Ground Loops and other Buzz

Miracles you can afford.

« Back to blog

JUL 3

MaxTroller: Arduino Control for MaxTrac radios

Motorola MaxTrac 800 MHz radios are easy to find in surplus, and they have excellent receivers and audio.

Unfortunately, tuning and reprogramming them is somewhat difficult. The Motorola "RSS" software to program MaxTrac radios is restricted in distribution and limited by the on-board firmware in the radio.

By replacing the Tuning CPU with an Arduino, the RF and Audio sections of the radio can be re-used to build ham radio receivers of your own design.

This project starts with a two-trunk (only) MaxTrac 820, and puts the radio tuning and audio muting under control of the Arudino instead of the MaxTrac 68HC11 microcontroller.

The pictures below are for logic board **HLN9313B**. If you have a different rev, or a 5-pin board, please inspect your own board drawings and schematics to make sure they are a match.

Proceed only with Caution!

If you chose to build a new radio out of parts and software you found on the internet, you must take ALL necessary measures to ensure you do not ever make

unauthorized transmissions or receive prohibitted bands. It is YOUR RESPONSIBILITY as a radio builder to secure any necessary licenses, certifications and permission needed to operate a homebrew radio.

Required safety measures may include programming your radio firmware to inhibit ALL transmission, setting tx deviation to zero, removing exciter sections, and powering your radio with a low-current (500mA?) supply or fuse.

This tuning modification, in itself, does not inhibit the transmit path.

Your radio, your ham license, YOUR responsibility.

Pick the right Arduino

Any size Arudino flash will do fine because the code is small. The code is small because it doesn't do all that much.

Make sure you have a 5-volt Arduino, as you have to drive five-volt signals. ATMega168, ATMega328.. either will work fine.

Keep Frequency Calibration

Unless your radio is a mix & match parts build, or "blanked" kit, it probably has accurate crystal calibration values in it. Since we tune only the PLL, it's important that the Reference Oscillator be correctly calibrated before we begin tuning. For this reason, we leave the MaxTrac CPU intact and connected to the DAC that tunes the reference oscillator.

If you believe your MaxTrac is totally uncalibrated or incorrectly calibrated, deal with that first. The **MaxTroller** uses the same reference oscillator, so it does need to be accurate.

