Introduction to SAE J1939

A primer for in-vehicle networking

PREPARED BY DR. JEREMY DAILY

SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

Jeremy S. Daily, Ph.D., P.E.

- U.S. Air Force (1995-2002)
  - Electronic Maintenance of Meteorological and Navigation Systems
  - Wright-Patterson Air Force Base, Dayton, OH
- Ph.D. in Engineering from Wright State University (2006)
- Associate Professor of Mechanical Engineering
  University of Tulsa (2006)
- Founder of Synercon Technologies, LLC (2013)
  - Commercialized digital forensic technology developed at U. Tulsa.
  - Sold Synercon to the Dearborn Group in 2018.
- Co-Founded the CyberTruck Challenge (2017)
- Treasurer for the CyberAuto Challenge (2021)
- Co-Founded the CyberBoat Challenge (2022)
- Instructor for the CyberTractor Challenge (2022)
- Research Performer for NSF, DARPA, DOJ, and NMFTA
- Associate Professor of Systems Engineering at Colorado State University (2019)

Agenda

- Vehicle Systems and Communications
- Networking and Wiring Diagrams
- Connecting to the Network
- Interpreting Data with J1939
- J1939 Transport Protocol
- J1939 Address Claims
- Diagnostics
  - SAE J1939-73
  - ISO14229 - UDS
  - Proprietary protocols
- Cybersecurity Challenges

Training Goals

Understand the need for in-vehicle communication using CAN and SAE J1939

Connecting to J1939 Networks

Interpret J1939 network traffic using the SAE Standard

Recognize SAE J1939 Transport Protocols for larger messages

Introduction to J1939 Address Claiming

Demonstration of RP1210 functionality for diagnostics

Realize J1939 is inherently an open (and potentially insecure) read-write bus

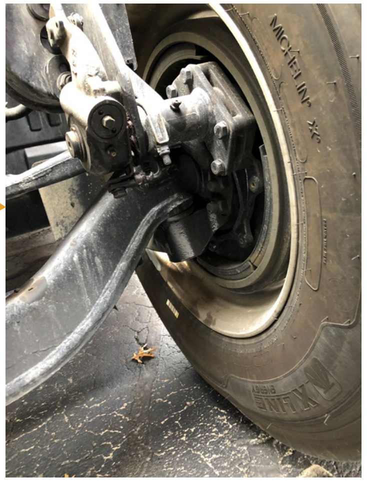
Truck Systems

Primary Functions

- Go – Convert fuel into mechanical energy to accelerate heavy loads
- Stop – Brake the tractor-trailer systems, often with anti-locking air brakes
- Steer – Give the driver the ability to guide the vehicle
- Haul – Support heavy loads and pull trailers

Additional Functions

- Protect – Restrain occupants in a crash. Assist drivers to avoid crashes.
- House – Provide places to sleep while on a long haul
- Entertain – Radio, CDs, Bluetooth, and Satellite options
- Monitor – Telematics and fleet management
- Diagnose – Provide information related to vehicle operation and potential faulty parts
- Comply – US DOT regulation, EPA and emissions regulations



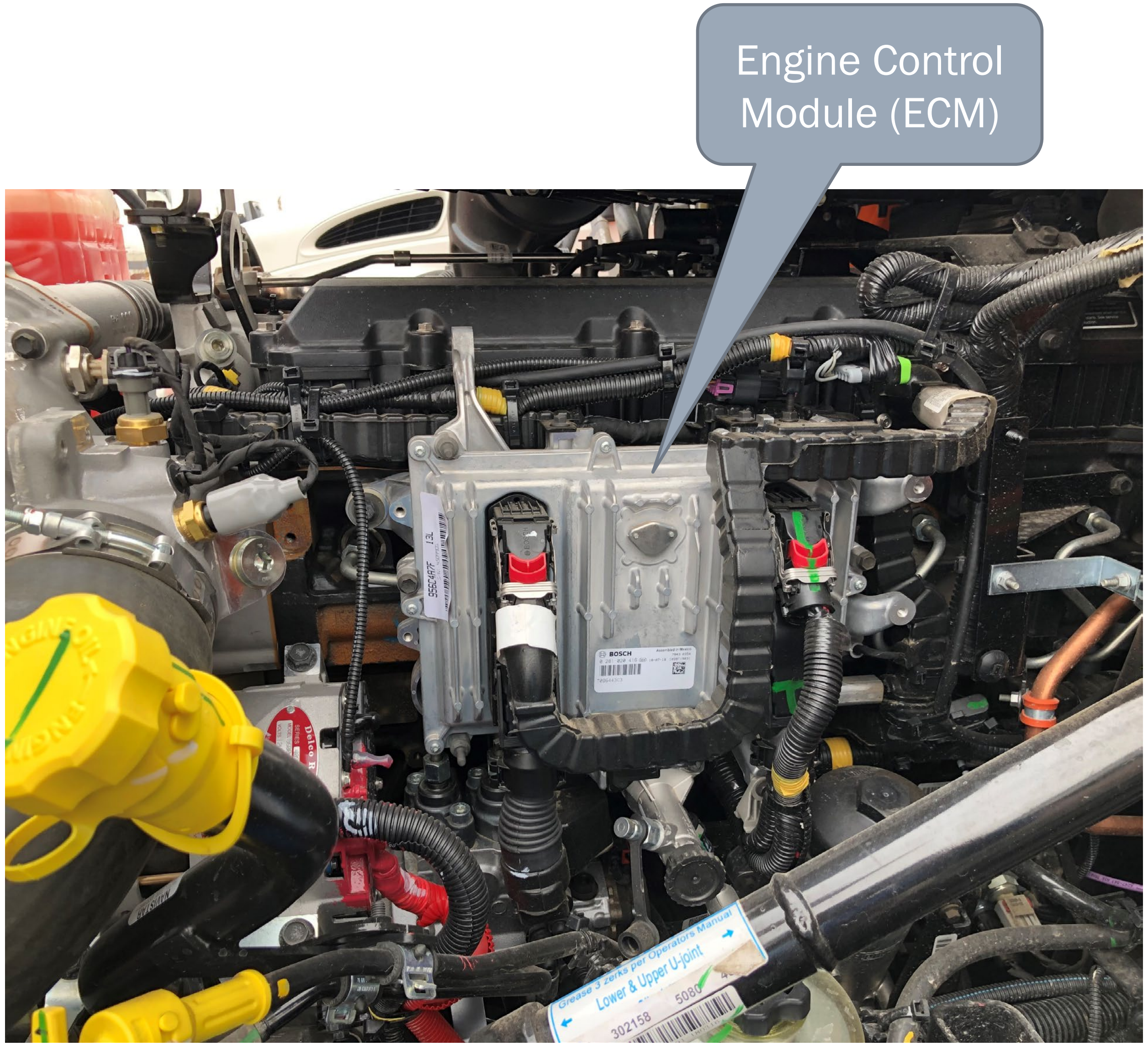
Truck Engines

Primary function to efficiently produce motive power

Also:

- Comply with emission requirements
- Aid diagnostics and troubleshooting
- Record driving and diagnostic events
- Additional Power for
  - Compressed Air
  - Power take off (PTO) equipment
  - Electrical systems

Computer controls are paramount to realize these functions



The driver's side of a Navistar A26 Engine in an International LT truck.

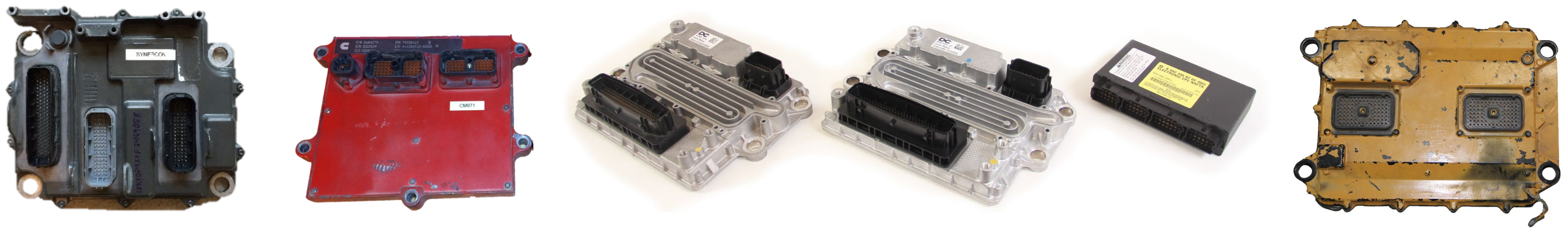
Engine Control Module (ECM)

The ECM is an electronic control unit (ECU) primarily responsible for the operation of the engine.

Single ECUs that are engine mounted (Cummins, PACCAR, Navistar)

Multiple ECUs communicating over a network (Detroit Diesel, Volvo)

Engine controllers are often the most advanced and expensive electronic units on a vehicle



Data from an ECM

Live Status Data

- Data broadcast to all ECUs regarding current status and operation
- Examples: Engine speed, accelerator pedal position, wheel-based vehicle speed and many others

Configuration Data

- Does not change with time
- Features, Parameters, Calibration Settings

Historical Data

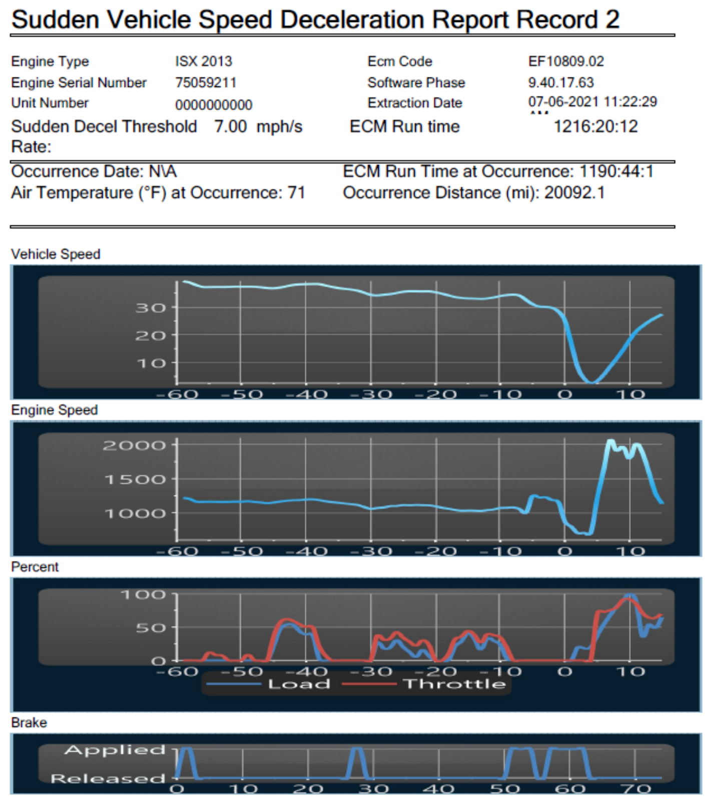
- Data that changes with time
- Mileage, hours, histogram data for aftertreatment, et al.

