Connects STM32-DVM-MTR2K to the MTR2000
J2	AVR_SPI_ICSP	Programming	.1” Pin Header 2x3	In-circuit SPI programmer for the ATmega328P
J3	I/O SERIAL	Programming	.1” Pin Header 1x6	Serial communication/programming for the ATmega328P
JP1	LED ENABLE	Enable MODEM LEDs	.1” Pin Header 1x2	Enables MODEM status LEDs
JP10	RESET	Programming	.1” Pin Header 1x2	Used to reset the MODEM, used with boot loader mode
JP11*	AVR RESET	Programming	.1” Pin Header 1x2	Connects ATmega328P reset to the station reset line
JP2	EXT PWR +7-15V	Testing/ Programming	.1” Pin Header 1x2	External power for the STM32-DVM-MTR2K while not inserted into an MTR2000
JP3	STM32 ICSP	Programming	.1” Pin Header 2x3	In-circuit SPI programmer for the STM32F446RET6
JP4	MODEM DISP	Peripheral	.1” Pin Header 1x4	Serial “repeater” connection via the MODEM
JP5	MODEM SERIAL	Testing/ Programming	.1” Pin Header 1x5	Exposes serial (/dev/ttyS1) connection between the MODEM (STM32F446RET6) and HOST (NanoPi NEO)
JP6	SBC DISP	Peripheral	.1” Pin Header 1x4	Serial (/dev/ttyS2) connection from the Host (NanoPi NEO) for serial display, peripheral or communication with the ATmega328P I/O processor
JP7**	I/O SERIAL EN.	Peripheral/ Programming	.1” Pin Header 2x2	Connects NanoPi NEO (/dev/ttyS2) to the ATmega328P I/O processor serial port
JP8***	ANALOG HANDSHAKE ENABLE	Peripheral	.1” Pin Header 2x2	Connects PTT_R to GPI_3 and INHIBIT_R to GPO_2. Used with wildcard GPIO features for in-cabinet repeat knockdown and modem inhibit for analog+digital operation
JP9	BOOT	Programming	.1” Pin Header 1x2	Used to place the MODEM into boot loader mode
LSP1	3.3V DC	Testpoint	Solder Pad	3.3V supply rail
LSP2	5.0V DC	Testpoint	Solder Pad	5.0v supply rail
LSP3	GND	Testpoint	Solder Pad	Ground bus
TP1	RX Buffer Feedback	Testpoint	Solder Pad	RX Buffer feedback loop test point
TP2	RX Buffer Output	Testpoint	Solder Pad	Output of RX audio buffer used for calibration/ testing
TP3	RX to MODEM	Testpoint	Solder Pad	RX audio as presented to the MODEM ADC
TP5	TX Buffer Output	Testpoint	Solder Pad	DAC output after buffer amplifier
TP6	TX to Repeater	Testpoint	Solder Pad	TX audio output to the exciter
TP7	RSSI	Testpoint	Solder Pad	RSSI from receiver, as supplied to the MODEM ADC

MTR2000 J5 System Connector:

ID	Name	Purpose	Description
A8	ATmega328P SCL	Expansion	I2c serial bus clock signal (5.0v, pull-up installed)
A9	ATmega328P SDA	Expansion	I2c serial bus data signal (5.0v, pull-up installed)
A26	NanoPi NEO SCL	Expansion	I2c serial bus clock signal (3.3v, pull-up installed)
A29	NanoPi NEO SDA	Expansion	I2c serial bus data signal (3.3v, pull-up installed)
A5	GPI 3	GPIO	Transistor buffered general purpose input. Used by the system when optional JP8 installed for analog handshake
C5	GPI 4	GPIO	Transistor buffered general purpose input
A22	GPI 7	GPIO	Transistor buffered general purpose input
C12	GPI 10	GPIO	Transistor buffered general purpose input
A12	GPO 0	GPIO	Open collector general purpose output
A11	GPO 2	GPIO	Open collector general purpose output. Used by the system when optional JP8 installed for analog handshake
A1	GPO 8	GPIO	Open collector general purpose output
B2	GPO 13	GPIO	Open collector general purpose output