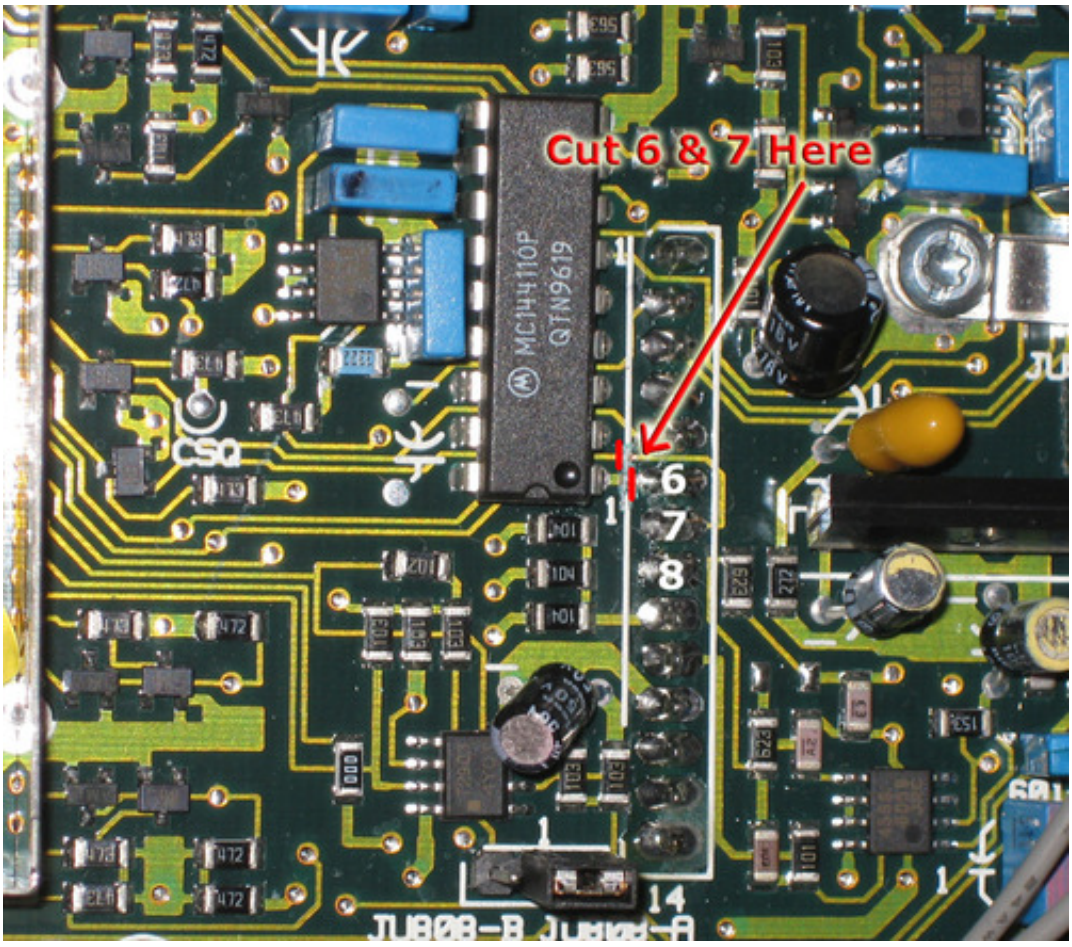
Discard PLL tuning

The first step is to cut the traces between the controlling CPU and pins 6 & 7 of the RF board. This ensures that the only serial clock/data received by the RF board is from our Arduino, while the CPU talks only to the DAC.

If you are noncommital or curious, there is a fair chance the Arduino can override the CPU and work just fine without cutting these traces. Some PLL programming may get garbled during a collision, especially on Trunk-only radios which retune often. Try it and see.

These traces are on the top surface -- really easy to cut. Count down to pins 6 and 7 and find a nice place to work your XActo knife.

The J6-8 (Synth LE) pin doesn't need to be disconnected/cut, which is good because it's hard to reach. We can just haggle with the CPU over it, and win every time. If you find an easy way to disconnect it from the RF board, it wouldn't hurt to do so. And post.