Diagnostics Data

- Diagnostic Messages including Failure Mode Indicators and Suspect Parameters (or subsystems)
- Freeze Frame Data

Event Data

- Recordings from Triggered Events like Last Stops, Hard Brake, External Triggers, or Fault Codes
- Useful for driver training and crash reconstruction



Sudden Vehicle Speed Deceleration Report Record 2

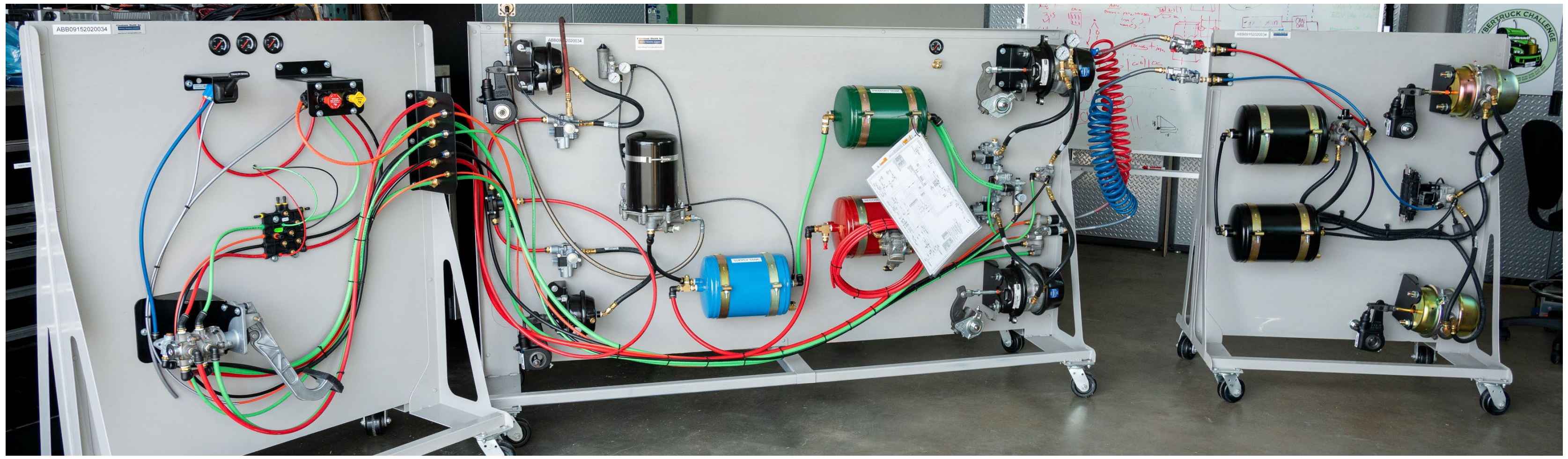
Engine Type	ISX 2013	Ecm Code	EF10809.02
Engine Serial Number	75059211	Software Phase	9.40.17.63
Unit Number	0000000000	Extraction Date	07-06-2021 11:22:29
Sudden Decel Threshold Rate:	7.00 mph/s	ECM Run time	1216:20:12

Occurrence Date: N\A | ECM Run Time at Occurrence: 1190:44:1
Air Temperature (°F) at Occurrence: 71 | Occurrence Distance (mi): 20092.1

Electronic Brake Controllers

1. Sense wheel speeds
2. Determine if wheel lock-up is impending
3. Modulate the air pressure to the brake chambers
4. Tell the engine to stop producing torque

store.partshighway.com

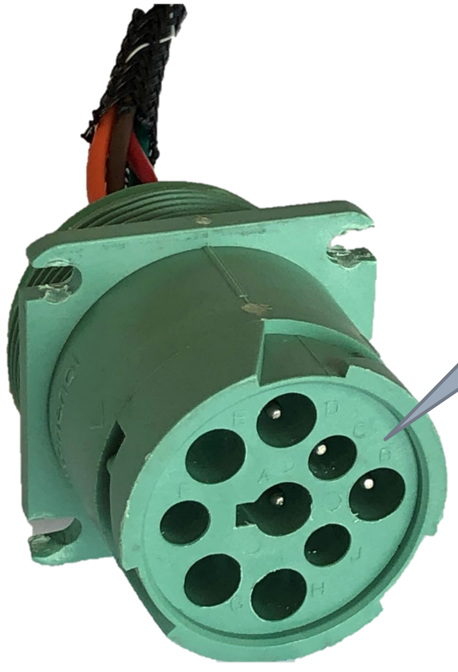


In-Vehicle Networking

HOW DOES THE BRAKE CONTROLLER COMMUNICATE WITH THE ENGINE CONTROLLER?

Controller Area Network (CAN) in Trucks

IN THE DIAGNOSTIC PORT

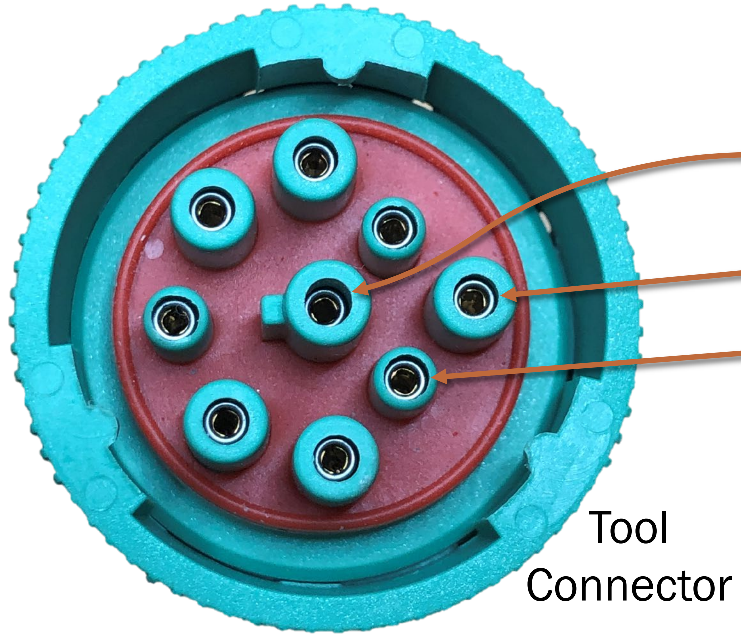


IN THE WIRING HARNESS

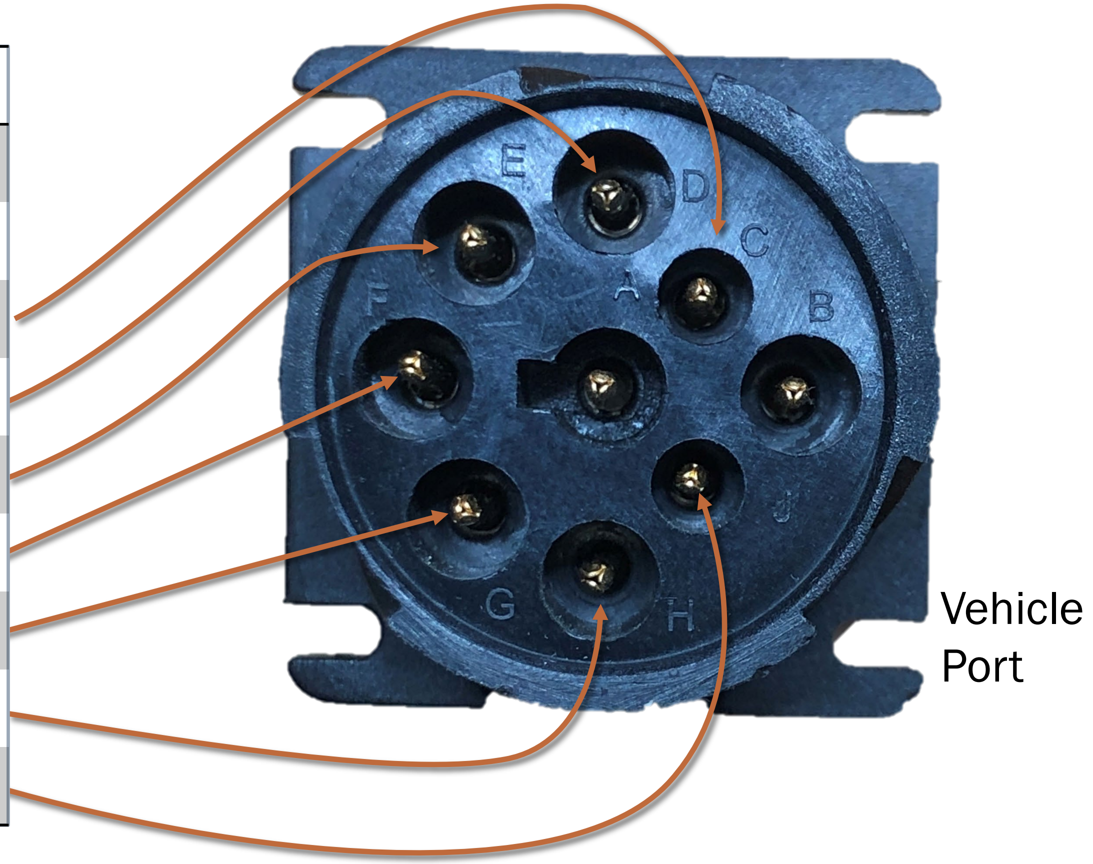
The CAN bus is typically a twisted-pair of copper wire connecting the ECUs.

SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

Diagnostic Connector Pinouts – SAE J1939



Pin	Signal
A	Ground
B	Vbatt (+12V)
C	J1939 High
D	J1939 Low
E	Shield
F	J1708 A(+)
G	J1708 B(-)
H	CAN2 High
J	CAN2 Low



250k - Black Connector
500k - Green Connector
Green goes into Black
Black cannot plug into Green

SAE J1708/J1587

A BRIEF INTERLUDE

Some History: J1708 and J1587

First mainstream heavy vehicle diagnostics communication protocol

Based on RS-485 communications at 9600 baud

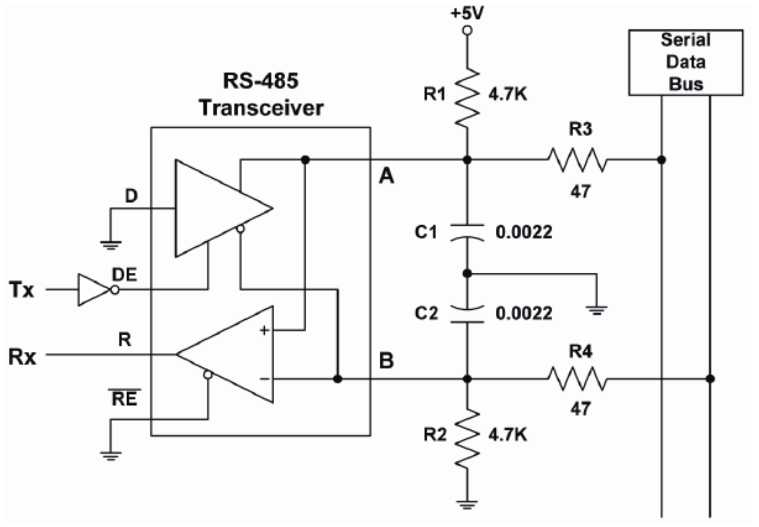
Message frames are determined by time spacing between bytes (this can be fragile)

8-bit check sum to verify message contents (not strong)

Physical wiring and signaling are defined in J1708

Meaning of the messages are defined in J1587

Message Composition:
MID = Message Identifier (like Source Address)
PID = Parameter Identifier (like SPN)
Data = encoded value of PID
Checksum = 1 byte to check message transmission errors



SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

J1708 Message Example

J1708 DATA FRAMES

```
80 FF B7 00 00 CA
80 54 1B BE C0 11 28 CF 5B 7F 5C 58 FD
80 54 1C BE 08 12 5B 80 5C 59 2B C7 B6
88 31 00 97 3F FF 54 1C A8 18 01 D1 01 FF 70
8C 41 F3 46 00 A8 17 01 3A
80 55 40 79 00 B7 85 01 B8 70 02 F4 04 2E 9A 56 00 F5
80 54 1C BE 3C 12 28 CF 5B 80 5C 59 7D
80 54 1C BE 52 12 5B 83 5C 5B 59
80 55 40 79 00 B7 96 01 B8 53 02 F5 04 2E 9A 56 00 00
80 54 1D BE B0 12 28 CF 5B 85 5C 5C 00
80 54 1D BE EC 12 5B 86 5C 5D 2C F0 9D
80 54 1D BE 0B 13 28 CF 29 C0 5B 89 5C 5F B4
```

J1708 MEANING

Message Identifiers are at the beginning

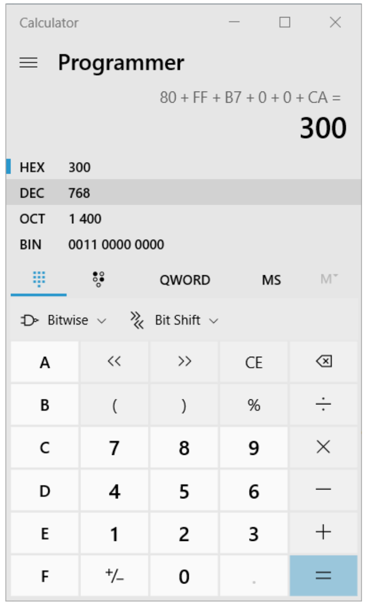
The message is considered valid if the 8-bit sum is zero.