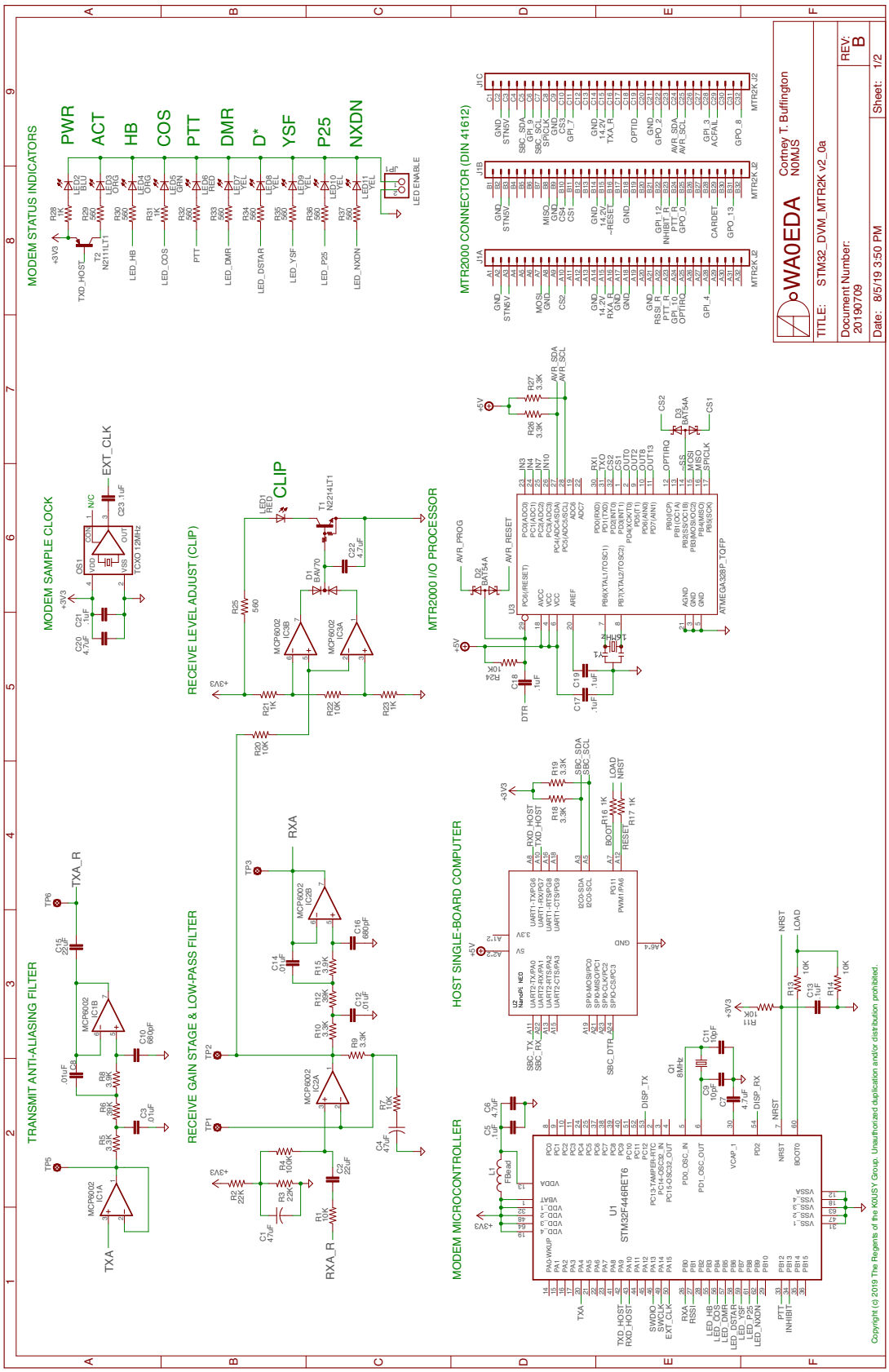
* JP11 AVR RESET: When closed, connects the hardware reset pin of the ATmega328P microcontroller to the Station Control Module external reset line. This is an output from the SCM that resets connected peripherals when the SCM itself resets. In order to ensure I/O synchronization, this jumper should not be removed. This line does not reset the ST32F446RET6 (MODEM) or the NanoPi NEO (Host).

** JP7 IO SERIAL EN.: Connects asynchronous port /dev/ttyS2 on the NanoPi NEO to the asynchronous port on the ATmega328P (through a 3.3v to 5.0v level converter). Jumpers must be removed to use NanoPi NEO /dev/ttyS2 with an MMDVM display (such as Nextion). Similarly, an MMDVM display must not be connected when using this port to update firmware on the ATmega328P, or for using the NanoPi NEO to read and/or set GPIO lines on the MTR2000 (future feature).

