COYOTE WAVE - CW Morse Code Generator

by: KG5JMV

Document Laste Updated: 8-30-2026

OPERATION INSTRUCTIONS

To run this safely, you must use a browser that supports the Web Serial API (such as Google Chrome, Microsoft Edge, or Opera).

Features:

- CW Code Audio Output - CW audio is produced when clicking |Start Beacon| the sound is sent to your devices default audio port.

- CW Code RTS - DTR Serial Output - CW signal is produced through the devices RS-232 serial port.

How to use the Manual Control Feature:

1. Load this file and connect your USB-to-serial device.

2. The **KEY DOWN (Transmit Carrier Continuous)** button will become active.

3. Clicking it turns the transmitter line **ON continuously**. This lets you adjust an antenna tuner, measure SWR, or check your transmission

path without sending a message.

4. Clicking it again releases the carrier safely.

In serial communications, RTS (Request to Send) and DTR (Data Terminal Ready) are hardware control pins found on standard RS-232 and USB-to-Serial chips (like FTDI, CP2102, or CH340).

In their original, historical design, they were used for "hardware flow control"—allowing a computer and a modem to signal to each other when they were ready to send or receive data.

However, in amateur radio and custom hardware hacking, we repurpose these pins as simple, programmable electronic switches to control external equipment.

1. RTS (Request to Send)

Original Purpose: The computer sets this pin high to tell an attached modem, "I have data ready to send to you; clear the line."

Radio Keying Purpose: In amateur radio applications, RTS is the most common standard used for PTT (Push-To-Talk) or CW Keying.

Behavior: When the Web Serial API executes setSignals({ rts: true }), the physical RTS pin on your USB adapter instantly changes from a low voltage state to a high voltage state (typically +3.3V or +5V). This feeds current into your transistor, triggering the transmitter carrier

wave.

2. DTR (Data Terminal Ready)

Original Purpose: The computer sets this pin high to tell the modem, "I am powered on, active, and ready to communicate."

Radio Keying Purpose: DTR acts exactly like RTS, but gives you a second, independent control switch on the same USB plug.

Behavior: When the Web Serial API executes setSignals({ dtr: true }), the physical DTR pin switches to high voltage, completely independent of whatever the RTS pin is doing.

Browser Sandbox Check: If you click the connection button and nothing happens, double-check that you are using a Chromium-based browser like Chrome or Edge. Firefox and Safari block Web Serial access entirely due to security policies.

Virtual Port Testing: If you want to verify that the code is working before building the transistor circuit, you can plug in a standard USB-to-TTL adapter (like an FTDI or CP2102 module) and hook a simple LED and a resistor between the RTS (or DTR) pin and the GND pin. The LED will flash perfectly in sync with the Morse dots and dashes.

RF Interference Prevention: Once you plug the line into a real radio transmitter, the RF energy from your antenna can sometimes travel

back down the key line and temporarily glitch out a computer's USB port. If your USB connection drops out right when the transmitter fires up, snapping a small ferrite choke onto your USB cable or keying line usually clears it right up.

Why do we offer a choice between both?

Offering both options in your application is highly useful for two reasons:

Dual-Control Interfaces: Many commercial ham radio USB interfaces use both pins simultaneously over a single USB cable. For example, a common layout uses DTR to key the Morse code (CW) and RTS to control the radio's Push-To-Talk (PTT) voice line.

Wiring Flexibility: Depending on how a user solders their cable, or what type of commercial adapter they plug in, their hardware switch might be hardwired to RTS or DTR. The drop-down selector ensures your app works with any custom-made cable setup without requiring a soldering iron change.

CW Key Jack Voltage LevelsThe CW key jack operates on standard TTL-level logic (0 to 5 V DC), utilizing an internal pull-up

resistor.

Key Open (Idle): The voltage across the open key contacts is approximately +5 V DC.

Key Closed (Transmit): The voltage drops to 0 V (grounded) to trigger the transmitter.

Current Draw: The circuit draws a very low current, typically around 0.5 mA to 2 mA.

Warning: Never connect an old tube-amplifier keying line or high-voltage keying system directly to this jack. It cannot handle vintage grid-block keying voltages (which can exceed -100 V DC) or high AC voltages. Doing so will permanently damage the radio's internal logic circuitry.

does not use RS-232 voltage levels directly on the radio. Instead, it uses the transceivers proprietary CI-V (Communication Interface-V) protocol via the 3.5mm "REMOTE" jack.

CI-V Rig Voltage: This port uses TTL logic levels (0 V to 5 V DC) over a single-wire, bi-directional open-collector bus.

Standard RS-232 Voltages: True computer RS-232 serial ports use bipolar voltages between +3 V to +15 V (for Logic 0) and -3 V to -15 V (for Logic 1).

