JL Wrangler CAN Bus Reverse Engineering

Complete Technical Reference — Full Thread Summary (All 52 Pages)

JLWranglerForums.com — Original Thread (52 pages, 140 watchers)

Thread started Nov 7, 2021 by jmccorm (Josh, Tulsa OK). Primary hardware: Raspberry Pi 3B + Waveshare 2-CH CAN HAT. Last major OP update: March 2023.

1. Project Overview & Key Resources

This project reverse-engineers the 2018+ JL Wrangler (and JT Gladiator) CAN-C and CAN-IHS bus networks. The CAN bus name used by Chrysler/FCA for this system is 'PowerNet' — the combination of the CAN-C and CAN-IHS buses. Started by jmccorm with earlier groundwork by redracer, it has grown into a community effort to decode message IDs, build automation scripts, and enable new software-based vehicle modifications.

1.1 Primary Resources

- Google Sheets spreadsheet (master decoded message list): https://docs.google.com/spreadsheets/d/16ypMADKinBBnH1pOY4-gMmVRjeR85fYplpV12aCHJC4
- GitHub (Raspberry Pi scripts): https://github.com/rstellhorn/jeep-jl-powernet-scripts — maintained by redracer
- DBC files for SavvyCAN: https://github.com/darksotmoon/jlcan — load into SavvyCAN to auto-decode packets
- Raw 4-mile drive CAN log (47MB candump + 104KB summary): attached to Post #37 as 4mi-loop.zip
- Virtual CAN / canplayer walkthrough for offline analysis: Post #844 by Drdyer9051 (page 26)
- commaai/openpilot — selfdrive/car/chrysler/ headers for cross-referencing Chrysler IDs
- karlyamashita/common_libraries — CHRYSLER_CAN_ID.h header for Chrysler V5 platform IDs

1.2 Completed Projects Built With This Knowledge

Post #48 — jmccorm (Dec 6, 2021) [Direct Link]

- Black box data recorder — auto-logs CAN traffic on vehicle start/remote-start, closes file on shutdown
- Remote start HVAC auto-control — activates full heat or AC on remote start, with idle bump to 2000 RPM
- Remote keyfob WiFi toggle — double-tap unlock button toggles vehicle WiFi server on/off
- Live dash display (Python/curses) — real-time display of RPM, speed, gear, tire pressures, temps, 4x4 status, battery voltage
- EVIC custom text injection — pushes arbitrary artist/song/input text to the music screen

- Horn control via UDS — plays custom horn patterns using CAN ID 620 and UDS service 2F
- Idle speed control via UDS — sets engine to 2000 RPM for remote start warm-up, auto-cancels when script exits
- CAN bus clock sync — reads date/time from ID 350 to keep Pi clock synchronized without internet

2. Hardware & Physical Setup

2.1 Connection Points

Post #1 — jmccorm (Nov 7, 2021) [Direct Link]

CAN-C and CAN-IHS bus connectors are behind the glovebox. Both use 13-way star connectors that are self-terminating — no external resistor needed. Multiple open slots available without disturbing existing devices. The connector block provides direct, full-access bus connections that bypass the SGM entirely (unlike the OBD-II port which is SGM-filtered).

Additional tap points: A CAN-C connector is also accessible behind the rear door jamb on the passenger side (and a CAN-IHS version on the driver side), accessible by pulling back the carpeting. Useful for shorter cable runs.

Power: No convenient 12V feed directly at the glovebox CAN connector. An AUX USB power cable is available nearby. Route 12V from the center console or engine bay.

2.2 Recommended Hardware Configuration

Post #39 — jmccorm (Dec 3, 2021) [Direct Link]

```
Hardware:  Raspberry Pi 3B (Pi 4 has interrupt-sharing bug — avoid for dual-
channel)
Adapter:   Waveshare 2-Channel CAN HAT (MCP2515)
           Amazon: https://www.amazon.com/dp/B08942X9QB/
OS:        Raspbian Buster or Bullseye (not Bullseye on Pi 4)

Alternative hardware used by community members:
  - Adafruit Feather M4 CAN (tiny, battery-powered option)
  - Arduino MKR 1010 with CAN shield
  - USB CANable adapter + SavvyCAN on PC
  - ESP32 with TWAI (built-in CAN controller)
```

2.3 /boot/config.txt — Dual Channel (Pi 3B)

```
dtparam=spi=on
# CAN0 = CAN-IHS (125 kbps) Interior High Speed
dtoverlay=mcp2515,spi0-0,spimaxfrequency=2000000,interrupt=23
# CAN1 = CAN-C (500 kbps) Critical Automotive
dtoverlay=mcp2515,spi0-1,spimaxfrequency=2000000,interrupt=25
dtoverlay=spi0-2cs
```

NOTE — Pi 4 bug: Both CAN channels share interrupt 25. Only one works at a time. Use Pi 3B. Tested with two different Waveshare units; both exhibit same behavior on Pi 4.