*** ANALOG HANDSHAKE ENABLE: When closed, connects PTT_R to GPI_3 and INHIBIT_R to GPO_2. Used with wildcard GPIO features for in-cabinet repeat knockdown and modem inhibit for analog+digital operation. GPI_3 and GPO_2 will not be available for general purpose use when connected.

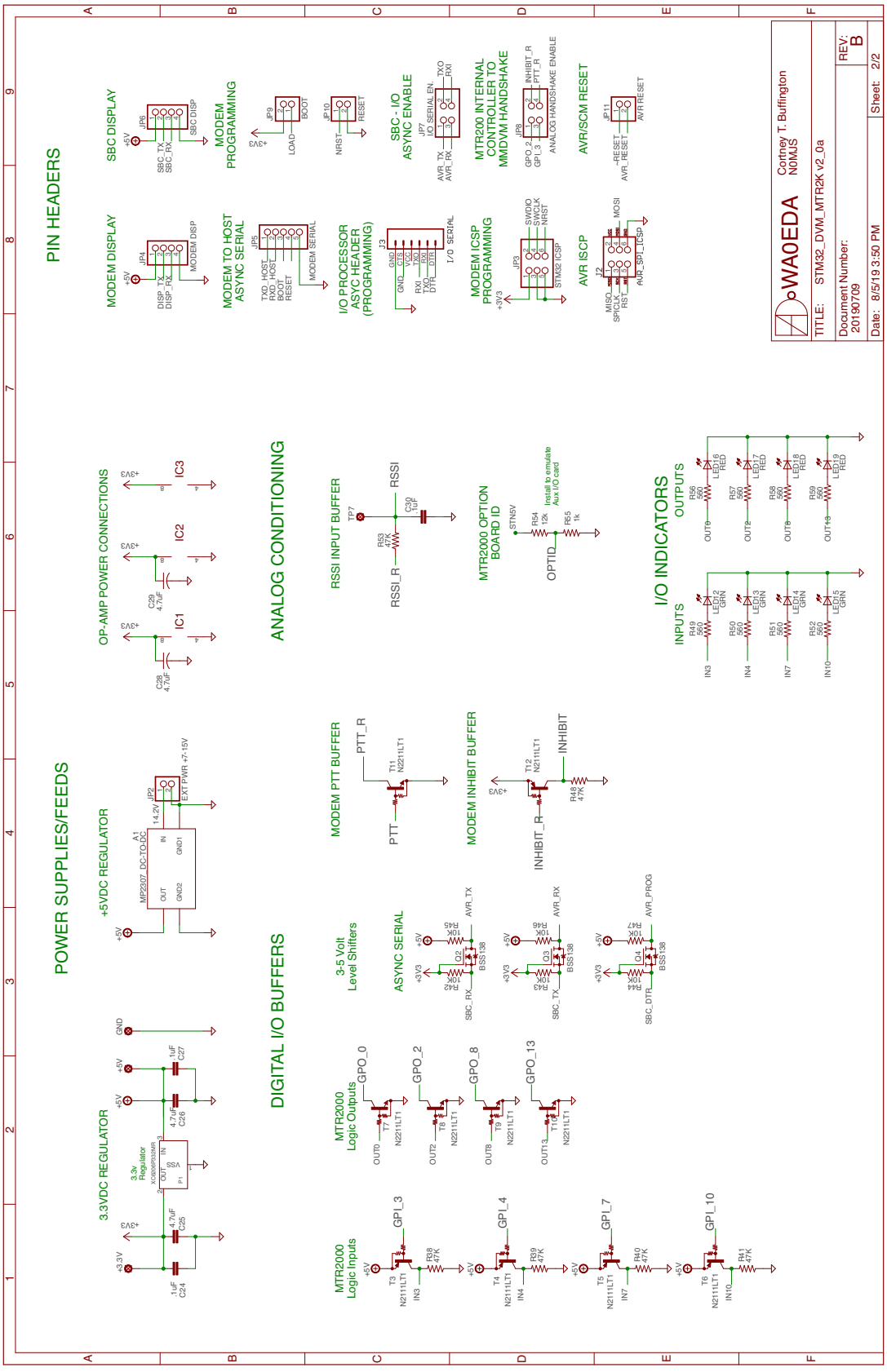
V2.0a Schematic Diagram

(page 1 of 2)