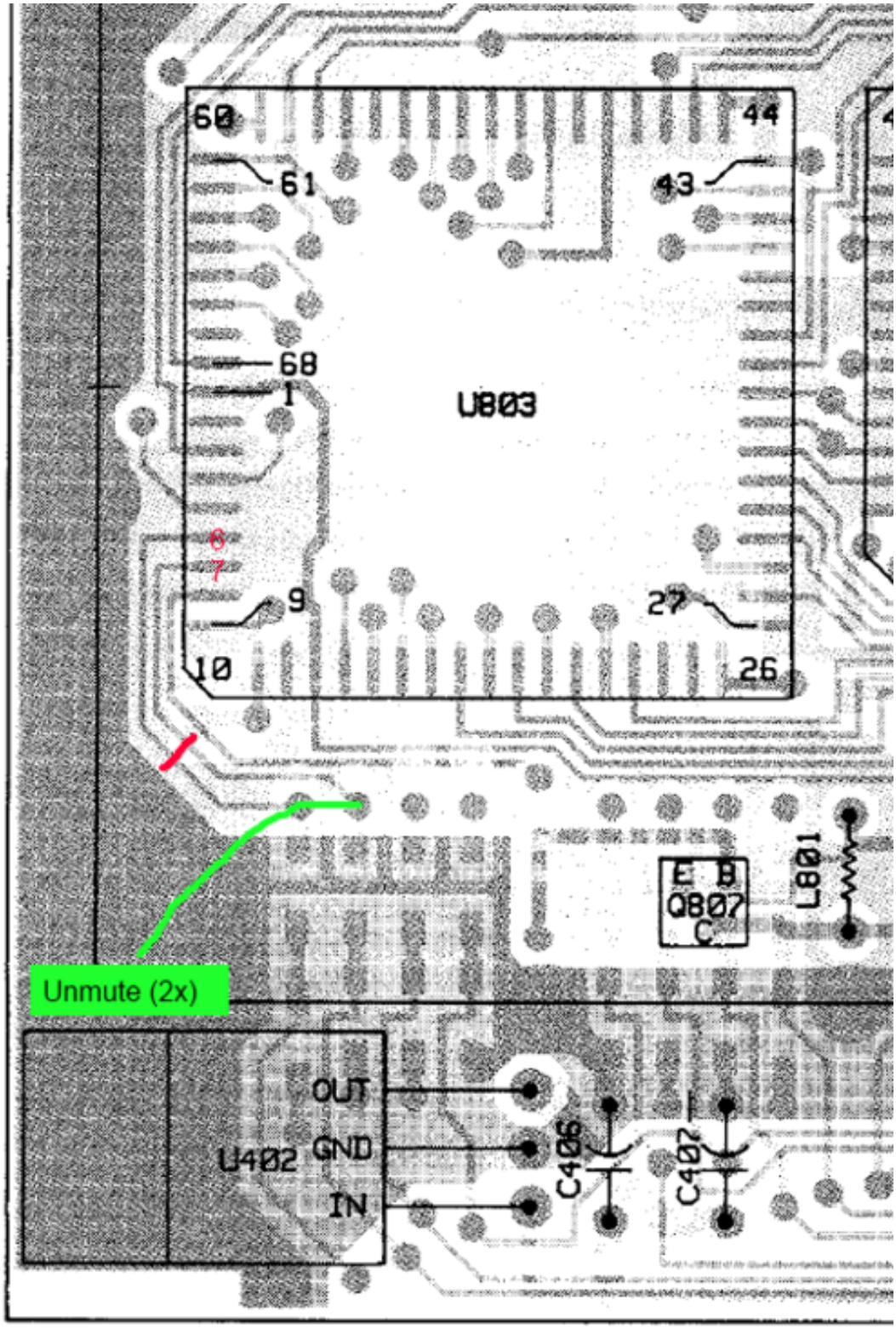


Download full size (368 KB)

Mute Audio

If you have a Conventional codeplug in the radio, you can park it on a conventional channel and rely on the CPU to unmute the audio when Carrier Squelch receives signal. Good for you.