2.4 Device Initialization Commands

```
# CAN-IHS: 125 kbps, listen-only, triple sampling
ip link set can0 type can bitrate 125000 listen-only on triple-sampling on
ifconfig can0 up

# CAN-C: 500 kbps
ip link set can1 type can bitrate 500000 listen-only on triple-sampling on
ifconfig can1 up

ifconfig can0 txqueuelen 65536
ifconfig can1 txqueuelen 65536

# Remove 'listen-only on' to enable transmit (verify error rate < 1% first!)
```

2.5 Auto-start on Boot

```
# /etc/network/interfaces.d/can0
auto can0
iface can0 can static
  bitrate 125000 listen-only off triple-sampling on
  post-up /sbin/ifconfig can0 txqueuelen 65536
```

2.6 Official Jeep CAN Connectors — Low-Cost Source

Post #110 — jmccorm (Jan 2022) [Direct Link]

Both the green (CAN-IHS) and white (CAN-C) 13-way star connectors are available from Mouser Electronics. Confirmed to plug into factory hubs without issue and with correct keying so they cannot be inserted into the wrong bus. Total cost for both connectors with terminals: under $2.00. Assembled using terminals from a Jeep CAN connector repair kit (crimping skill required).

2.7 CAN Bus Analysis Tools

- candump / cansniffer / cansend — standard Linux SocketCAN utilities, included with can-utils package
- SavvyCAN — PC application (Windows/Mac/Linux) for graphical CAN decoding. Load DBC files from github.com/darksotmoon/jlcan to auto-decode packets. Crashes frequently but powerful.
- isotpsend / isotpdump — for multi-frame UDS messages (some ECU replies exceed 8 bytes)
- bindechexascii — Raspbian utility: apt-get install bindechexascii — converts hex/decimal/binary
- printf for hex conversion: value=$(printf '%d' 0x$value)

3. Bus Architecture & Module Map

3.1 Official Name

Chrysler/FCA's official name for the CAN-C + CAN-IHS combination is 'PowerNet'. This is the same network shared across modern Dodge/Chrysler/Jeep/Ram vehicles.

3.2 CAN-C Bus — 500 kbps (Critical Automotive)

Modules: Secure Gateway Module (SGM), Radio, Emergency Assistance, Occupant Restraint Controller (airbags), Power Steering Pump, Instrument Cluster, PCM, Steering Column Control, ABS, TCM, BCM, TPMS, RF Hub, Auto Sway Bar, Electronic Shift, Steering Column Lock, Exhaust Temp, Occupant Classification, Drivetrain Control, Driver Detection, Star CAN-C Body/IP Hub, OBD-II/DLC (SGM-filtered). Optional: Park Assist, Headlamp Leveling, Diesel PCU, eTorque Battery Pack + MGU.

3.3 CAN-IHS Bus — 125 kbps (Interior High Speed / Comfort)

Modules: SGM, BCM, HVAC, Integrated Center Stack Switches, Tail Lamps (L+R), Radio, Emergency Assistance, Radio Amplifier, Comfort Seat / Steering Wheel, Star CAN-IHS IP/Body Hub, Data Link Connector 1. Optional: Sky One Touch Power Top.

3.4 Private Buses (Not Accessible)

- Electronic Shift Module ↔ TCM (direct)
- PCM ↔ NOx Sensors x2 + Particulate Matter Sensor (diesel)

4. Secure Gateway Module (SGW)

Post #38 — jmccorm (Dec 2, 2021) [Direct Link]

The SGW stands between the OBD-II/DLC port, the radio, and the rest of the vehicle CAN networks. It acts as two firewalls:

- OBD-II port: Read-only by default. Third-party tools can sniff all bus traffic but cannot write unless authenticated by Jeep corporate (via dealer connection). This is why JScan requires an SGW bypass cable to make config changes.
- Radio (uConnect): Only whitelisted messages pass. Prevents a hacked head unit from issuing drivetrain commands. Chrysler whitelists config messages (e.g. 'horn on lock' preference) but blocks safety-critical IDs.

Bypassing the SGW (with a bypass cable or Tazer): Grants full read/write CAN access via OBD-II port, and full access to the radio. The Tazer replaces the SGW entirely, bridging all three

connections. Behind-the-glovebox 13-way star connector ports already give full CAN access and do NOT need an SGW bypass.

OBD-II port is OUTSIDE the SGW for listening, INSIDE for writing. The glovebox star connector is FULLY inside — direct bus access in both directions.

5. Bus Timing, TTCAN, and Transmission Behavior

5.1 CAN ID Update Frequency Reference

Post #16 — jmccorm (Nov 9, 2021) [Direct Link]

CAN ID	Bus	Description / Notes
000-06F	Both	10ms (100 Hz) — fastest; critical safety/engine data. Counter+CRC protection.
070-0BF	Both	20ms (50 Hz)
0D0-0DF	Both	50ms (20 Hz)
100-1FF	Both	100ms (10 Hz)
200-FFF	Both	~1 sec or event-driven. Human/comfort systems, status flags.

5.2 TTCAN (Time-Triggered CAN)

Post #39 — jmccorm (Dec 3, 2021) [Direct Link]

The JL Wrangler buses appear to use TTCAN on top of standard CAN 2.0. Most time slots are pre-assigned to specific message IDs; only limited slots allow open CAN arbitration for event-driven messages. Suspected TTCAN controller IDs: 400, 401, 402, 403, 407 (possibly 409) on CAN-C; 401 and 403 on CAN-IHS.

Implication for translators and aftermarket devices: Transmissions may land in reserved time slots, causing the legitimate owner of that slot to lose its message. To mitigate: use irregular transmission intervals. Avoid common intervals (10ms, 100ms, 500ms, 1s) as they repeatedly collide with the same factory slots. Avoid bulk transmissions on CAN-C.

5.3 Bus Saturation

CAN-C measured at 58-59% saturation at engine idle. This is expected for a TTCAN system. Third-party tools that are not TTCAN-aware (like a Pi or Arduino) face roughly 60% chance of collision on any given transmission attempt.