- `80+FF+B7+00+00+CA = 300`

8-bit sums ignore carry

Popular MIDs

- Engine 1: 128 (0x80)
- Brakes, Power Unit: 136 (0x88)
- Brakes, Trailer #1: 137 (0x89)
- Instrument Cluster: 140 (0x8C)
- Cab Climate Control: 146 (0x92)
- Off-Board Diagnostics Tool: 172 (0xAC)

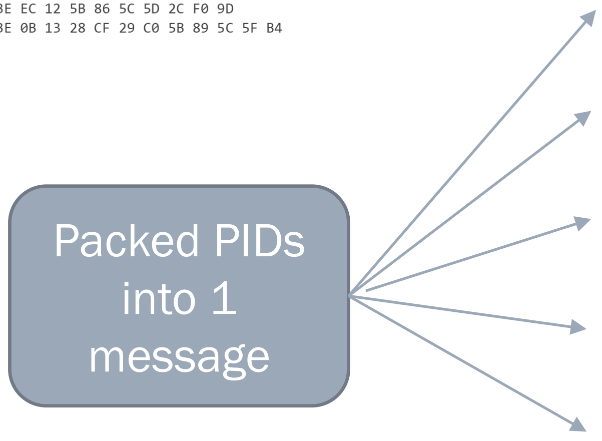

Calculator
Programmer
QWORD
MS
Bitwise
Bit Shift
A B C D E F « » CE % ( ) 7 8 9 4 5 6 1 2 3 0 +/- . = ÷ × − +

```
80 + FF + B7 + 0 + 0 + CA =
300
HEX 300
DEC 768
OCT 1 400
BIN 0011 0000 0000
```

J1587 Message Decoding Example

SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

```
80 FF B7 00 00 CA
80 54 1B BE C0 11 28 CF 5B 7F 5C 58 FD
80 54 1C BE 08 12 5B 80 5C 59 2B C7 B6
88 31 00 97 3F FF 54 1C A8 18 01 D1 01 FF 70
8C 41 F3 46 00 A8 17 01 3A
80 55 40 79 00 B7 85 01 B8 70 02 F4 04 2E 9A 56 00 F5
80 54 1C BE 3C 12 28 CF 5B 80 5C 59 7D
80 54 1C BE 52 12 5B 83 5C 5B 59
80 55 40 79 00 B7 96 01 B8 53 02 F5 04 2E 9A 56 00 00
80 54 1D BE B0 12 28 CF 5B 85 5C 5C 00
80 54 1D BE EC 12 5B 86 5C 5D 2C F0 9D
80 54 1D BE 0B 13 28 CF 29 C0 5B 89 5C 5F B4
```



```
80 54 1B BE C0 11 28 CF 5B 7F 5C 58 FD
```

- 80 - MID for Engine #1
- 54 – PID 84, Road Speed
- 1B – Value of Road Speed (27 * 0.5 = 13.5 mph)
- BE – PID 190, Engine Speed
- C0 11 – Value of Engine Speed (3089 * 0.25 = 772.25 rpm)
- 28 – PID 40, Engine Retarder Switches Status
- CF – (11001111) Engine Retarder switch Off
- 5B – PID 91, Percent Accelerator Pedal Position (APP)
- 7F – Value of Percent APP (127*0.4 = 50.8%)
- 5C – PID 9C, Percent Engine Load
- 58 – Value of Engine Load (88 * 0.5 = 44%)
- FD – Checksum (80+54+1B+BE+C0+11+28+CF+5B+7F+5C+58+FD = 0x600)

These interpretations are from the SAE J1587 Document: https://www.sae.org/standards/content/j1587_201301/

Power Line Carrier SAE J2497

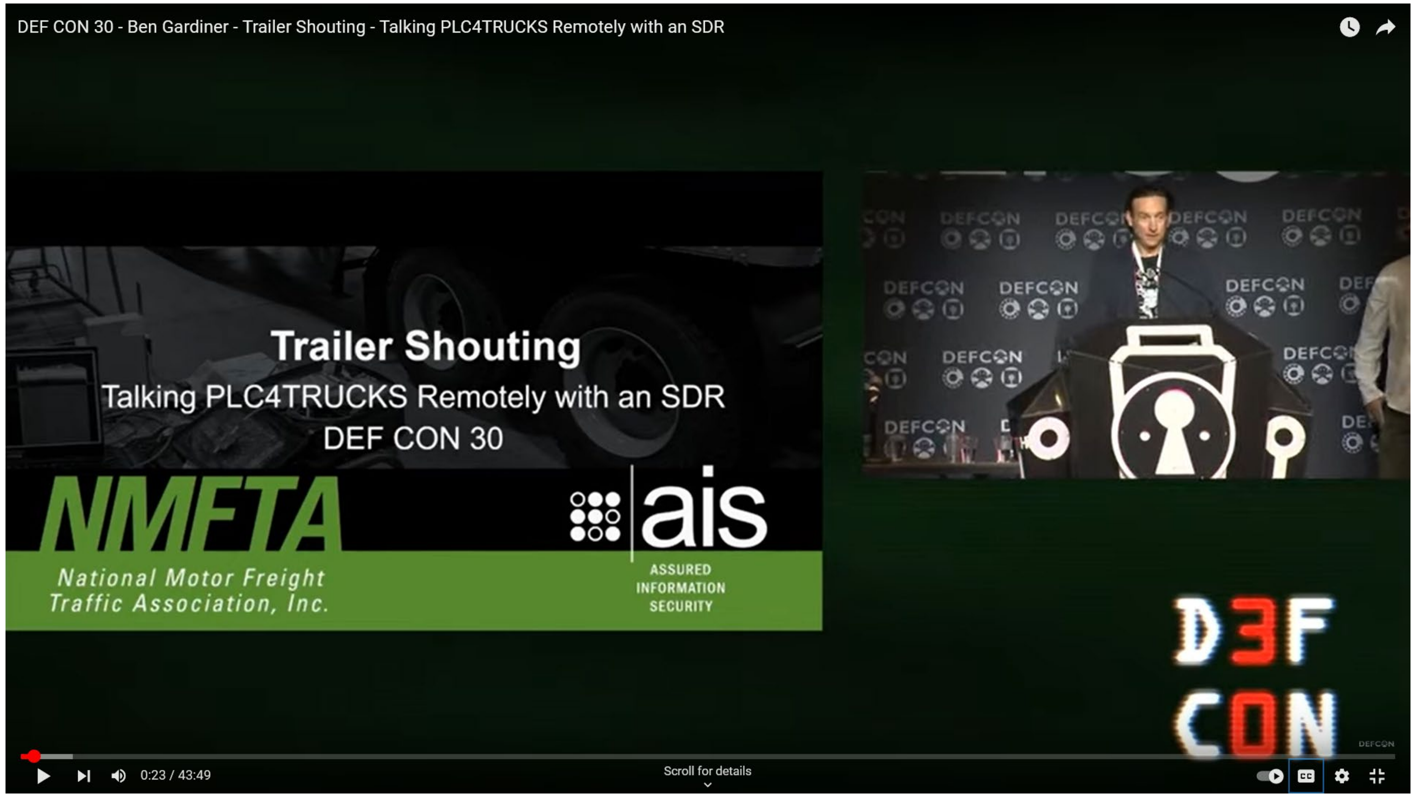
When trailers are connected, similar J1708/J1587 data is transmitted over the trailer power (+12V) line using frequency spectrum modulation.

Trailer Diagnostic Tools

Electronic Brake Controller

J560 Connectors

Why SAE J2497 is important

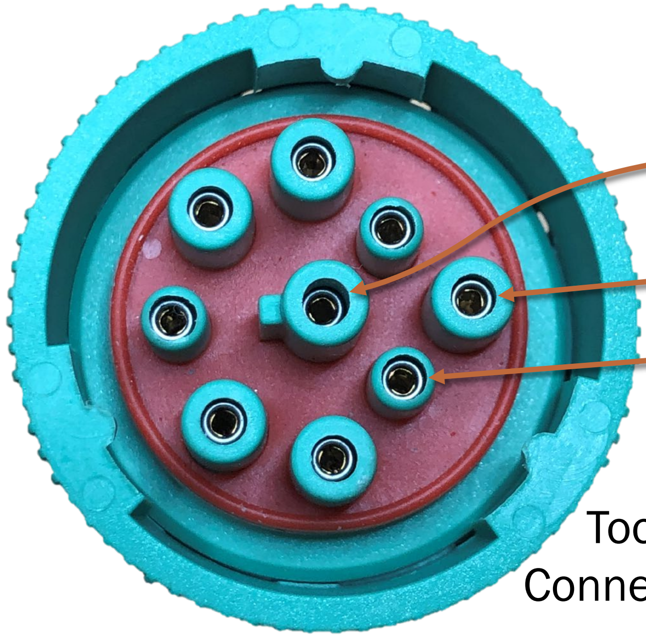

DEF CON 30 - Ben Gardiner - Trailer Shouting - Talking PLC4TRUCKS Remotely with an SDR
Trailer Shouting
Talking PLC4TRUCKS Remotely with an SDR
DEF CON 30
NMFTA
National Motor Freight Traffic Association, Inc.
ais
ASSURED INFORMATION SECURITY
D3F CON
0:23 / 43:49
Scroll for details

https://www.youtube.com/watch?v=Na8K_fVEzQo

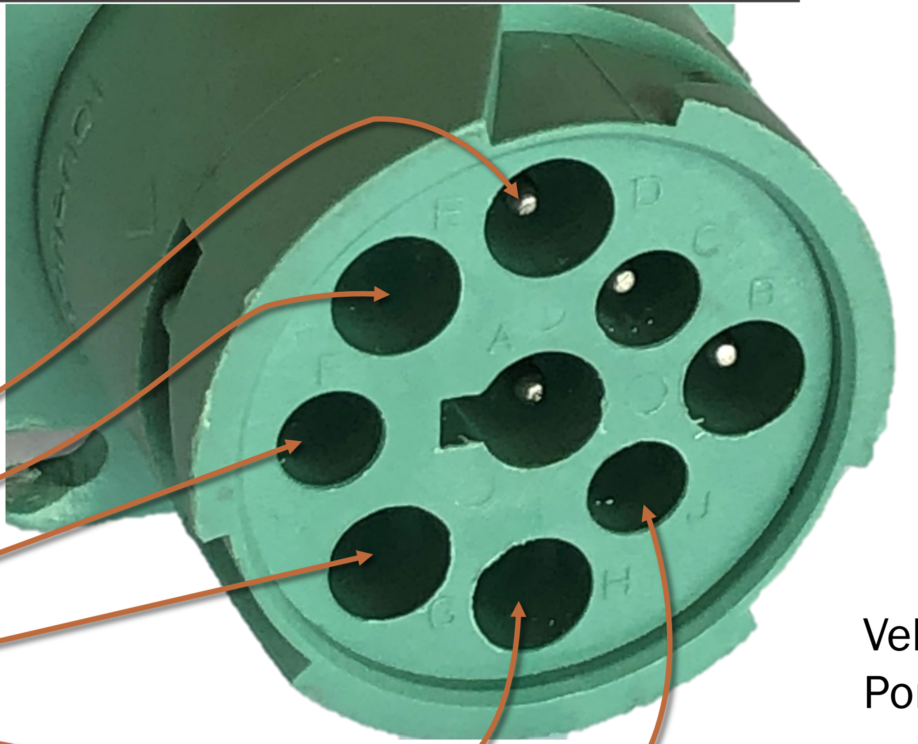
Back to J1939

DIAGNOSTIC CONNECTORS

Diagnostic Connector Pinouts – PACCAR



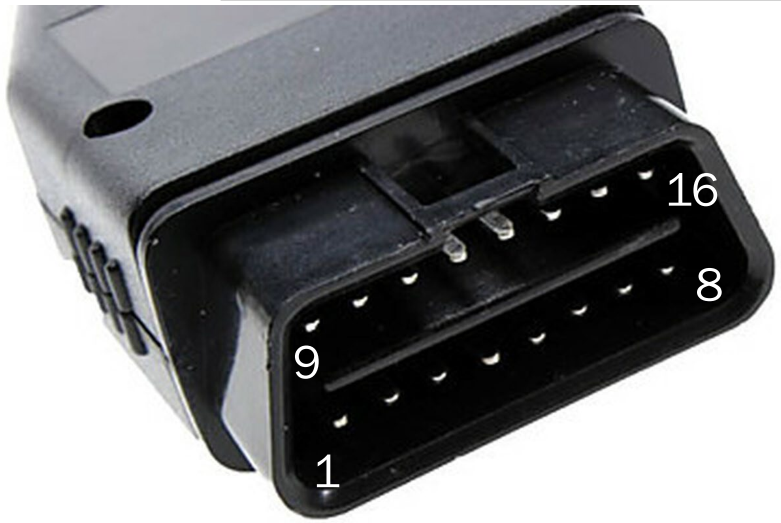
Pin	Signal
A	Ground
B	Vbatt (+12V)
C	J1939 High
D	J1939 Low
E	Shield
F	Diag CAN H
G	Diag CAN L
H	CAN3 High
J	CAN3 Low