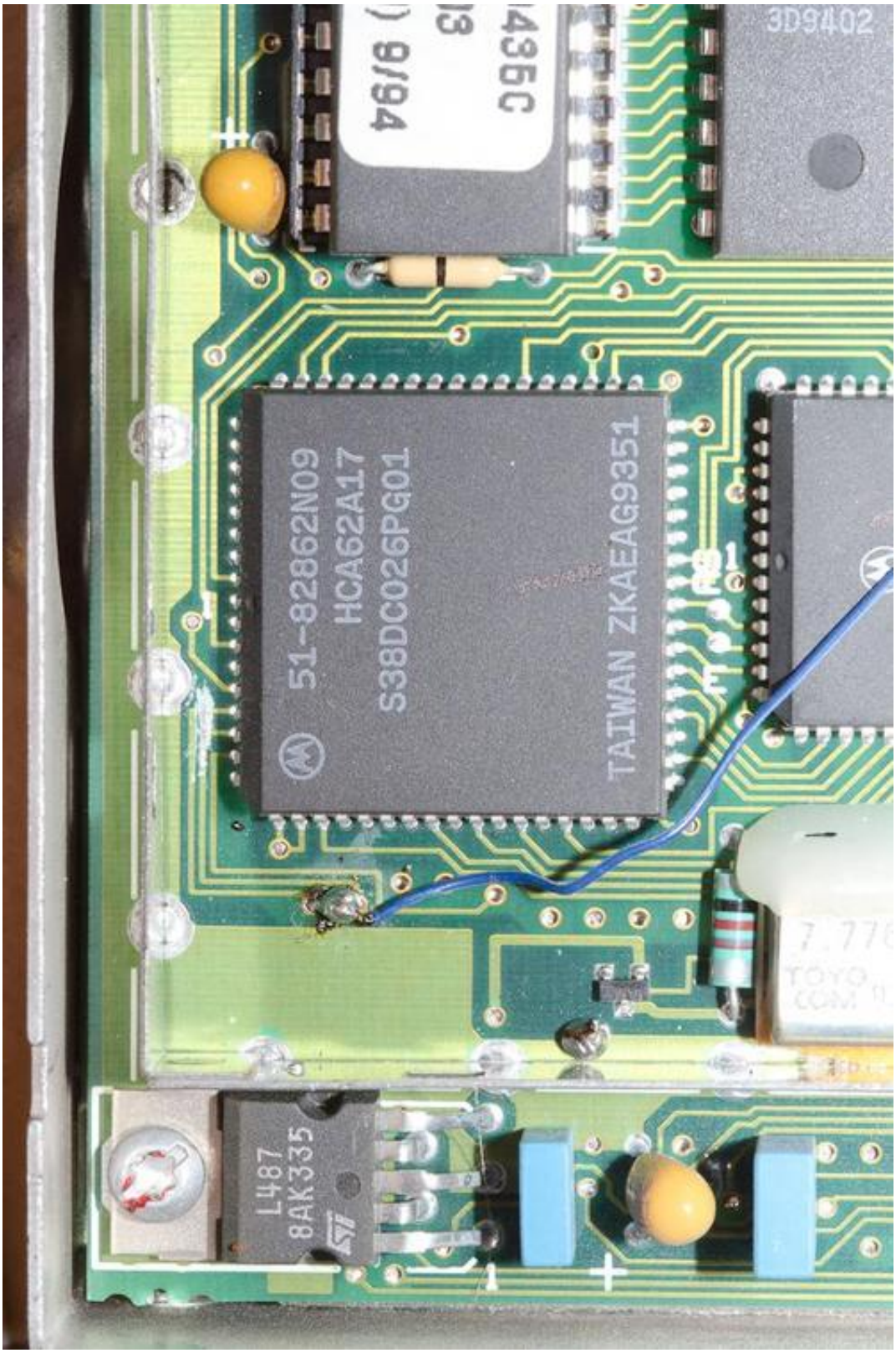
However, if you have trunking-only firmware in your radio, then you must put the Arduino in charge of RX MUTE and PA MUTE as well, or you will never hear the audio on the internal amplifier. Maybe that's okay with you if you use the non-muted (always on) discriminator/flat audio tap found on Pins 7 & 11 of the 16-pin back connector. Otherwise, you have more trace cutting to do.



Download full size (127 KB)

A trunk-only radio will never 'unmute' itself, regardless of carrier squelch. So, you must cut the CPU signals from RX MUTE and PA MUTE and drive those from the Arduino. This is Pin 6 (RX MUTE) and Pin 7 (PA MUTE) on the CPU803, which is found in the Computing Can part of the logic board. These traces are easy to access and cut. Then you must tap the same lines (maybe at the nearby vias), connect them together, and bring those to Digital 6 of the

Arduino.



Download full size (141 KB)

If this sounds too hard, maybe replacing your EPROM or whole radio is an easier way to get at least one Conventional channel?

There is a jumper option that lets you get muted audio on the back accessory connector as well, with no PA involved, but you still
have to reroute RX MUTE to let the Arduino control that gate for you.

Don't forget that the back-panel 16-pin accessory plug is needed to route the audio to the internal speaker. If yours is missing, your radio will remain silent until you connect the right pins (15 & 16) together.