5.4 Data Protection (Counter + Checksum)

CAN IDs below 0x0100 use additional protection: an incrementing counter byte (prevents spoofing/stuck detection) and a checksum byte (sum of other 7 bytes). Writing to these IDs with incorrect counter/CRC will generate a stored DTC. Safety-critical IDs (brakes, steering, airbags, acceleration) all use this protection.

5.5 Wake / Sleep Behavior

- Vehicle off: no CAN traffic on either bus.
- Wake: Accessory mode, Run mode, engine start, or keyfob press (partial wake).
- Software wake: cansend can0 2D3#0700000000000000 (fake center button press, no buttons pressed)
- IDs 400-402: appear during wake/sleep sequences — bus status messages.
- IMPORTANT: If vehicle doesn't sleep, a module is staying active and will drain battery. Disconnect your interface.

6. Decoded CAN Message IDs — Complete Reference

Convention: can0 = CAN-IHS (125 kbps), can1 = CAN-C (500 kbps). All IDs are hex. Unverified entries noted.

6.1 Engine, Drivetrain & Speed

Post #25 — jmccorm (Nov 12, 2021) [Direct Link]

CAN ID	Bus	Description / Notes
322	CAN-IHS	Engine RPM — confirmed, live-decoded in real-time drive logs.
340	CAN-IHS	Vehicle speed (MPH/KMH), gear selector position (PARK=0x0D, REVERSE=0x0B, NEUTRAL=0x00, DRIVE=0x01). Full drive log in Post #37. ESS status bit also present.
08B	CAN-C	Individual tire speed — each of 4 tires, unit = MPH * 20 (decimal). Resolution 0.05 MPH. Updates at 50 Hz. Note: reads slightly lower than speedometer ID 340 (speedo has calibration offset).
128	CAN-C	Power steering pump fluid temperature and pressure. Confirmed vs JScan.
296	CAN-C	Tire pressures (TPMS) — 4 bytes, one per tire, straight PSI in hex (e.g. 0x24 = 36 PSI). Updates every 1-2 seconds while driving; static when parked unless using Tazer tire-fill feature.
137	CAN-C	ESS (Engine Stop/Start) status bit. Also appears in ID 340.
277	CAN-C	4x4 status and front axle disconnect status. Related to locker operation.

6.2 Steering & Suspension

CAN ID	Bus	Description / Notes
023	CAN-C	Steering wheel angle and torque. 10 00 = dead center. Counter+CRC protected. 100 Hz.
025	CAN-C	Suspected X/Y/Z acceleration (companion to 02B).
027	CAN-C	Additional IMU / motion data.
02B	CAN-C	Yaw / pitch / roll (IMU). Counter-protected. 100 Hz.
07F	CAN-IHS	Wheel travel counter — one unsigned byte per wheel, increments with any movement. After ~5 feet, 0xFF flag set.
24E	CAN-IHS	Axle rotation + 2-axis data (front and rear, 6 channels total). Updates at 10 Hz.
252	CAN-IHS	Additional axle rotation / axis data (companion to 24E).
4CA	CAN-C	Sway bar disconnect status (suspected).

6.3 Sway Bar & Differential Lockers

Post #305 — redracer / community (Dec 2021) [Direct Link]

Discovered by redracer on page 9. These are the CAN IDs controlling the Rubicon sway bar disconnect and axle lockers:

CAN ID	Bus	Description / Notes
26F	CAN-IHS	Sway bar request. 00 40 FF FF FF 7F 7F 7F = unlock/lock toggle. Needs testing.
371	CAN-IHS	Sway bar status. 51 02 = unlock in progress; 70 01 = unlocked; 42 02 = lock in progress; 61 00 = locked.
25D	CAN-IHS	Locker request. 23 FC 00 00 7F 7F 7F 7F = rear locker. 13 FC 00 00 7F 7F 7F 7F = front+rear. 43 FC 00 00 7F 7F 7F 7F = unlock both.
2C2	CAN-IHS	Locker status. 14 B3 78 00 = unlocked. 14 B3 77 00 = locked (rear only or all — unconfirmed).
277	CAN-C	Related to locker operation (correlation not fully decoded).

6.4 Braking

CAN ID	Bus	Description / Notes
02F	CAN-C	Brake booster / system pressure. Strong positive numbers when braking, rare small negatives. Smoother than pedal position sensor.
087	CAN-C	Brake activation (pedal position). Counter byte present.
028	CAN-C	Brake-related (companion to 087).
079	CAN-C	Brake-related. Lowest two bytes increment as counter.
083	CAN-C	Brake-related. Counter present.
093	CAN-C	Brake-related.

127	CAN-C	Brake-related.
12B	CAN-IHS	RAP (Remote Accessory Power) status — 1 byte on IHS. NOTE: Same ID on CAN-C is 8 bytes with different content. Use message length to identify which bus.
407	CAN-C	Possible 3rd brake light activation bitmap.