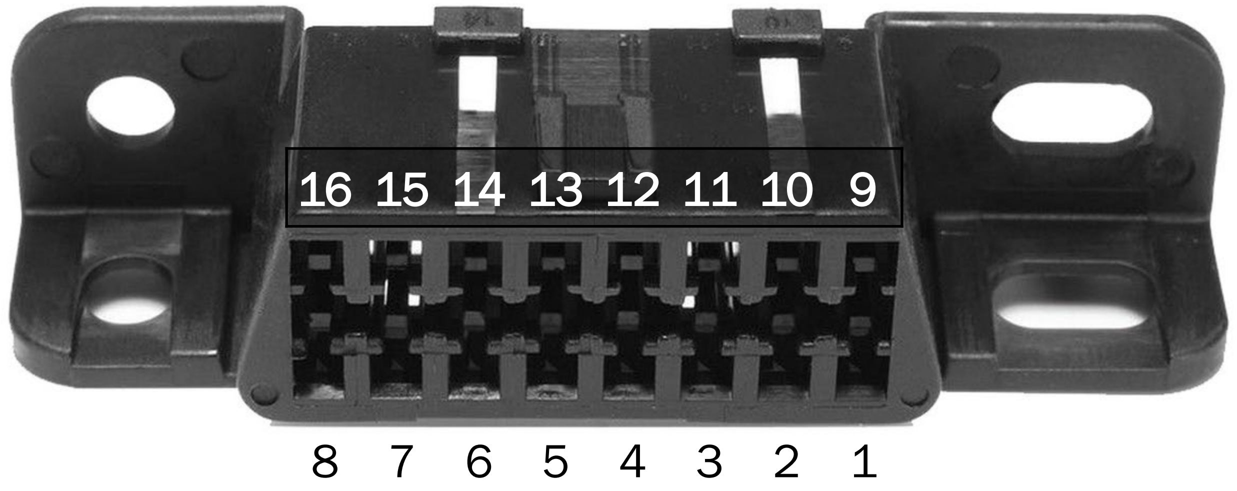
Diagnostics CAN is at 250k.

PACCAR Sells a crossover cable for diagnostics.

Diagnostic Connector Pinouts – Mack/Volvo



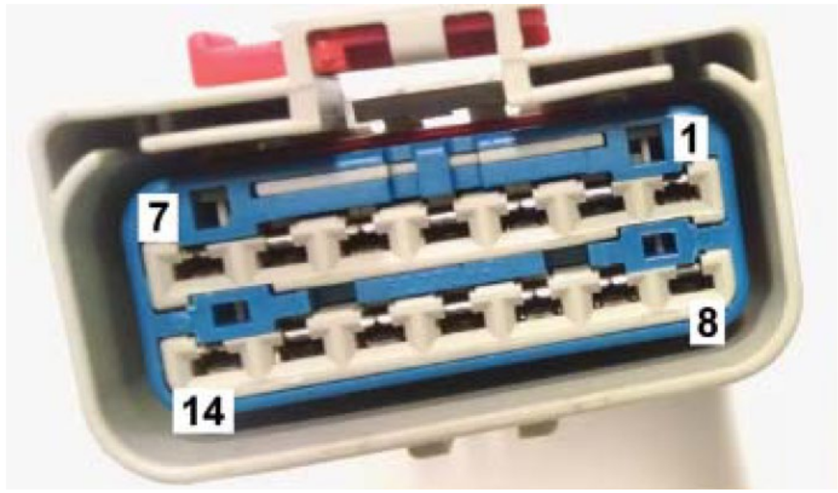
Pin	Signal
3	J1939 High
5	Ground
6	ISO 15765 L
11	J1939 Low
12	J1708 A(+)
13	J1708 B(-)
14	ISO 15765 L
16	VBatt



Source: DG Technologies Product Pinouts and Industry Connectors Reference Guide
https://dgtech.com/wp-content/uploads/2016/04/Pinouts_ICR.pdf

RP1226 Accessory Connector Pinouts

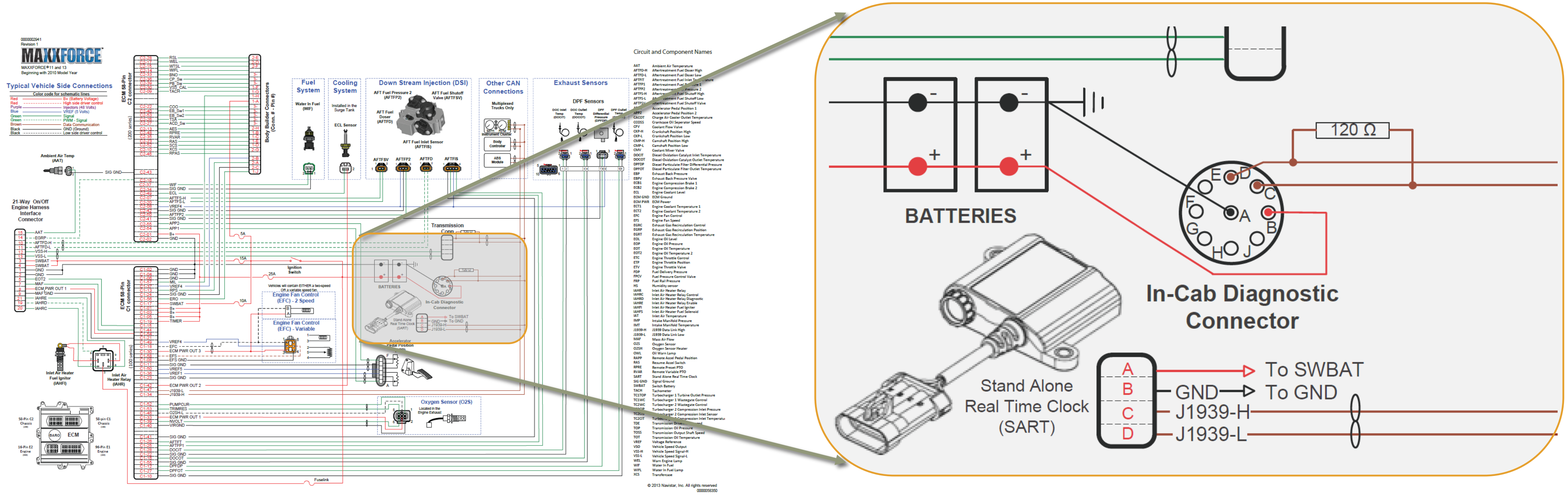
Pin	Signal	Pin	Signal
1	Switched Battery	8	Ground
2	CAN 1 High	9	CAN 1 Low
3	Reserved (not used)	10	Reserved (not used)
4	CAN 2 High	11	CAN 2 Low
5	OEM Reserved	12	OEM Reserved
6	J1708 A(+)	13	J1708 B(-)
7	Ignition (PLC)	14	Battery (always on)



Source: ATA TMC RP1226
https://www.atabusinesssolutions.com/Shopping/Product/viewproduct/2675472/undefined

Diagnostic Connector Pinouts – Key Point

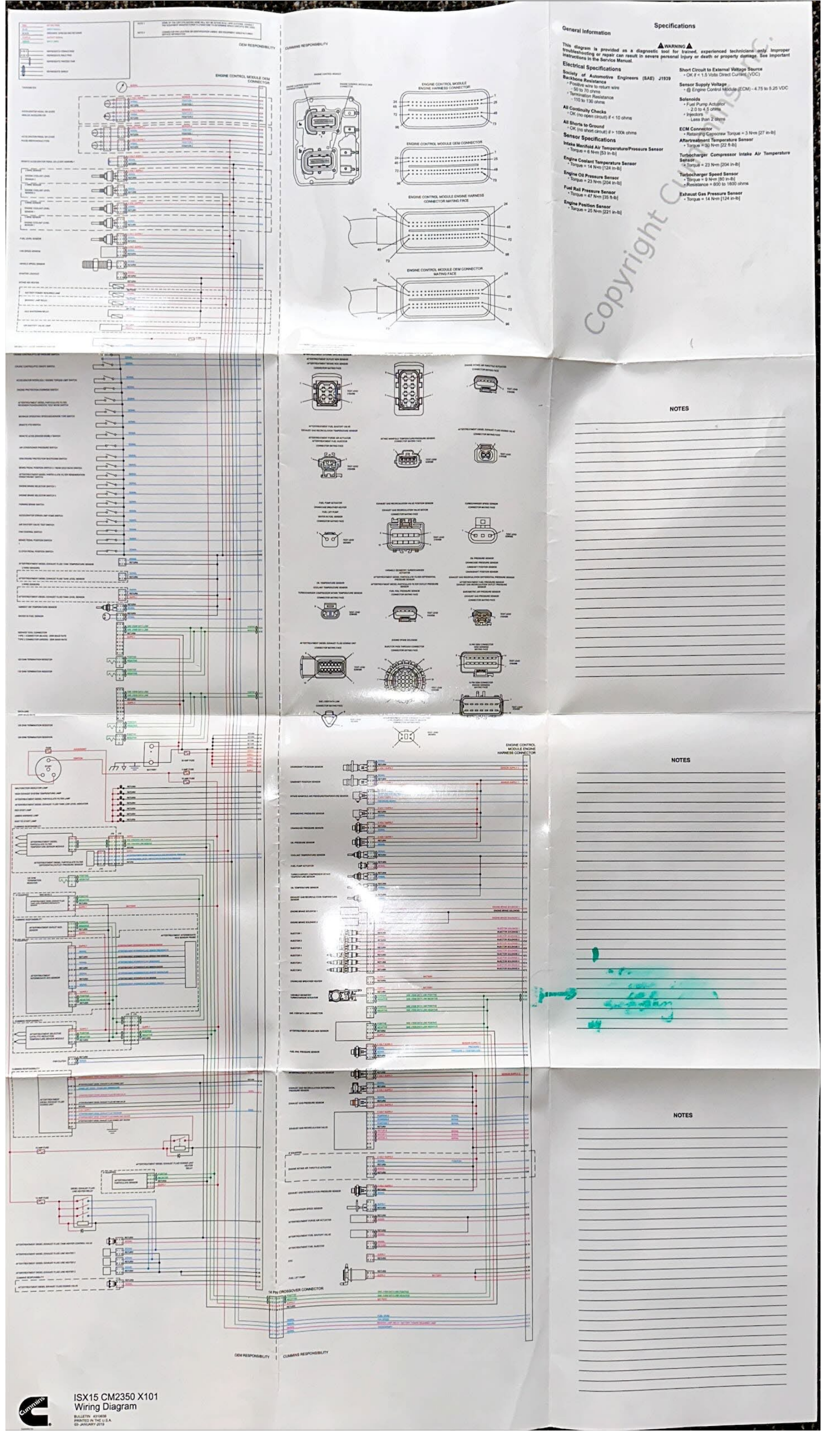
Use the wiring diagrams to verify the actual pinouts of the diagnostic connector.