Because of this mismatch, you cannot plug a standard RS-232 computer port directly into the radios REMOTE jack without destroying the interface circuitry.

To bridge them safely, you must use a level converter (such as an CT-17 or a third-party USB-to-CI-V adapter). This converter translates the computer's high-voltage RS-232 signals down to the radio's safe 0–5 V TTL level signals.

To safely build your own interface for both CW keying and computer control (CI-V) on the transceiver, you need to convert the computer's high-voltage RS-232 signals down to safe TTL levels.
Here are the standard schematics and components you can use to build a robust, DIY interface.
1. The CW Keying Circuit (RS-232 to Radio Key Jack)
Since the radios key jack expects a simple switch-to-ground at 5 V, you must isolate the computer from the radio. The safest way to do this is using an optocoupler or a standard NPN transistor.
Computers typically use the DTR (Data Terminal Ready) or RTS (Request to Send) pins on the RS-232 port to trigger CW keying.

* Using an Optocoupler (Highly Recommended):
* Component: 4N25, 4N33, or PC817 optocoupler.

* How it works: The computer's RS-232 pin drives an internal LED inside the chip. This optically triggers a phototransistor, which shorts the radio's key line to ground without any electrical connection between the two devices. This completely prevents ground loops and RF feedback.
* Wiring: Connect the RS-232 DTR (or RTS) pin to a 1k-ohm resistor, then to the optocoupler's anode (Pin 1). Connect the RS-232 Ground to the cathode (Pin 2). Connect the radio's CW Key Tip to the optocoupler's collector and the radio's ground to the emitter.

* Using an NPN Transistor:

* Component: 2N2222 or 2N3904.
* Wiring: Place a 4.7k-ohm resistor between the RS-232 DTR/RTS pin and the transistor's Base. Connect the Emitter to shared ground. Connect the Collector to the radio's CW key tip. Add a standard diode (like a 1N4148) backward across the emitter-base junction to protect the transistor from the negative voltage (-12V) that RS-232 outputs when the key is open.

2. The Rig Control Circuit (RS-232 to CI-V Remote Jack)

The Radios REMOTE jack is a single 3.5mm mono plug where Transmit (TX) and Receive (RX) share a single wire. Your homebrew circuit needs to combine the computer’s separate RS-232 TX and RX lines into this one wire while shifting the voltages.

* The Max232 Chip Method (Most Reliable):

* Use a MAX232 or MAX3232 integrated circuit.
* This chip automatically converts RS-232 (+/- 12V) to TTL (0-5V).
* You connect the computer TX/RX to the RS-232 side of the chip. On

the TTL side, you tie the TTL TX and TTL RX lines together using a standard steering diode (1N4148) so they can share the single wire going to the
3.5mm tip.
* The Simple Transistor Method (Minimalist):
* You can build a cheaper, two-transistor circuit using common BC547 or 2N3904 transistors and a few resistors/diodes to handle the level shifting and line combining. (Many schematics for "DIY CI-V interface" online use this exact layout).

Quick Pin Reference for Your Build

* PC DB9 Serial Port Pins:
* Pin 3: TX (Transmit Data)
* Pin 2: RX (Receive Data)
* Pin 4: DTR (Often used for CW Keying)
* Pin 7: RTS (Alternative for CW Keying / PTT)
* Pin 5: GND (Signal Ground)
* Radio Connections:
* CW Jack (1/4" or 3.5mm): Tip = Key Line (+5V open), Sleeve = Ground.
* REMOTE Jack (3.5mm mono): Tip = CI-V Data Line, Sleeve = Ground.

Here is the exact pin-to-pin wiring guide to build an optoisolated CW

keying circuit using a DB9 serial port and a standard 4N25 (or 4N35 / PC817) optocoupler.
This design protects your radio from ground loops and accidental high voltage by ensuring the computer and radio are completely isolated electrically.

Required Components

* Optocoupler: 4N25, 4N35, or PC817
* Resistor: 1k-ohm (1/4 watt)
* Diode: 1N4148 or 1N4001 (Protects the optocoupler LED from negative RS-232 voltage)
* Connectors: Female DB9 connector, 1/4" (or 3.5mm) mono phone plug (depending on your chosen radio key jack).

Step-by-Step Pin Mapping
Most logging software (like N1MM, Ham Radio Deluxe, or DXKeeper) lets you choose either DTR or RTS for CW keying. Pick one of the configurations below based on what you plan to select in your software.
Option A: Using the DTR Pin for Keying (Most Common)

From: Computer (DB9 Female)	Connection / Component	To: Optocoupler Pin
Pin 4 (DTR)	Connect to one side of the 1k-ohm Resistor. Connect the other side of the resistor to...	Pin 1 (Anode / +)

| Pin 5 (GND) | Connect directly to... | Pin 2 (Cathode / -) |

Option B: Using the RTS Pin for Keying

From: Computer (DB9 Female)	Connection / Component	To: Optocoupler Pin
Pin 7 (RTS)	Connect to one side of the 1k-ohm Resistor. Connect the other side of the resistor to...	Pin 1 (Anode / +)
Pin 5 (GND)	Connect directly to...	Pin 2 (Cathode / -)

Crucial Component: The Protection Diode

Because a standard RS-232 serial port swings between roughly +12V (Key Closed) and -12V (Key Open), the negative voltage can destroy the delicate LED inside the optocoupler.