6.5 Vehicle State & Power

CAN ID	Bus	Description / Notes
122	CAN-IHS	Power mode. 0x00=OFF, 0x03=Accessory, 0x04=Run/Ignition, 0x15=Crank. Mask: 0x1F.
350	Both	Date and time (from uConnect radio). Broadcasts 1/sec. Bytes: seconds, minutes, hours, year (2 bytes LE), month, day — all BCD hex. 0xFF = clock not initialized. Readable via gettime script.
3D2	CAN-IHS	Odometer in tenths of a kilometer (always metric regardless of display setting).
3E0	Both	Vehicle VIN — 3 multipart frames, each ~0.1 sec apart. Frame 0 (byte 0x00) = chars 1-7, Frame 1 (0x01) = chars 8-14, Frame 2 (0x02) = chars 15-17.
400	CAN-C	Suspected primary TTCAN controller / time reference message.
401	Both	TTCAN bus controller / wake-sleep sequence.
402	CAN-C	Shutdown / sleep sequence.

6.6 Lighting, Turn Signals & Horn

Post #130 — redracer (Jan 2022) [Direct Link]

CAN ID	Bus	Description / Notes
291	CAN-C	Lighting master: headlights, running lights, cabin lights, interior dimmer, turn signals. Byte layout: Byte 5 offset: 0x00=off, 0x01=running lights on, (0x03 not confirmed) Byte 6 offset: 0x00=off, 0x02=low beam, 0x04=high beam Byte 7 offset: interior dimmer 0x22=lowest, 0xC8=highest Byte 8 offset: 0x00=off, 0x01=cabin lights on Also: cansend can1 291#000000F000000000 = briefly dims dash + turn signal click sound
318	Both	Turn signal stalk + wipers. Bit 0x01=left blink, 0x02=right blink, 0x04=hold high beam, 0x08=momentary high beam, 0x10=momentary wipers.
3B2	CAN-IHS	Lighting status (full state).
2FA	CAN-IHS	Dimmer / door status / parking brake / power locks.
13F	CAN-C	Power consumption — bits change on AUX buttons, headlights, wipers.
620	CAN-C	Horn control via UDS. Pattern: 02 10 03 to enter extended diagnostic session; then 05 2F D0 AD 03 01 00 00 = horn ON, 05 2F D0 AD 03 00 00 00 = horn OFF.

6.7 HVAC & Climate Control

Post #18 — jmccorm (Jan 2022) [Direct Link]

All HVAC commands go through CAN-IHS ID 0x2D3. The vehicle's existing HVAC module broadcasts current state 10x/sec over 2B1 and 332, which will undo any changes you make unless you update at a higher rate or use UDS direct module control. ID 342 controls driver/passenger sync.

CAN ID	Bus	Description / Notes
2D3	CAN-IHS	HVAC control commands (center dash buttons). Key bit values: 0700000000010000 = HVAC power on/off 0700000000000100 = A/C compressor on/off 0700000000000200 = Air recirculation on/off (also turns A/C on automatically) 0700000000040000 = Driver temp UP 0700000000080000 = Driver temp DOWN 0700000000100000 = Passenger temp UP 0700000000200000 = Passenger temp DOWN 0700000008000000 = Fan speed UP 0700000004000000 = Fan speed DOWN
342	CAN-IHS	Driver/passenger temperature sync. 0000000400 = engage sync.
2B1	CAN-IHS	Fan speed and A/C baffle. Byte 0: fan speed 0x01-0x07, 0x00=auto.
230	CAN-IHS	HVAC button status. 0x01=A/C off, 0x03=A/C on, 0x07=defrost, 0x00=all off.
332	CAN-IHS	Actual current fan speed (sensing). 6th byte.

HVAC Vent Mode Commands (via 2D3 ventmode helper)

CAN ID	Bus	Description / Notes
ventmode 00	2D3	Front defroster (clears all other modes — use as reset)
ventmode 02	2D3	Windshield + floor vents
ventmode 08	2D3	Panel (main dash) vents

6.8 Steering Wheel Controls

CAN ID	Bus	Description / Notes
22D	CAN-IHS	Left-cluster buttons. Byte 3+4: Left=0x10 0x00, Right=0x00 0x01, Down=0x40 0x00, Up=0x00 0x04.
0B1	CAN-IHS	Cruise control buttons. Highest byte = button state; next two = counters. Also confirmed as cruise + button ID.
07D	CAN-C	Cruise control related (reported by community; 0B1 confirmed as primary cruise button ID).

6.9 EVIC Display

Post #5 — jmccorm (Nov 7, 2021) [Direct Link]

CAN ID 328 (CAN-IHS) controls the EVIC music screen. Send to can0. Characters are 2-byte UTF-16LE (ASCII uses 00 high byte). Confirmed working — custom text injected successfully. Bottom line MUST be sent first (it clears all fields). Each line terminated with all-zero message. Delay 10ms between frames.

CAN ID	Bus	Description / Notes
328	CAN-IHS	EVIC music screen. Line types: 0x01=Input Name (bottom/clears screen), 0x02=Artist (middle), 0x03=Song Title (top). Byte 0: upper nibble=lines remaining, lower nibble=0x4 for start of new line.

6.10 Lock/Unlock, Remote Control & Keyfob

CAN ID	Bus	Description / Notes
1C0	CAN-C	Remote keyfob / uConnect app commands (lock, unlock, remote start, panic).
1C8	CAN-C	Remote lock/unlock companion.
09B	CAN-C	Driver door status (suspected).

6.11 Audio / Radio

CAN ID	Bus	Description / Notes
25D	CAN-IHS	Mute audio active (does not track mute button directly). Also rear locker request in some frames — verify context.
30F	CAN-IHS	Radio audio status.
30D	CAN-IHS	Radio phone/handsfree audio status.
273	CAN-IHS	Volume/tune knob.
354	CAN-IHS	Amplifier status.