MAXXFORCE
Typical Vehicle Side Connections
Fuel System
Cooling System
Down Stream Injection (DSI)
Other CAN Connections
Exhaust Sensors
Circuit and Component Names
BATTERIES
120 Ω
In-Cab Diagnostic Connector
Stand Alone Real Time Clock (SART)
A
B
C
D
GND
J1939-H
J1939-L
To SWBAT
To GND
© 2013 Navistar, Inc. All rights reserved

In Class Exercise: Reading Schematics

Identify the pin numbers for the J1939 Service Tool Connector for the Cummins CM2350.

1. What pin is J1939 Positive on the CM2350?
2. What pin is J1939 Negative on the CM2350?
3. What pin is J1939 Positive on the Service Tool Connector?
4. What pin is J1939 Negative on the Service Tool Connector?
5. How many CAN channels are on the CM2350?



SOLUTIONS

In Class Exercise: Reading Schematics

Identify the pin numbers for the J1939 Service Tool Connector for the Cummins CM2350.

1. What pin is J1939 Positive on the CM2350? 22
2. What pin is J1939 Negative on the CM2350? 46
3. What pin is J1939 Positive on the Service Tool Connector? C
4. What pin is J1939 Negative on the Service Tool Connector? D
5. How many CAN channels are on the CM2350? 3 on the Schematic, 4 on the harness.

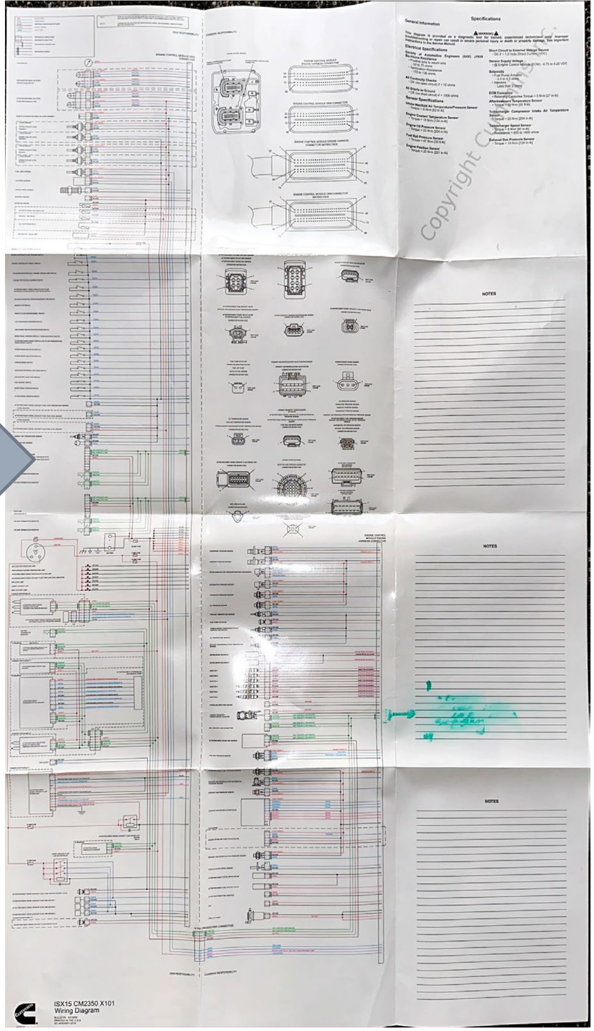
Notes:

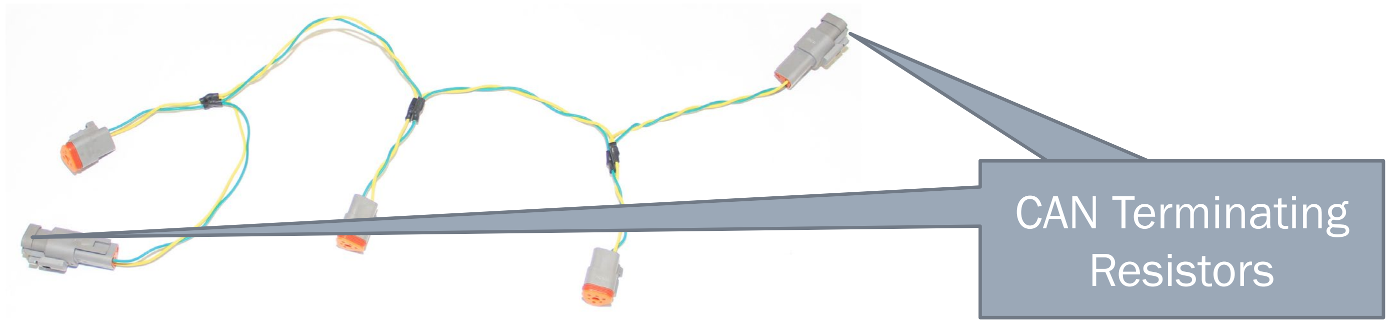
1. There are terminating resistors on each CAN/J1939 Channel.

2. The Vehicle OEM connector is brown on the CM2350,

3. The Engine connector is gray.

4. The engine connector has a J1939 channel for smart devices
   - Variable Geometry Turbo
   - Aftertreatment NOX sensor
   - Selective Catalytic Reduction (SCR) temperature sensors





Bill of Materials to Build a J1939 Diagnostic Connector

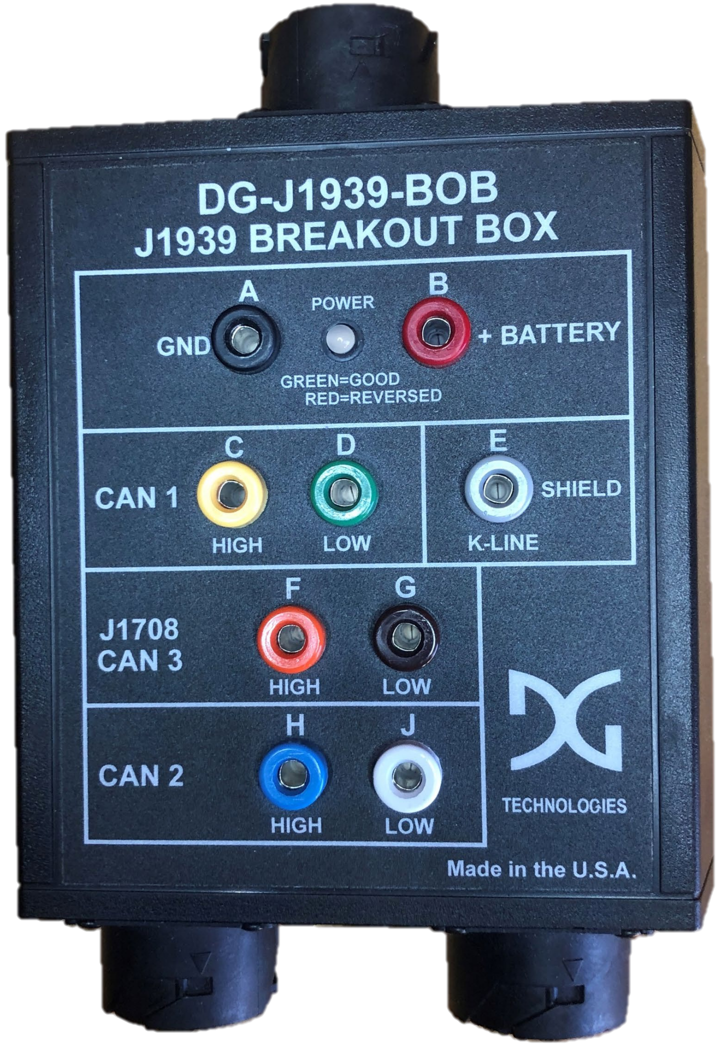
Qty.	Manufacturer	Part Number	Supplier	Supplier Part Num	Description
1	Amphenol Sine Systems	AHD16-9-1939S8R	Digi-Key	889-2245-ND	Green Plug Housing
4	Amphenol Sine Systems	AT62-201-16141	Digi-Key	889-1469-ND	Nickle Socket Contact
	Allied Wire and Cable			GXL-18 YELLOW	J1939 H Wire
				GXL-18 GREEN	J1939 L Wire
				GXL-18 RED	VBatt Wire
				GXL-18 BLACK	Ground Wire

Detailed Deutsch Connector Systems are described in the TE catalog:
https://www.te.com/ict-catalog#page=1

After Class Exercise

Design a J1939 breakout box with the following features:

1. Split the signals coming in to two signals going out
2. Connect test points to each signal



If you don't have time to build one yourself, they are commercially available. For example: https://www.dgtech.com/product/j1939-breakout-box/

Connecting to the CAN Bus

BOOT YOUR COMPUTERS TO UBUNTU LINUX

Controller Area Network

Introduced by Bosch in the 1980s

Multi-master priority-based bus access with non-destructive message arbitration

Utilizes a 15-bit cyclic redundancy check (CRC) to reliably detect transmission errors

Reliable delivery is built in with an acknowledgement bit at the end of the frame

Low latency with up to 8 bytes of data per frame (Classic CAN)

Bit rates up to 1mbit/second

Required on all passenger cars for emission compliance starting in 2008 (Standard 11-bit CAN ID)