* Solder a 1N4148 diode directly across Pin 1 and Pin 2 of the optocoupler.
* The stripe (cathode) of the diode must go to Pin 1.
* The non-stripe side (anode) of the diode must go to Pin 2.
* Why: When the serial port drops to -12V, this diode safely conducts the reverse current away from the internal LED.

Connecting to the Radios Key Jack

Now, wire the output side of the optocoupler to your radio’s CW key

plug. Do not connect the computer ground (DB9 Pin 5) to the radio ground.

From: Optocoupler Pin	Connection	To: Radio Key Plug
Pin 4 (Emitter)	Connect directly to...	Sleeve (Ground / Outer shield)
Pin 5 (Collector)	Connect directly to...	Tip (Key Line)
Pin 6 (Base)	Leave disconnected	Do not connect

(Note: If you are using a 4-pin optocoupler like the PC817 instead of a 6-pin 4N25, the Emitter is Pin 3 and the Collector is Pin 4).

Testing Your Build

1. Plug the interface into your computer, but do not plug it into the radio yet.
2. Open your ham radio software and set the CW keying to the correct port and pin (DTR or RTS).
3. Set your digital multimeter to Continuity/Diode test mode and place the probes on the radio plug's Tip and Sleeve.
4. Send a test "CQ" or string of dots via the software. Your multimeter should beep or show near 0 ohms in time with the CW code.
5. If it works, it is safe to plug into your transceiver.

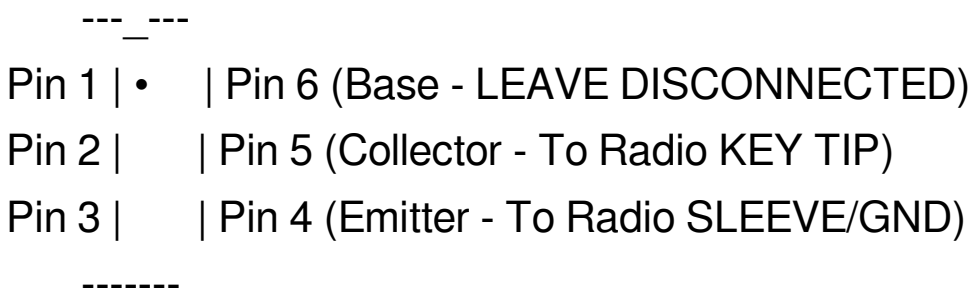
Since the CW keying circuit is entirely self-contained, you can house it in a tiny project box or even solder the few components directly inside the shell of the DB9 connector if you use a standard 1/4-watt resistor and a

small optocoupler.
Before you button everything up, make sure to configure your logging or keying software with the following settings to match your new hardware:

* Port: Select the specific COM port number assigned to your DB9 serial interface.
* CW/Key Line: Set this explicitly to DTR (or RTS, depending on which pin you soldered to Pin 1).
* PTT Line: Set this to None or Unassigned (unless you are using the other pin for a push-to-talk circuit, though the transceiver can handle PTT automatically via its internal VOX/Break-in settings when it senses the CW key closure).
* Handshaking/Flow Control: Set to None. This prevents the computer from randomly toggling the DTR/RTS lines to manage data flow, which would cause the radio to transmit unexpectedly.

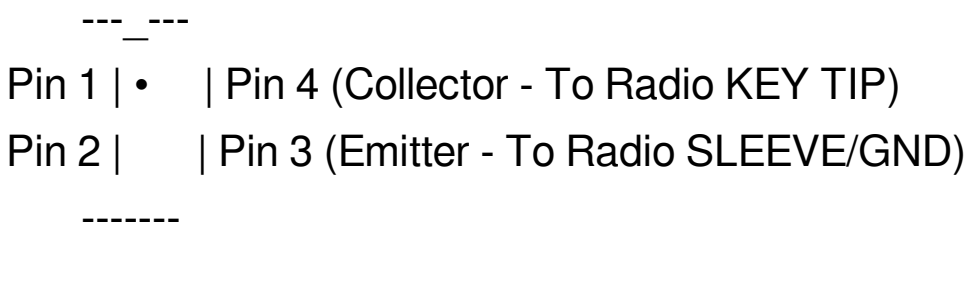
To identify the pinout of your optocoupler, you need to look at the top of the chip for a small circular dimple, a notch, or a printed stripe at one end. This marker always indicates Pin 1.
Hold the chip so the text reads right-side up, with the Pin 1 marker on your left. Pin 1 will be the bottom-left corner. From there, the pin numbers count counterclockwise around the chip.
Here is the exact pinout for the two most common types used for this project:

1. Six-Pin Optocoupler (4N25, 4N35, 4N26)
This is the most common style for DIY serial interfaces. It has three pins on each side.