6.12 Gas Pedal, Throttle & Transmission

CAN ID	Bus	Description / Notes
081	CAN-C	Gas pedal position — value in a byte; inverse mirrored in multiple locations.
09D	CAN-C	Gas pedal / throttle valve (companion to 081). Black box log confirms: 'GASPEDAL: 34% (VALVE 33%)' format.
340	CAN-IHS	Also contains brake % from pedal sensor alongside speed/gear. (See Section 6.1)

6.13 Temperature & Environmental

CAN ID	Bus	Description / Notes
035	CAN-IHS	Cabin/vent temperature.
370	CAN-IHS	Compass heading (magnetic and true north) + odometer data.

350	Both	Date and time broadcast. (See Section 6.5)

Temperature via UDS (Section 7 covers protocol details): Cabin temp, steering wheel temp, and ambient temp are all readable via UDS Service 22. Formula: raw byte (hex) converted to decimal, minus 54 = Fahrenheit. Range: -54°F to 201°F.

6.14 Miscellaneous & Tazer-Related

CAN ID	Bus	Description / Notes
2D3	CAN-IHS	Center dash buttons (full HVAC + system controls — see Section 6.7).
2FA	CAN-IHS	Door status (first byte = which doors open), power locks (4th byte).
4C2	CAN-C	Tazer light show in progress (first 5 bytes distinctive). Not seen in normal operation.
504	CAN-C	Used by JScan/Tazer for interior control bitmaps.
620	CAN-C	Used by JScan/Tazer; also horn UDS commands (see Section 6.6).
742	CAN-C	Only seen during Tazer light show operation.

7. Unified Diagnostic Services (UDS) Over CAN

7.1 Overview

Post #11 — jmccorm (Nov 8, 2021) [Direct Link]

UDS (ISO 14229) is a diagnostic protocol layered on top of CAN. The Tazer uses UDS to directly override vehicle outputs — enabling features like Light Show (forces lights on/off regardless of switch state). UDS allows direct module output override using mechanisms originally designed for factory testing. UDS is what enables the high-quality, persistent behavior of Tazer features, rather than simple CAN message-rate-racing.

UDS operations require entering an Extended Diagnostic Session (Service 0x10, subfunction 0x03 or 0x92). While in a diagnostic session, the ECU expects a 'Tester Present' keepalive message (Service 0x3E subfunction 0x00) every ~250ms. When the keepalive stops, the ECU returns to normal mode and cancels any UDS-commanded overrides.

7.2 Module Addressing (CAN-C Bus)

CAN ID	Bus	Description / Notes
7DF	CAN-C	OBD-II functional broadcast — queries all ECUs simultaneously
7E0	CAN-C	PCM/ECM (Powertrain Control Module) — primary UDS target

7E8	CAN-C	PCM response address
7E1	CAN-C	TCM (Transmission Control Module)
7E9	CAN-C	TCM response address
620	CAN-C	BCM (Body Control Module) — horn, lights, locks via UDS
504	CAN-C	Module (used by Tazer/JScan for adaptations)
740+	CAN-C	Additional module addresses: 740, 743, 747, 749, 74B, 762 (DTC clearing sequence)

7.3 UDS via isotpsend / isotpdump (Multi-Frame)

Post #209 — jmccorm (Jan 2022) [Direct Link]

For UDS commands longer than 8 bytes (or responses longer than 8 bytes), use isotpsend and isotpdump instead of cansend/candump:

```
# Enter extended diagnostic session on PCM
echo '10 92' | isotpsend -s 7E0 -d 7E8 -p 00:00 -P a can1
sleep 0.25

# Keepalive (tester present) — send every 250ms while session active
echo '3E 00' | isotpsend -s 7E0 -d 7E8 -p 00:00 -P a can1

# Set engine RPM to 2000 via UDS
echo '31 05 07 D0' | isotpsend -s 7E0 -d 7E8 -p 00:00 -P a can1
```

7.4 Reading Temperatures via UDS (Service 0x22 — Read Data by ID)

Post #180 — jmccorm (Jan 2022) [Direct Link]

UDS Service 22 (Read Data By ID) allows polling specific module parameters. The temperature formula applies to all temperature reads: raw byte (decimal) - 54 = degrees Fahrenheit. Range: -54°F to 201°F. Formula verified for cabin temp, steering wheel temp, and ambient temp.

```
# Cabin temperature (BCM on CAN-C, module address 620, response on 504)
( sleep 0.1 ; cansend can1 620#0322A01000000000 ) &
COMMAND='timeout -s 1 0.6 /usr/bin/candump -L can1,0504:0FFF'
RESPONSE=$( $COMMAND | grep '#100A62A010' | cut -d# -f2 | cut -c11-12 | tail -1 )
RESPONSE=$( printf '%d' 0x$RESPONSE )
RESPONSE=$( expr $RESPONSE - 54 )
echo ${RESPONSE}F

# Steering wheel temperature (CAN-C, different module address)
# Uses same formula: decimal - 54 = degrees F
# Uses isotpsend for the query
```

7.5 OBD-II Mode 01 PIDs via UDS on CAN-C

Post #166 — jmccorm (Dec 2021) [Direct Link]

Send to 7E0, receive on 7E8. Format: cansend can1 7E0#0201{PID}0000000000

CAN ID	Bus	Description / Notes

PID 0C	CAN-C/7E0	Engine RPM. Formula: ((A*256)+B)/4. Range: 0-16383.75 rpm.
PID 0D	CAN-C/7E0	Vehicle speed. Formula: A (km/h). Range: 0-255.
PID 05	CAN-C/7E0	Engine coolant temperature. Formula: A-40 (°C). Range: -40 to 215°C.
PID 11	CAN-C/7E0	Throttle position. Formula: A*100/255 (%). Range: 0-100%.
PID 04	CAN-C/7E0	Calculated engine load. Formula: A*100/255 (%).
PID 0F	CAN-C/7E0	Intake air temperature. Formula: A-40 (°C).
PID 0E	CAN-C/7E0	Timing advance. Formula: A/2-64 (degrees).
PID 10	CAN-C/7E0	MAF air flow rate. Formula: ((A*256)+B)/100 (g/s).