Utilized by SAE J1939 as the foundational networking protocol in the 1990s

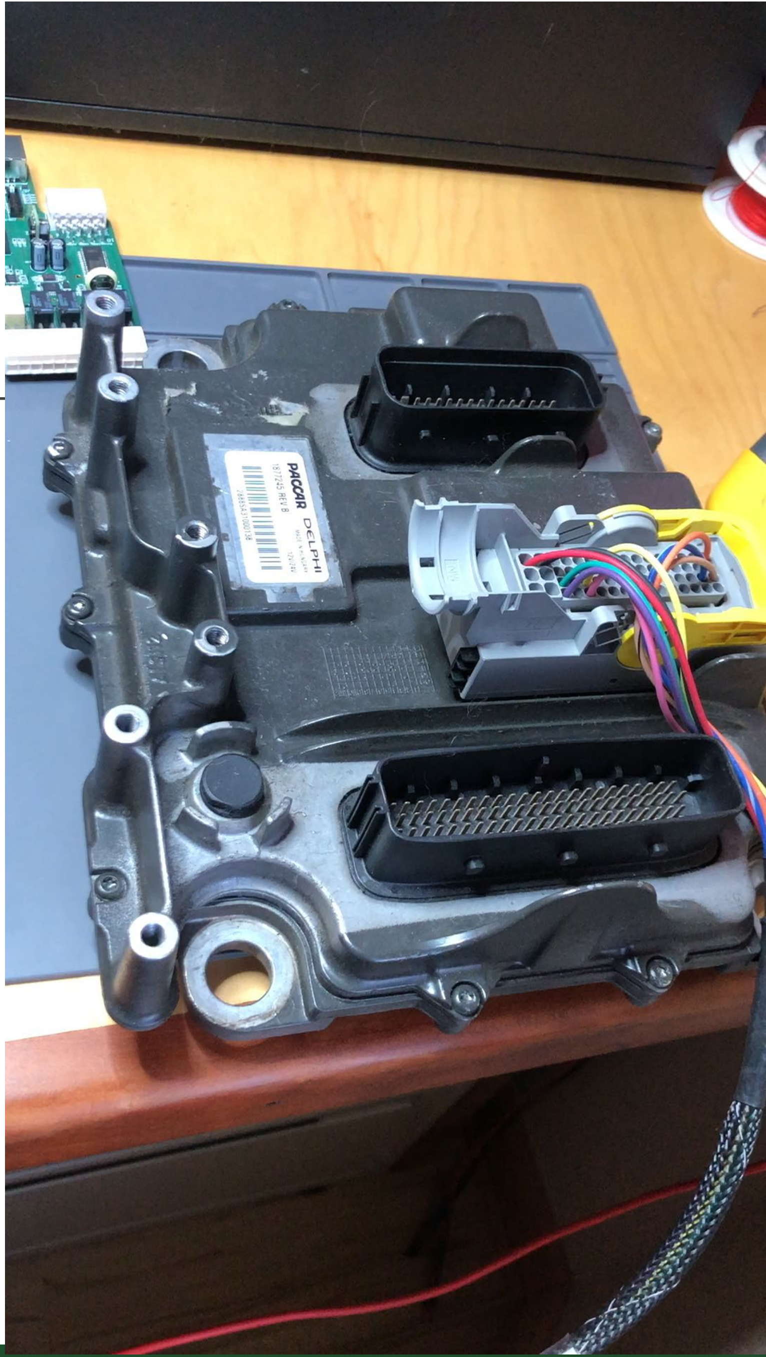
Extended 29-bit CAN Frame

SOF	11-bit CAN ID	SRR	IDE	18-bit CAN ID	RTR	Control Field	Data Bytes	CRC	ACK	EOF

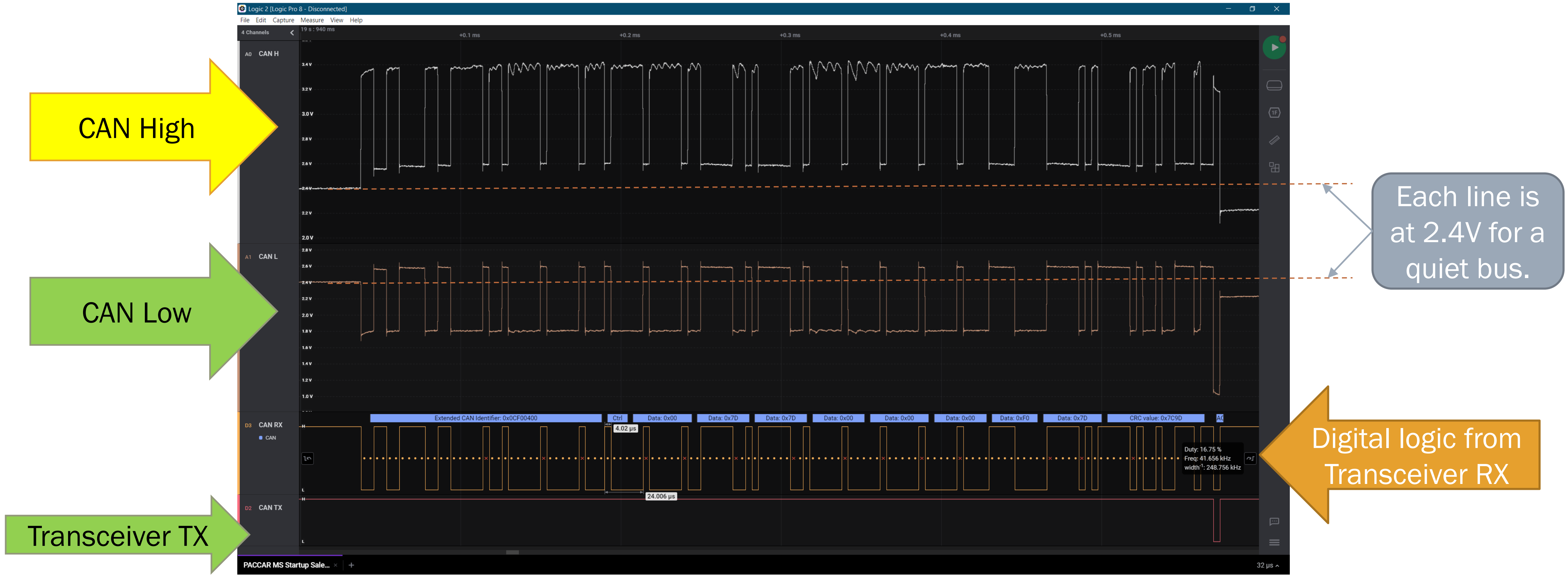
CAN Signaling: Measurement Example

- PACCAR MX Engine Control Module (ECM)
- Synercon Technologies Smart Sensor Simulator
  - Completes the CAN network circuit
  - Provides connectivity for the ECM
- DG Technologies J1939 Breakout Box
- Raspberry Pi with a CAN-FD Hat
  - Runs embedded Linux with SocketCAN
  - Records CAN traffic using can-utils `candump` command
- Fluke Scope Meter as an Oscilloscope
  - Measures voltage traces between CAN High and CAN Low
- Saleae Logic Probe
  - Analog Voltage measurements (duplicating the oscilloscope)
  - Digital measurements from the CAN Transceiver
  - CAN signal decoding features
  - PC application interface

What is on the wire? Let's monitor the yellow CAN-H and green CAN-L lines.



CAN Signaling: Single Frame



CAN Measurements Observations

A bit time is about 4 microseconds. This is 1/250000.

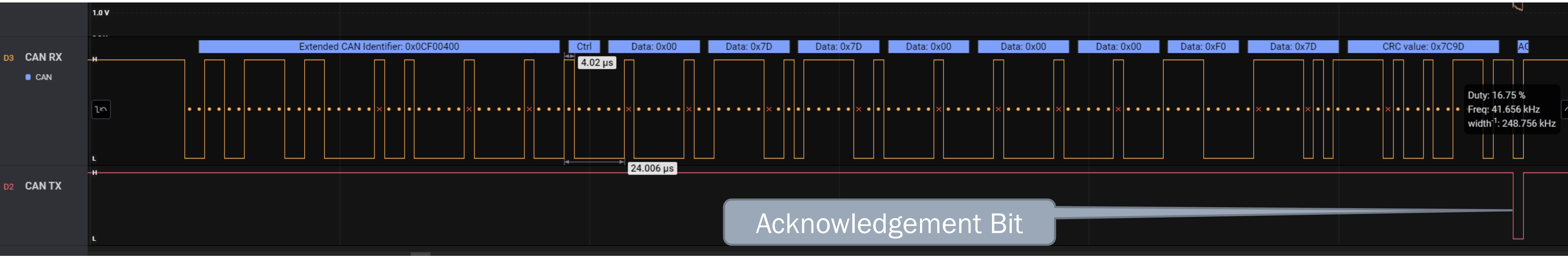
Data that has all zeros still has extra bits in the field. These are called stuff bits.

Stuff bits are inserted after 5 sequential bits of the same value.

At the end of the message, A bit is seen on the TX line, which indicates an acknowledgement the message was received.

The total message length is about 500us.

Signaling is non-return-to-zero (NRZ).



After Class Exercise

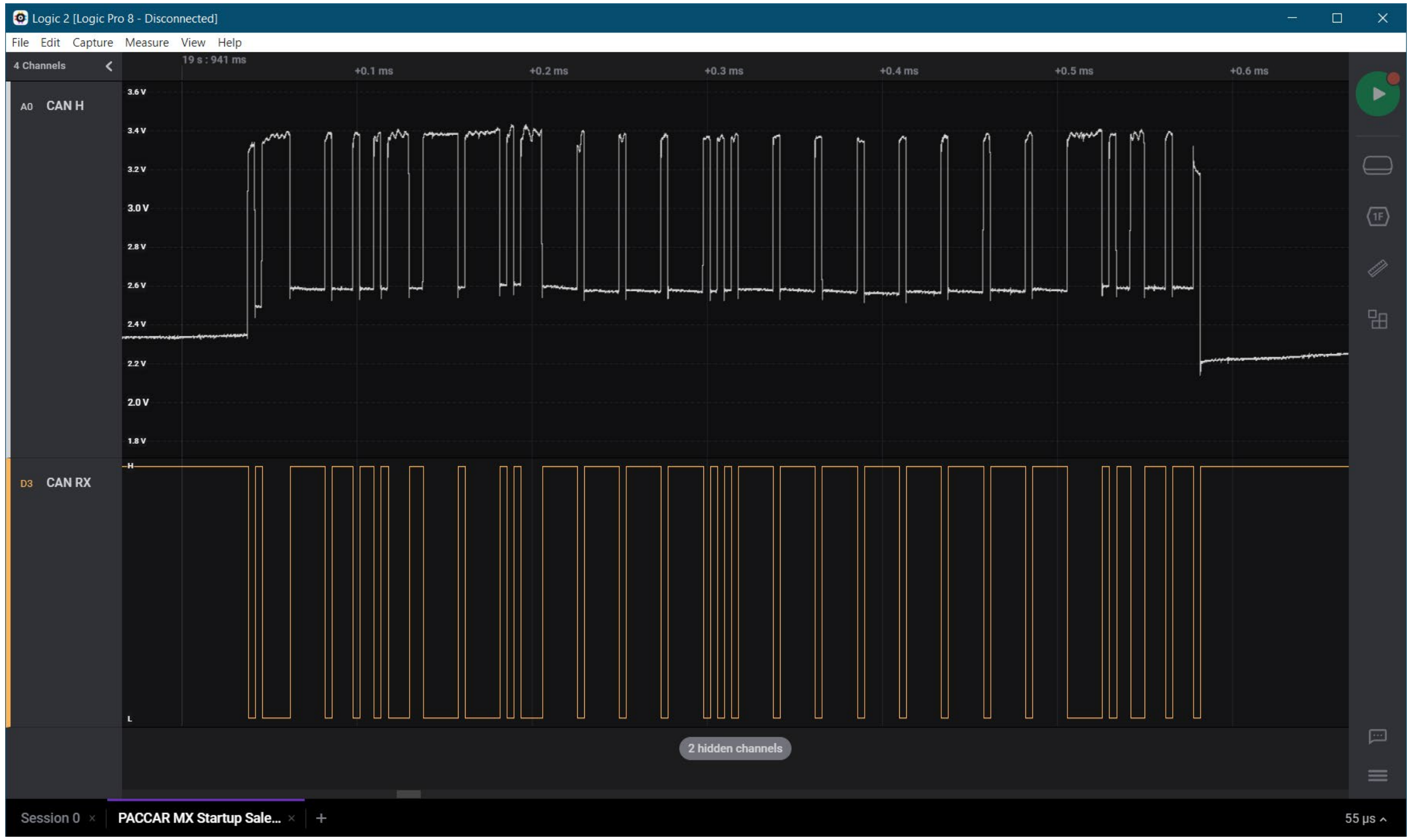
Determine the J1939 priority, parameter group number, destination address, and source address based on the trace of a CAN message.