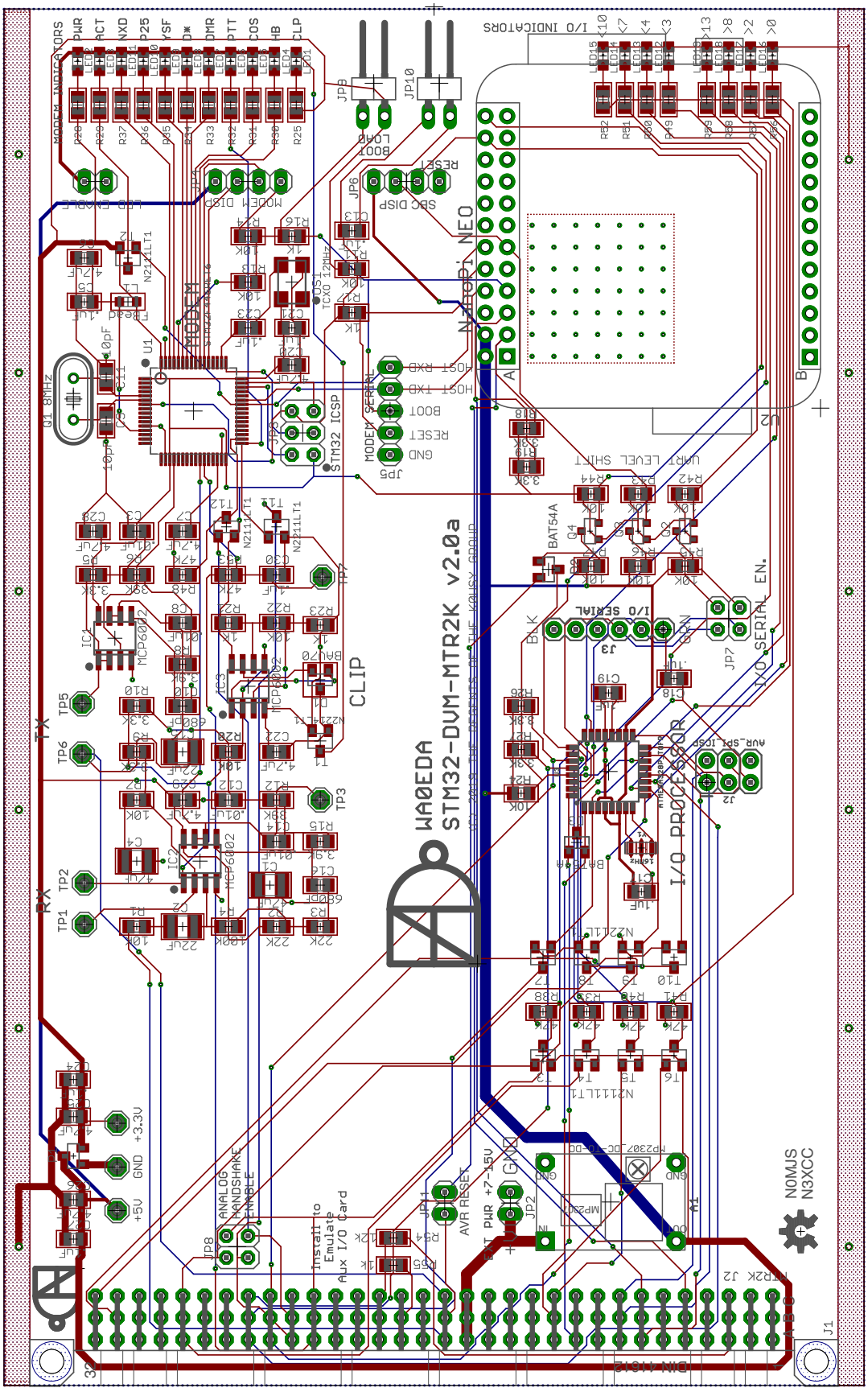


V2.0a Schematic Diagram

(page 2 of 2)