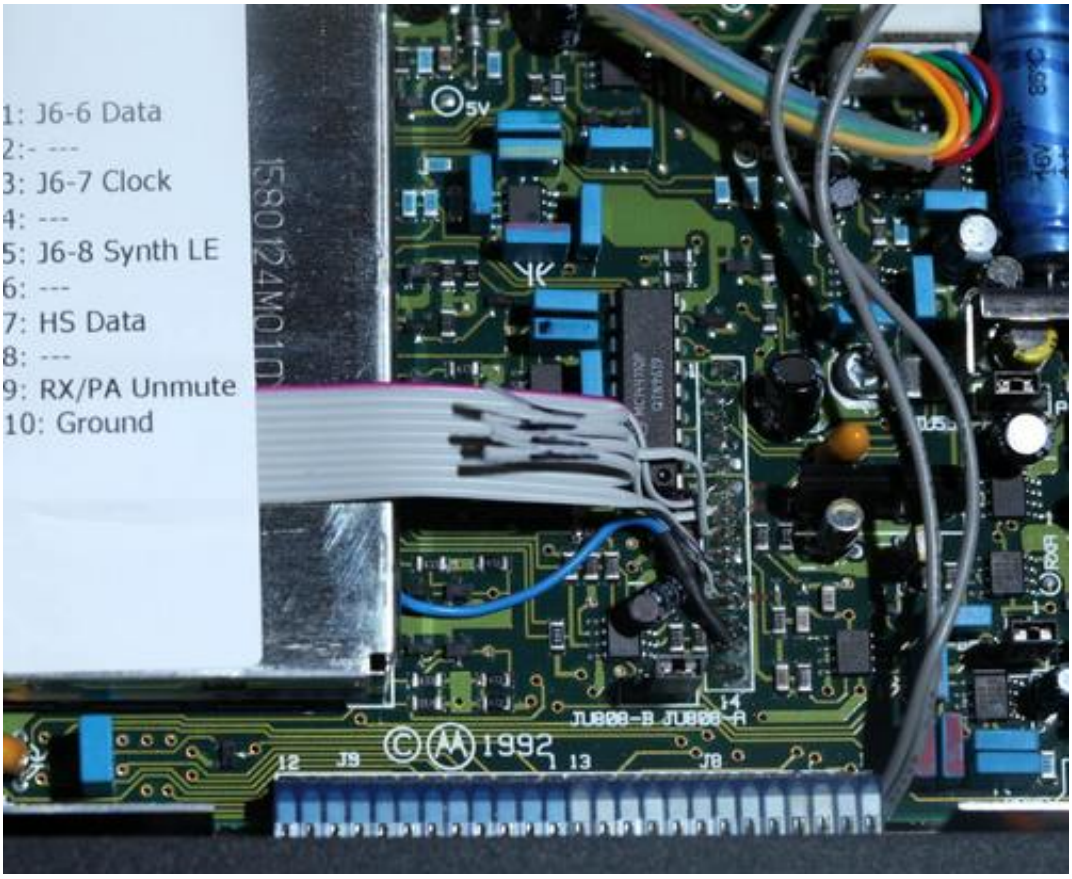
"High Speed" Data Tap

The filtered discriminator audio feeds an op-amp Comparator in the MaxTrac. This provides an excellent 5v data signal for the Arudino to sample as a data bitstream and avoids any audio-card or A/D analog processing later. To take advantage of the on-board discriminator, you must tap a signal that doesn't come to J6. U601 pin 1 (corner pin facing J6 and the front panel) has the signal, but it's easiest to pick up at the pull-up resistor R605, on the side facing the CPU. It's less cluttered here, and a bigger solder pad to get on.

Download full size (69 KB)

Note that your J601 jumper selects the audio feed for this comparator's input. I've been using the "B" position, which taps before the 300Hz High-Pass filter (raw as possible). I'm not sure it matters at all for the 3600bps control channel decoding, but you might try experimenting to see which gives the lowest packet error rate. The comparator output is always active, immune to mute.

Connect Arduino



Download full size (159 KB)

All set? Connect the Arduino Digital I/O pins as follows:

- Pin 2 to J6-6 Serial Data
- Pin 3 to J6-7 Serial Clock
- Pin 4 to J6-8 Synth LE
- Pin 5 to U601-1 ("High Speed" Data)
- Pin 6 to the cut U803 Pin6 & Pin7 (RX UNMUTE & PA UNMUTE)
- Pin 7 to J9-2 (Display Enable) OPTIONAL. Five-button front panels only.
- Arduino Ground (any) to MaxTrac Ground, available at J6-11

If you intend to use the Accessory port Pin 11 discriminator audio output, make sure JU551 is in the "A" position for no emphasis or muting. If you intend to connect speakers for audio on Pin 11, you want the opposite ("B") to take advantage of Arduino muting.

- Comment

About

Ground Loop's profile »