There are 2 signals available:

1. Analog with 50MHz sampling
2. Digital with a time history of transitions.

Hints:

- There are stuff bits that need to be removed
- The CAN bus speed is 250k bits/second

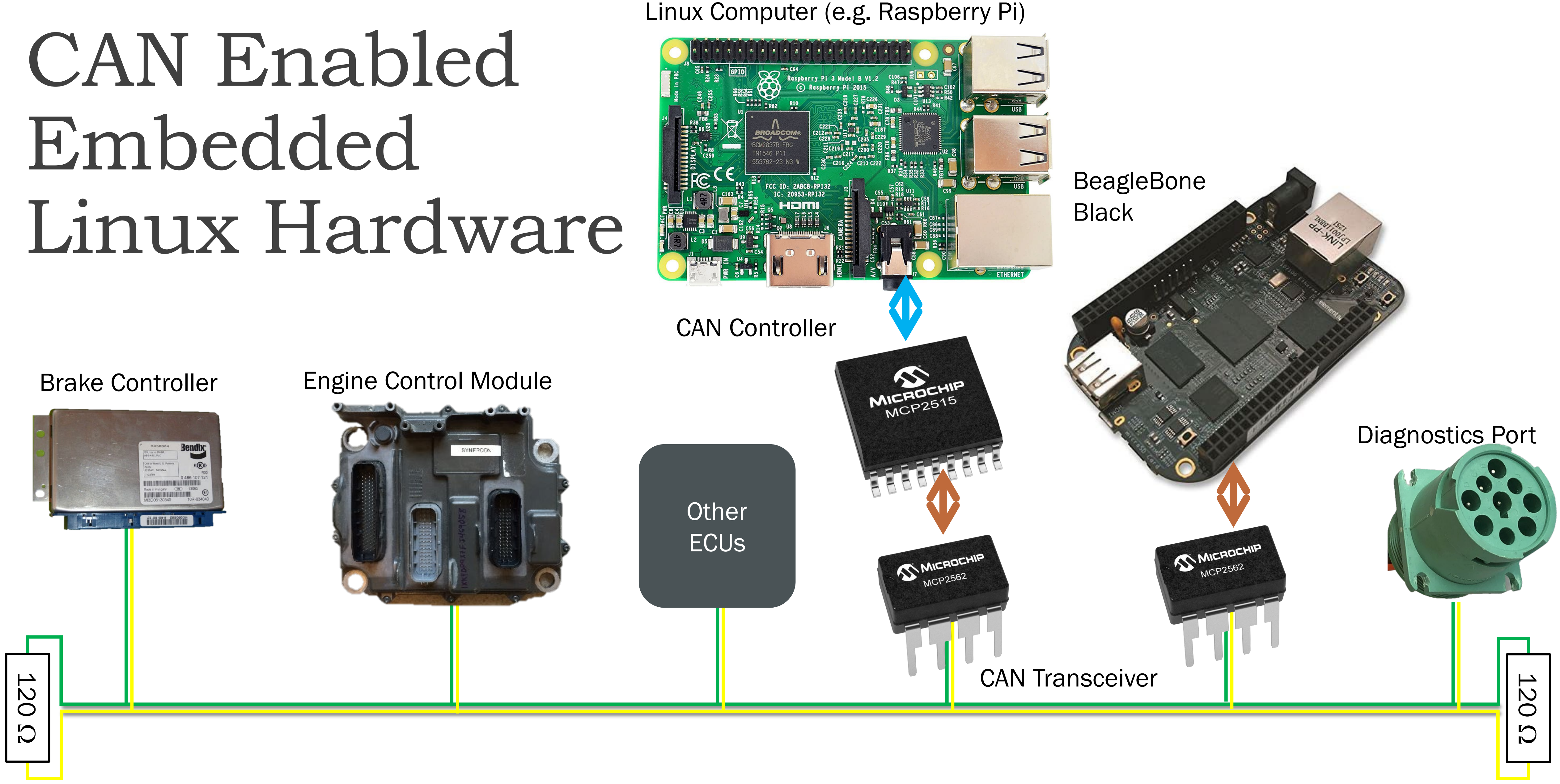


Download the traces from https://www.engr.colostate.edu/~jdaily/cyber/challenge_data.html