7.6 PCM UDS Service 22 PIDs (Advanced — Page 50)

Discovered by community members using specialized scan tools on the PCM (address 7E0):

CAN ID	Bus	Description / Notes
01 AE	7E0	Knock retard. Formula: value * 0.5 = degrees. In normal highway driving, knock retard is often 0.
02 27	7E0	Fuel-related PID (formula TBD from context).

8. Code Samples & Scripts

8.1 Wake Up CAN Bus

Post #29 — jmccorm (2022) [Direct Link]

```
#!/bin/bash
# Safe version — sends a 'no buttons pressed' center dash message
cansend can0 2D3#0700000000000000
```

WARNING: Using random bytes in byte 8 of this message can inadvertently turn off the uConnect radio. Stick to the above.

8.2 Common candump / cansniffer Commands

```
# Listen to all traffic on any interface
candump -tA -a -c -d -e -x any

# Filter to specific ID (e.g. ID 087 only)
candump -ta -c -a -d -e -x can0,087:fff

# Filter to an ID range (e.g. IDs 400-4FF)
candump -ta -c -a -d -e -x can0,ffff:0400

# Log to file
candump -L can0 > my_log.log
```

```
# Replay a log file (for analysis with virtual CAN)
canplayer -I my_log.log

# Interactive sniffer with change highlighting
cansniffer -B -c -v 0x3ff -m 0xf00 can0
cansniffer -B -c -v 0x03ff -m0xf00 -v 0x04ff -t 100 -h 20 -l 4 can0

# Send a test message
cansend can0 328#00.00.00.00.00.00.00.00

# Check bus saturation
canbusload can0@125000
canbusload can1@500000

# Detailed interface stats (error rate check)
ip -details -statistics link show

# Shut down interface
ifconfig can0 down
```

8.3 Get Clock from CAN (Bash)

Post #40 — jmccorm (Dec 5, 2021) [Direct Link]

```
#!/bin/bash
# Reads current date/time from CAN ID 0x350 on CAN-IHS (can0)
# Output: 12/5/2021 13:11:44
COMMAND='timeout -s 9 1 /usr/bin/candump -L can0,0350:0fff'
can350=$( $COMMAND | tail -1 )
if [ "$can350" == "" ] ; then echo 'ERROR: vehicle off'; exit 1; fi
if [ "$( echo "$can350" | cut -c30-43 )" == 'FFFFFFFFFFFFFF' ] ; then echo 'ERROR:
clock not set'; exit 1; fi
rawtime="$( echo "$can350" | cut -d# -f2 )"
rawtime="$( echo "$rawtime" | fold -w2 | paste -sd' ' )"
rawtime="$( echo "$rawtime" | cut -c1-11,13-20 )"
time="$( echo $rawtime | { read sec min hr yr mo dy ;
  printf '%d/%d/%d %02d:%02d:%02d\n' 0x$mo 0x$dy 0x$yr 0x$hr 0x$min 0x$sec ; } )"
echo "$time"
```

8.4 EVIC Display Injection (Bash)

Post #15 — jmccorm (Nov 8, 2021) [Direct Link]

```
#!/bin/bash
# SEND BOTTOM LINE FIRST — it clears all other fields
cansend can0 328#20.41.00.42.00.6C.00.75  # Input Name
sleep 0.01
cansend can0 328#10.01.00.65.00.46.00.6F
sleep 0.01
cansend can0 328#00.01.00.78.00.00.00.00
sleep 0.01
cansend can0 328#00.00.00.00.00.00.00.00  # terminate
sleep 0.1
# Then send Artist (line type 0x02) and Song Title (0x03) in same format
```

8.5 Idle Speed Control via UDS (Bash)

Post #209 — jmccorm (Jan 2022) [Direct Link]

```
#!/bin/bash
# Raises engine to 2000 RPM via UDS. Auto-cancels when script exits.
# Enter extended diagnostic session
echo '10 92' | isotpsend -s 7E0 -d 7E8 -p 00:00 -P a can1 ; sleep 0.25
echo '3E 00' | isotpsend -s 7E0 -d 7E8 -p 00:00 -P a can1 ; sleep 0.25
# Set engine to 2000 RPM (0x07D0 = 2000 decimal)
echo '31 05 07 D0' | isotpsend -s 7E0 -d 7E8 -p 00:00 -P a can1
# Keepalive loop — ECU returns to normal when this stops
while [ 1 ]; do
  sleep 0.25
  echo '3E 00' | isotpsend -s 7E0 -d 7E8 -p 00:00 -P a can1
done
```