V2.0a PCB Layout Diagram



V2.0a Parts List

STM32_DVM_MTR2K v2_0a

Value	Device	Package	Parts
AVR RESET	.1” Pin Header 1x2	1X02	JP11
BOOT	.1” Pin Header 1x2	1X02/90	JP9
EXT PWR +7-15V	.1” Pin Header 1x2	1X02	JP2
LED ENABLE	.1” Pin Header 1x2	1X02	JP1
RESET	.1” Pin Header 1x2	1X02/90	JP10
MODEM DISP	.1” Pin Header 1x4	1X04	JP4
SBC DISP	.1” Pin Header 1x4	1X04	JP6
MODEM SERIAL	.1” Pin Header 1x5	1X05	JP5
I/O SERIAL	.1” Pin Header 1x6	2X06	J3
ANALOG HANDSHAKE ENABLE	.1” Pin Header 2x2	2X02	JP8
I/O SERIAL EN.	.1” Pin Header 2x2	2X02	JP7
STM32 ICSP	.1” Pin Header 2x3	2X03	JP3
AVR_SPI_ICSP	.1” Pin Header 2x3	2X03	J2
XC6206P332MR	3.3V Linear Regulator	XC6206_SOT-23	P1
MTR2K J2	96 Pin (3x32) Eurocard	DIN41612	J1
.01uF	Ceramic Capacitor	C0805	C3, C8, C12, C14
.1uF	Ceramic Capacitor	C0805	C5, C13, C17, C18, C19, C21, C23, C24, C27, C30
10pF	Ceramic Capacitor	C0805	C9, C11
22uF	Ceramic Capacitor	C1210	C2, C15
4.7uF	Ceramic Capacitor	C0805	C6, C7, C20, C22, C25, C26
4.7uF	Ceramic Capacitor	C0805	C28, C29
47uF	Ceramic Capacitor	C1210	C1, C4
680pF	Ceramic Capacitor	C0805	C10, C16
100K	Chip Resistor	R0805	R4
10K	Chip Resistor	R0805	R1, R7, R11, R13, R14, R20, R22, R24, R42, R43, R44, R45, R46, R47
12k	Chip Resistor	R0805	R54
1K	Chip Resistor	R0805	R16, R17, R21, R23, R28, R31
1k	Chip Resistor	R0805	R55
22K	Chip Resistor	R0805	R2, R3

3.3K	Chip Resistor	R0805	R5, R9, R10, R18, R19, R26, R27
3.9K	Chip Resistor	R0805	R8, R15
39K	Chip Resistor	R0805	R6, R12
47K	Chip Resistor	R0805	R38, R39, R40, R41, R48, R53
560	Chip Resistor	R0805	R25, R29, R30, R32, R33, R34, R35, R36, R37, R49, R50, R51, R52, R56, R57, R58, R59
8MHz	Crystal	HC49/S	Q1
MP2307	DC-DC Converter	Module	A1
BAV70	Dual Diode Array	SOT23C	D1
BAT54A	Dual Schottky Diode Array	SOT23	D2, D3
L-USL2012C	Ferrite Bead	L2012C	L1
BLU	LED Blue	CHIPLED_0805	LED2
GRN	LED Green	CHIPLED_0805	LED5, LED12, LED13, LED14, LED15
ORG	LED Orange	CHIPLED_0805	LED3, LED4
RED	LED Red	CHIPLED_0805	LED1, LED6, LED16, LED17, LED18, LED19
YEL	LED Yellow	CHIPLED_0805	LED7, LED8, LED9, LED10, LED11
ATMEGA328P	Microcontroller	TQFP32-08	U3
STM32F446RET6	Microcontroller	QFP64N	U1
N2111LT1	MUN2111LT1-PNP Prebias	SOT23-BEC	T3, T4, T5, T6, T12
N2211LT1	MUN2211T1-NPN Prebias	SC59-BEC	T2, T7, T8, T9, T10, T11
N2214LT1	MUN2211T1-NPN Prebias	SC59-BEC	T1
BSS138	N-Channel MOSFET	SOT23	Q2, Q3, Q4
MCP6002	Operational Amplifier	SO08	IC1, IC2, IC3
TCXO 12MHz	OSC-OE-CFPS-9	5.2X3.4-4-PAD	OS1
16MHz	RESONATOR-16MHZ	SMD-3.2X1.3	Y1
NanoPi NEO	Single-Board Computer	Module	U2
Solder Pad	Solder Pad	MCS10B	LSP1, LSP2, LSP3, TP1, TP2, TP3, TP5, TP6, TP7