* Pin 1 (Anode): Connects to the 1k-ohm Resistor (from DB9 Pin 4 or 7).
* Pin 2 (Cathode): Connects to DB9 Pin 5 (GND).
* Pin 3 (NC): No internal connection. Leave disconnected.
* Pin 4 (Emitter): Connects to the radio plug Sleeve (Ground).
* Pin 5 (Collector): Connects to the radio plug Tip (Key Line).
* Pin 6 (Base): Leave disconnected for this circuit.

2. Four-Pin Optocoupler (PC817, EL817)
If you are using a smaller, space-saving four-pin chip, it has two pins on each side.



* Pin 1 (Anode): Connects to the 1k-ohm Resistor (from DB9 Pin 4 or 7).
* Pin 2 (Cathode): Connects to DB9 Pin 5 (GND).
* Pin 3 (Emitter): Connects to the radio plug Sleeve (Ground).
* Pin 4 (Collector): Connects to the radio plug Tip (Key Line).

Solder Tip:
When soldering the protection diode (1N4148) across Pins 1 and 2, remember that the striped side of the diode goes to Pin 1, and the non-striped side goes to Pin 2.

Recommendation: Go with the PC817 (4-Pin)
If you are buying a chip specifically for this, look for the PC817 (or EL817).

* Why it's best: It only has 4 pins instead of 6. This makes it much smaller and easier to solder directly inside the plastic shell of your DB9 connector, saving you from needing a separate project box.
* Where to find it: They are incredibly common, costing only a few cents each on Amazon, eBay, or at any electronics supply store.

Your Final Shopping / Parts List
When you go to order or pick up your parts, here is exactly what you need to grab:

1. Optocoupler: PC817 (Sharp) or EL817 (Everlight) — quantity: 1
2. Resistor: 1k-ohm (1/4-watt or 1/8-watt axial resistor) — quantity: 1
3. Diode: 1N4148 (switching diode) or 1N4001 (rectifier diode) — quantity: 1
4. Serial Connector: DB9 Female connector with a plastic housing/hood — quantity: 1
5. Radio Plug: 1/4" (6.35mm) mono phone plug or 3.5mm mono phone plug (Check the "KEY" jack on the back of your specific radio — it is usually a 1/4" jack, while the front panel is sometimes different).

Solder Preparation Cheat-Sheet
When you are ready to build, here is the most compact soldering sequence to fit everything neatly inside that DB9 hood:

1. Prep the Diode & Resistor: Cut the legs of the 1N4148 diode short. Solder it directly across Pin 1 (stripe side) and Pin 2 (no stripe side) of your PC817 chip.
2. Attach the DB9 Ground: Solder a short jumper wire from DB9 Pin 5 directly to PC817 Pin 2.
3. Attach the Key Pin: Solder one leg of your 1k-ohm resistor directly to DB9 Pin 4 (or Pin 7). Solder the other leg of that resistor directly to PC817 Pin 1.

4. Connect the Radio Cable: Take a 2-conductor shielded audio cable. Solder the center wire to PC817 Pin 4 (Collector) and the outer shield wire to PC817 Pin 3 (Emitter).

5. Finish the Cable: Solder the other end of that audio cable to your 1/4" mono plug (Center wire to the Tip, shield wire to the Sleeve).

Fix FireFox Browser so it will Allow Web to Serial Port Access

Because Firefox does not natively support the Web Serial API due to security philosophies, this extension acts as a "polyfill" (a translator).

* The Extension Layer: It tricks your web browser into thinking the Web Serial code (navigator.serial) is fully supported.
* The Native Bridge Application: To actually talk to your computer's physical USB port, the extension requires you to install a small "native application" directly onto your Windows, Mac, or Linux operating system.

What to Expect When Setting It Up

1. Install the extension from the Mozilla Add-ons page.
2. Open your Morse Beacon HTML page in Firefox.
3. Click your "Connect USB Serial Port" button.

4. The extension will automatically open a pop-up window/GUI offering a link to download and install the required companion desktop app for your computer.

Once that companion app is installed, your beacon's hardware keying script will run seamlessly inside Firefox just like it does in Chrome and Edge.
If you run into any permission blocks or device connection errors while testing it in Firefox, let me know and we can troubleshoot it!

Extention:
https://addons.mozilla.org/en-US/firefox/addon/webserial-for-firefox/