8.6 Horn Script via UDS (Bash)

Post #116 — jmccorm (Dec 2021) [Direct Link]

```
#!/bin/bash
cansend can0 2D3#0700000000000000 ; sleep 0.1  # wake bus
# Enter extended diagnostic session (BCM on CAN-C, ID 620)
cansend can1 620#02.10.03.00.00.00.00.00 ; sleep 0.1
# 5 quick horn taps
for i in $(seq 1 5); do
  cansend can1 620#05.2f.d0.ad.03.01.00.00 ; sleep 0.025  # horn ON
  cansend can1 620#05.2f.d0.ad.03.00.00.00 ; sleep 0.025  # horn OFF
done
```

8.7 AutoHeat Script (Remote Start HVAC — Bash)

Post #190 — jmccorm (Jan 2022) [Direct Link]

Full HVAC automation for remote start. Sets full heat, recirculates air, mutes radio, syncs driver/passenger. Uses ID 2D3 (HVAC commands) and 342 (temp sync). Verified to bring cabin from 20°F to 65°F in 15 minutes with engine warm-up routine.

```
#!/bin/bash
# Key 2D3 commands used in autoheat:
cansend can0 2D3#0700000000000000   # wake bus
# ventmode 00 = front defroster (reset all modes)
# ventmode 02 = windshield + floor
cansend can0 2D3#0700000000000200   # air recirculation ON (also enables A/C)
# Step driver temp up 26 times in 0.36s intervals
cansend can0 2D3#0700000000040000   # driver temp UP
cansend can0 2D3#0700000000000100   # A/C OFF (was auto-enabled by recirculate)
# Sync driver/passenger
cansend can0 342#0000000400
```

8.8 Live Dashboard Script (Python)

Post #410 — community (2022) [Direct Link]

Python curses-based live display reading multiple CAN IDs simultaneously. Uses python-can library. Monitors: battery voltage, roll/tilt/yaw, RPM, MPH, gear, 4x4 status, steering angle, tire pressures (4), brake activation/pedal, motion, temps (IAT/coolant), lights.

```
#!/usr/bin/python3
import can, curses
```

```
# Filters IDs: 0x2C2 (locker), 0x02B (IMU), 0x291 (lights),
#              0x322 (RPM), 0x340 (speed/gear), 0x296 (TPMS),
#              0x277 (4x4), 0x128 (PS pump), 0x087 (brakes), etc.
bus = can.interface.Bus('', bustype='socketcan',
  filter=[{'can_id': 0x2C2, 'can_mask': 0xFFF}, ...])
```

9. OBD-II Standard PID Reference

Send to CAN-C via ID 7E0 (PCM). Response on 7E8. Format: cansend can1 7E0#0201{PID}0000000000. Use bash script from page 15 for automated OBD-II PID reading.

CAN ID	Bus	Description / Notes
PID 0C	7E0→7E8	Engine RPM. Formula: ((A*256)+B)/4. 2 bytes.
PID 0D	7E0→7E8	Vehicle speed km/h. Formula: A. 1 byte.
PID 05	7E0→7E8	Engine coolant temp. Formula: A-40°C. 1 byte.
PID 04	7E0→7E8	Engine load %. Formula: A*100/255. 1 byte.
PID 11	7E0→7E8	Throttle position %. Formula: A*100/255. 1 byte.
PID 0F	7E0→7E8	Intake air temp. Formula: A-40°C. 1 byte.
PID 0E	7E0→7E8	Timing advance degrees. Formula: A/2-64. 1 byte.
PID 10	7E0→7E8	MAF air flow g/s. Formula: ((A*256)+B)/100. 2 bytes.
PID 06	7E0→7E8	Short term fuel trim Bank 1 %. Formula: (A-128)*100/128.
PID 07	7E0→7E8	Long term fuel trim Bank 1 %. Same formula.
PID 0A	7E0→7E8	Fuel pressure kPa gauge. Formula: A*3.
PID 0B	7E0→7E8	Intake manifold pressure kPa. Formula: A.

10. Chrysler V5 Platform Cross-Reference

The JL Wrangler uses a Chrysler-based ECU (not inherited from the JK or FIAT). Many IDs match the Chrysler V5 header file (karlyamashita/common_libraries). Gear and power mode values confirmed identical.

CAN ID	Bus	Description / Notes
0x122	IHS	POWER_MODE_ID — confirmed JL
0x22D	IHS	SWC_ID (audio controls) — confirmed JL
0x273	IHS	VOLUME_TUNE_KNOB_ID
0x2D3	IHS	DASH_BUTTONS_ID — confirmed JL
0x2FA	IHS	DIMMER_DOOR_PARKING_BRAKE_ID — confirmed JL
0x30D	IHS	RADIO_PHONE_AUDIO_STATUS_ID
0x30F	IHS	RADIO_AUDIO_STATUS_ID
0x318	IHS	SWC_TURN_SIGNAL_ID — confirmed JL
0x322	IHS	RPM_SPEED_ID — confirmed JL

0x332	IHS	BRAKE_GEAR_INFO_ID — confirmed JL
0x340	IHS	VEHICLE_SPEED_ID — confirmed JL
0x354	IHS	AMP_STATUS_ID
0x3B2	IHS	LIGHTING_STATUS_ID — confirmed JL
0x3E0	Both	VIN_ID — confirmed JL
0x12B	IHS	RAP_ID — confirmed JL (1-byte version on IHS)
0x2A8	IHS	TURN_SIGNALS_SWITCH_ACTIVE_ID

```
# Gear values (Chrysler V5, confirmed JL):
GEAR_PARK    = 0x0D
GEAR_REVERSE = 0x0B
GEAR_NEUTRAL = 0x00
GEAR_DRIVE   = 0x01  (specific gear 1-8 in second nibble)

# Power mode values:
POWER_MODE_OFF       = 0x00
POWER_MODE_ACCESSORY = 0x03
POWER_MODE_IGNITION  = 0x04
POWER_MODE_CRANK     = 0x15
POWER_MODE_MASK      = 0x1F
```

11. LS Swap / Engine Swap CAN Translator Notes

This section summarizes the minimum viable set of CAN messages a translator needs to generate on CAN-IHS to keep the factory Jeep dash alive after an engine swap.

- RPM → ID 0x322 on CAN-IHS. Rate: ~50 Hz (20ms). Decode formula from raw sniffing needed; bytes confirmed but exact scaling TBD.
- Vehicle speed → ID 0x340 on CAN-IHS. Rate: ~10 Hz (100ms). Also contains gear position (use 0x01 for 'Drive' if your ECU doesn't report individual gears, or map actual gear). Coolant temp may also be in 0x340 — verify with sniff.
- Coolant temp → Most likely in 0x340 or a 0x3xx ID. OBD-II PID 0x05 formula (A-40°C) applicable if polled via UDS. Sniff with temp variation to find the passive broadcast ID.
- Power mode → ID 0x122 on CAN-IHS. Value 0x04 (Run) keeps dash from throwing errors. Translator should transmit this continuously at ~1 Hz.
- TPMS → If not needed, no action required. Dash will show TPMS warning but won't cause other issues.
- Bus rate: CAN-IHS is 125 kbps — within range of MCP2515-based adapters and ESP32 TWAI.
- TTCAN: Transmit at irregular intervals (e.g. 19ms, 23ms, 17ms) not exact 20ms to avoid repeated collisions with the same TTCAN slot.
- EVIC injection: ID 0x328 works independently — can push custom text to music screen to show engine status (coolant temp, oil pressure, etc.) as a debug/status display.
- SGW bypass: If connecting behind glovebox star connector, no SGW bypass needed. If using OBD-II port for any writes, SGW bypass required.

- SavvyCAN + DBC files (darksotmoon/jlcan): Most efficient way to identify which bytes encode what. Sniff while varying one engine parameter at a time.

12. Safety Notes & Best Practices

Post #39 — jmccorm (Dec 3, 2021) [Direct Link]

- Start in listen-only mode. Verify error rate under 1% (via 'ip -details -statistics link show') before transmitting.
- Never fuzz (send random data to random IDs). Risk includes airbag deployment and bricked ECUs.
- IDs below 0x0100 are safety-critical (airbags, brakes, steering). Only write to fully decoded IDs.
- Shorting the CAN bus or pulling a wire will throw DTCs but not cause permanent damage. If you see loss-of-comms messages on dash, unplug Pi immediately.
- CAN bus is designed to survive single-wire failures (differential signaling). Corrosion at terminals behind glovebox can disrupt bus — keep area dry.
- Writing to a CRC/counter-protected ID with incorrect values generates a DTC. Can be cleared like any code, but it's logged.
- UDS diagnostic sessions must be continuously refreshed (keepalive 'Tester Present' every 250ms). If your script crashes, UDS commands auto-cancel — this is a safety feature, not a bug.
- Do not use common transmission intervals (10ms, 100ms, 1s) to avoid repeated TTCAN collisions.
- Use Raspberry Pi 3B with Waveshare 2-CH HAT. Pi 4 has confirmed interrupt-sharing hardware bug preventing dual-channel operation.
- Before writing to an ID, read and record its current value. Keep notes of all changes made.

13. Key Links & Post Index

Thread homepage (all 52 pages)
CAN Message ID Spreadsheet (Google Sheets)
GitHub: rstellhorn/jeep-jl-powernet-scripts
DBC files for SavvyCAN: darksotmoon/jlcan
commaai/openpilot (Chrysler CAN IDs in selfdrive/car/chrysler/)
karlyamashita/common_libraries (CHRYSLER_CAN_ID.h)
Waveshare 2-CH CAN HAT (Amazon)

Key posts by content:

Post #1 — Project overview, initial CAN IDs, spreadsheet link
Post #9/#16 — Bus timing chart, update rates by ID range

Post #25 — First confirmed RPM/speed/gear live drive log
Post #37 — 4mi drive log file download (47MB candump)
Post #38 — SGW deep dive, OBD-II port architecture, wiring diagrams
Post #39 — Complete hardware setup guide, TTCAN, best practices
Post #40 — gettime bash script (clock from CAN ID 350)
Page 9 — Sway bar and locker CAN IDs (redracer discovery)
Page 10 — Cabin temperature via UDS, UDS script framework
Page 12 — UDS idle speed control (isotpsend)
Page 13 — Remote start HVAC project results (83°F in 15 min)
Page 15 — OBD-II over CAN bash script; 291 lighting deep decode
Page 18 — AutoHeat/AutoCool scripts; WiFi keyfob toggle
Page 24 — Python live dashboard; tire pressure decode (0x296)
Page 26 — Virtual CAN / canplayer setup for offline analysis
Page 50 — SavvyCAN/DBC workflow; PCM UDS PIDs (knock retard)
Page 51 — Cruise control ID (0B1); A/C compressor via 2D3 confirmed

Summary compiled from all 52 pages of the thread. Thread remains active; check source for updates beyond March 2023.