Collecting CAN Data

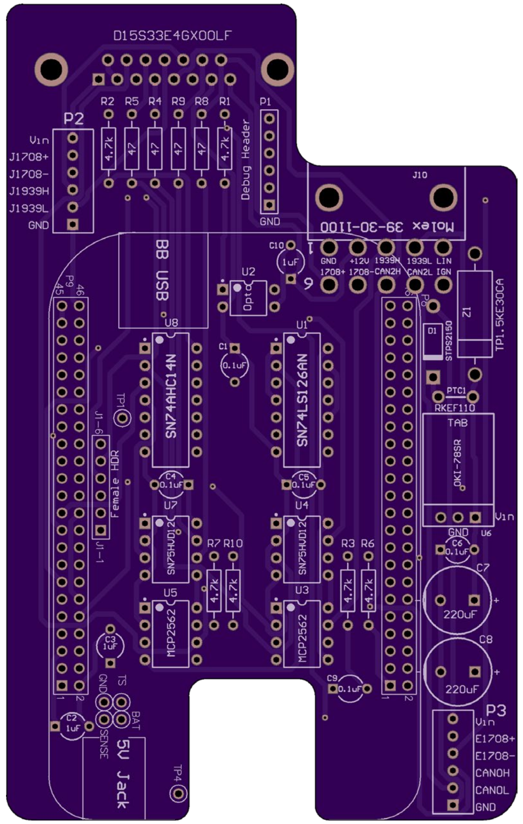
LET'S GET SOME DATA FROM A REAL ELECTRONIC CONTROL UNIT

CAN Enabled Embedded Linux Hardware



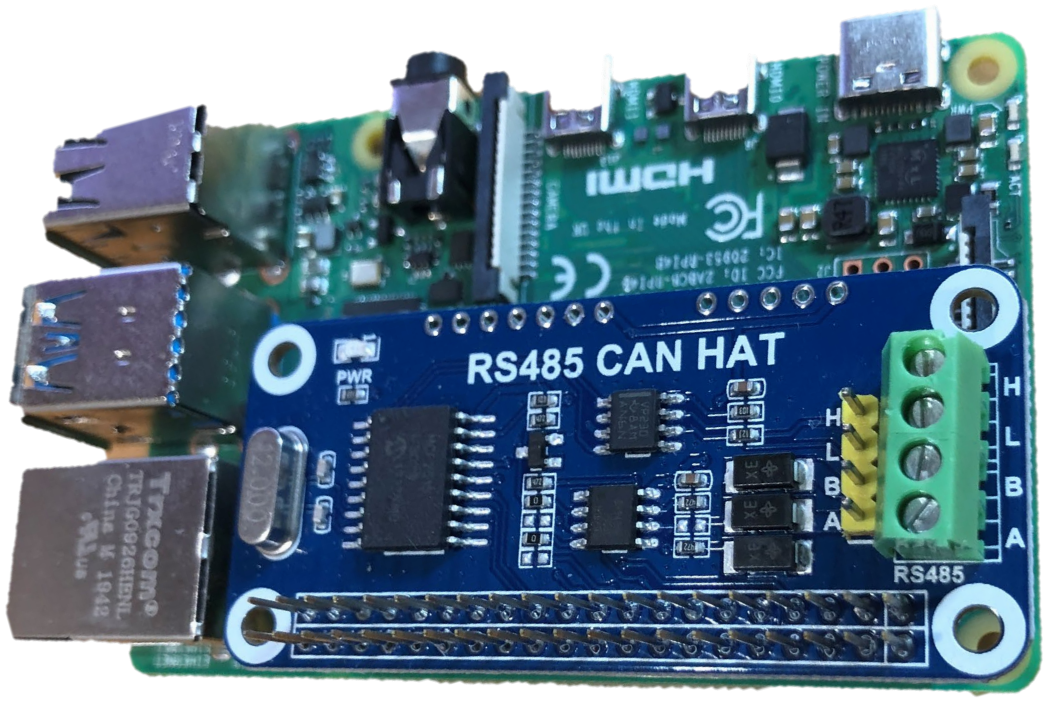
BeagleBone Black with Heavy Truck Cape

https://github.com/SystemsCyber/TruckCapeProjects/tree/master/hardware



https://oshpark.com/shared_projects/FXh7K628

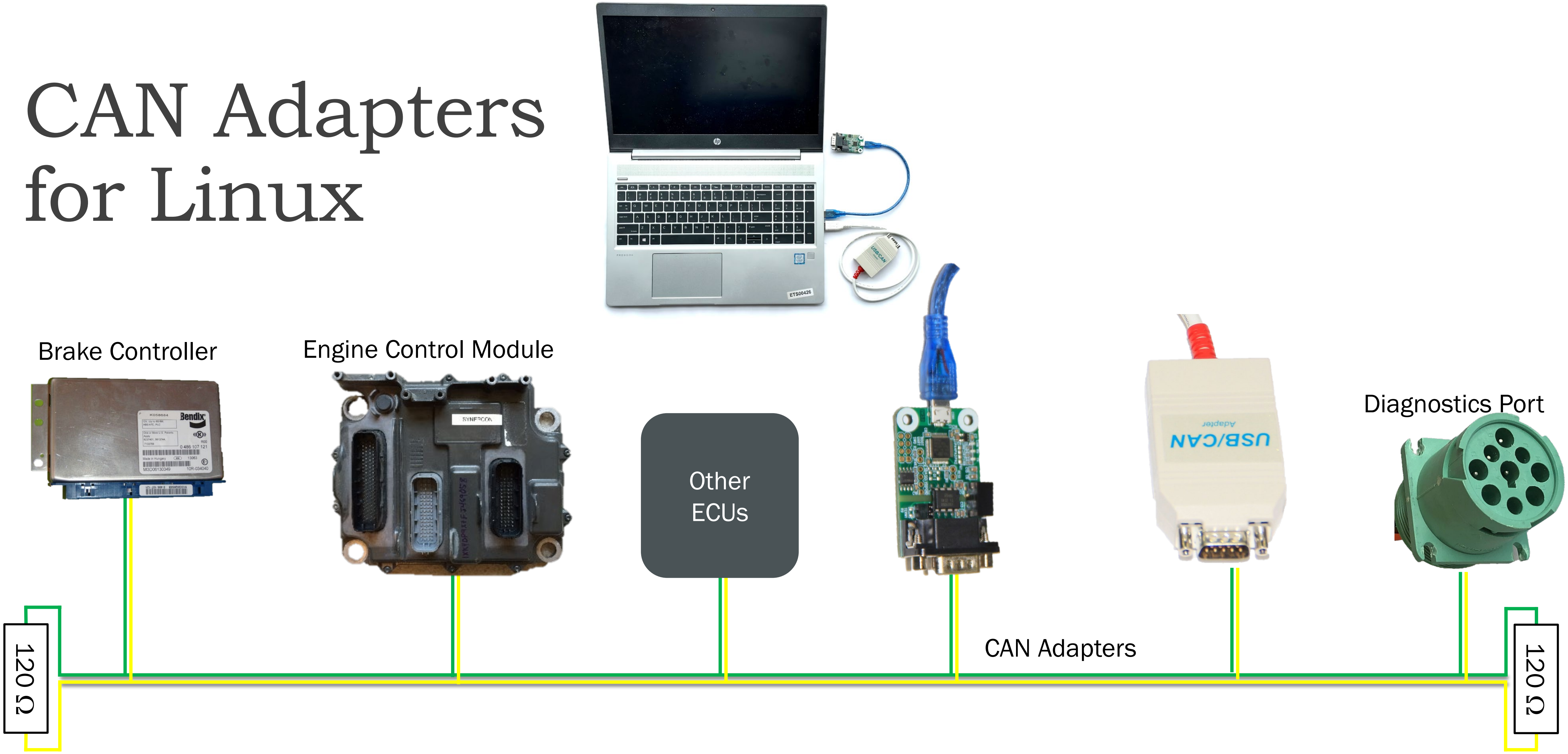
Raspberry Pi with CAN Hat



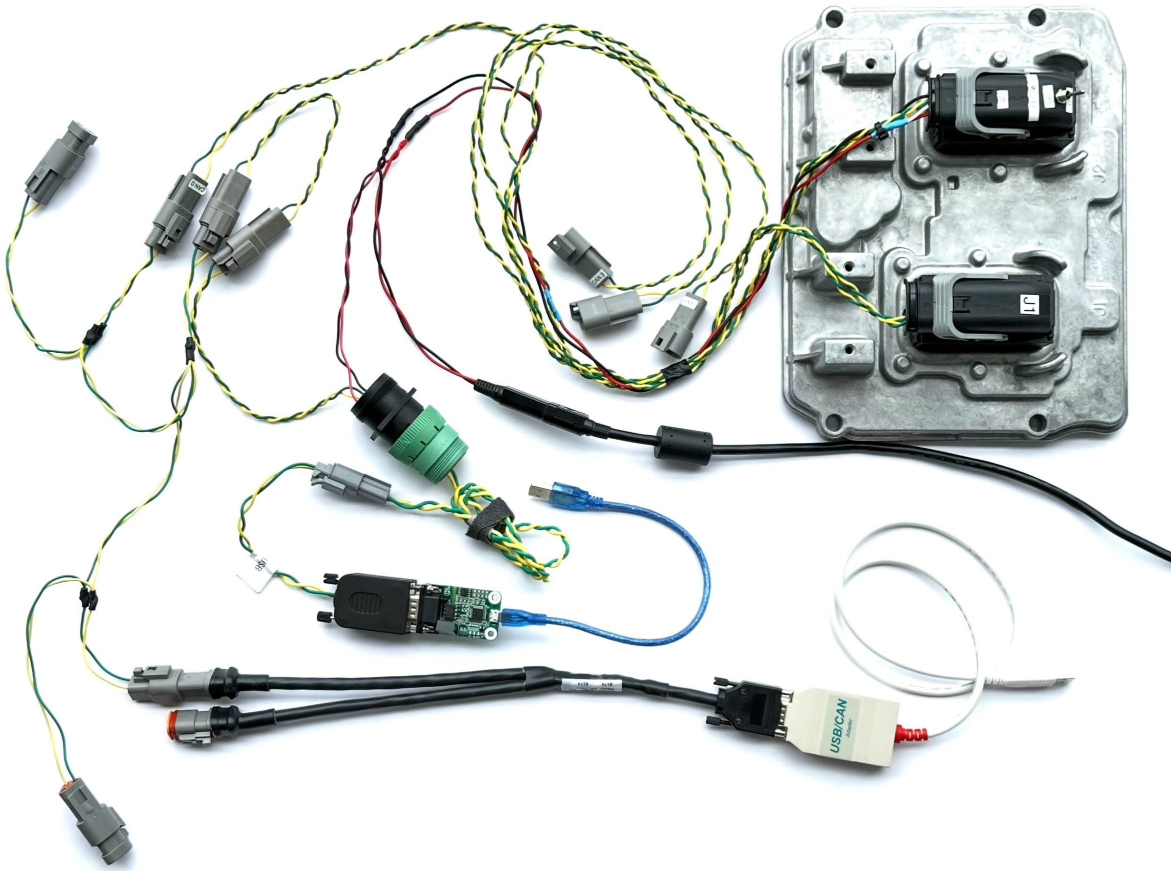
https://www.amazon.com/RS485-CAN-HAT-Long-Distance-Communication/dp/B07VMB1ZKH/

https://www.amazon.com/Raspberry-Pi-MS-004-00000024-Model-Board/dp/B01LPLPBS8/

CAN Adapters for Linux



Exercise: Connect the CAN Adapters for Linux



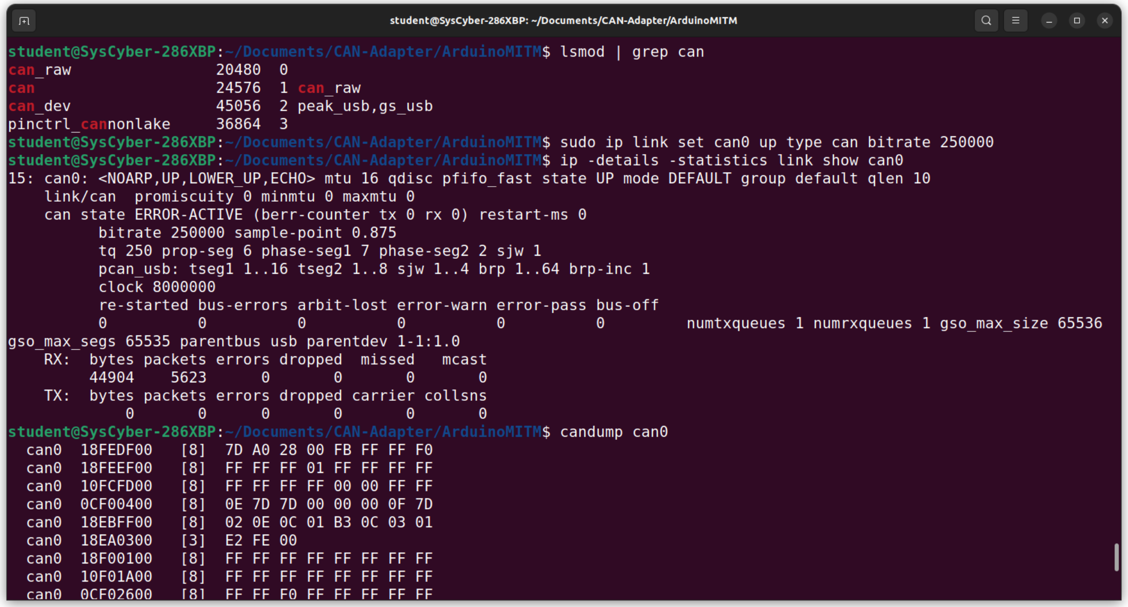
Exercise: Connect with SocketCAN on Linux

Plug in the adapter (Peak or USB2CAN)

In Ubuntu Linux, Open a Terminal and run these commands:

- `lsmod | grep can`
  - `Confirm the CAN kernel modules are installed`
- `sudo ip link set can0 up type can bitrate 250000`
  - Password is `student`
- `ip -details -statistics show can0`
- `candump any`
  - Ctrl-c to stop
- `sudo ip link set can0 down`

Make sure the keyswitch is on, cables are plugged in, and power is on.



student@SysCyber-286XBP: ~/Documents/CAN-Adapter/ArduinoMITM

```
student@SysCyber-286XBP:~/Documents/CAN-Adapter/ArduinoMITM$ lsmod | grep can
can_raw                20480  0
can                    24576  1 can_raw
can_dev                45056  2 peak_usb,gs_usb
pinctrl_cannonlake     36864  3
student@SysCyber-286XBP:~/Documents/CAN-Adapter/ArduinoMITM$ sudo ip link set can0 up type can bitrate 250000
student@SysCyber-286XBP:~/Documents/CAN-Adapter/ArduinoMITM$ ip -details -statistics link show can0
15: can0: <NOARP,UP,LOWER_UP,ECHO> mtu 16 qdisc pfifo_fast state UP mode DEFAULT group default qlen 10
    link/can  promiscuity 0 minmtu 0 maxmtu 0
    can state ERROR-ACTIVE (berr-counter tx 0 rx 0) restart-ms 0
          bitrate 250000 sample-point 0.875
          tq 250 prop-seg 6 phase-seg1 7 phase-seg2 2 sjw 1
          pcan_usb: tseg1 1..16 tseg2 1..8 sjw 1..4 brp 1..64 brp-inc 1
          clock 8000000
          re-started bus-errors arbit-lost error-warn error-pass bus-off
          0          0          0          0          0          0         numtxqueues 1 numrxqueues 1 gso_max_size 65536
gso_max_segs 65535 parentbus usb parentdev 1-1:1.0
    RX:  bytes packets errors dropped  missed   mcast
         44904    5623      0       0       0       0
    TX:  bytes packets errors dropped carrier collsns
             0       0      0       0       0       0
student@SysCyber-286XBP:~/Documents/CAN-Adapter/ArduinoMITM$ candump can0
  can0  18FEDF00   [8]  7D A0 28 00 FB FF FF F0
  can0  18FEEF00   [8]  FF FF FF 01 FF FF FF FF
  can0  10FCFD00   [8]  FF FF FF FF 00 00 FF FF
  can0  0CF00400   [8]  0E 7D 7D 00 00 00 0F 7D
  can0  18EBFF00   [8]  02 0E 0C 01 B3 0C 03 01
  can0  18EA0300   [3]  E2 FE 00
  can0  18F00100   [8]  FF FF FF FF FF FF FF FF
  can0  10F01A00   [8]  FF FF FF FF FF FF FF FF
  can0  0CF02600   [8]  FF FF F0 FF FF FF FF FF
```

Connecting to a Truck and Reading Data

Check the bitrate of the physical channel:

```
ip -details -statistics link show can0
```

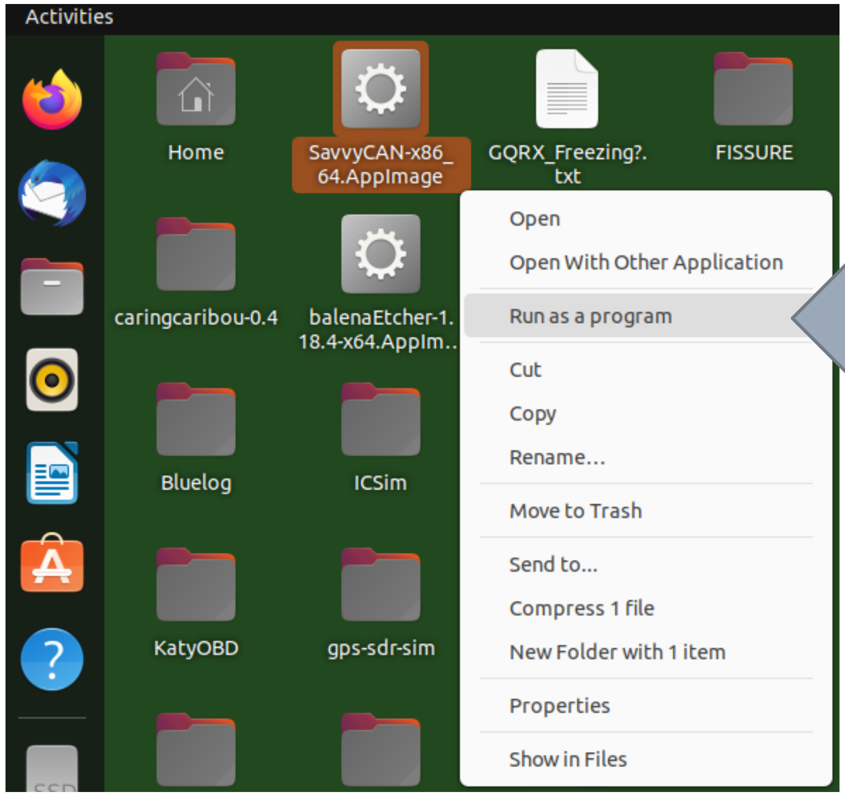
Change bitrate to match system:

```
sudo ip link set cano down
sudo ip link set can0 up type can bitrate 250000
```

Log the CAN data to a file:

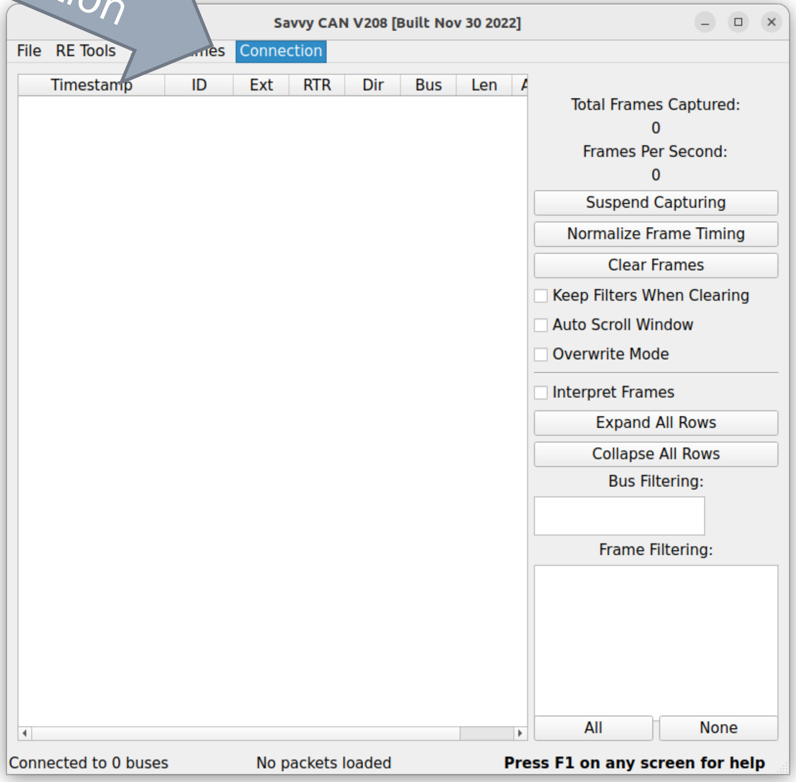
```
candump -l -e any
```

Exercise: SavvyCAN (Ubuntu)

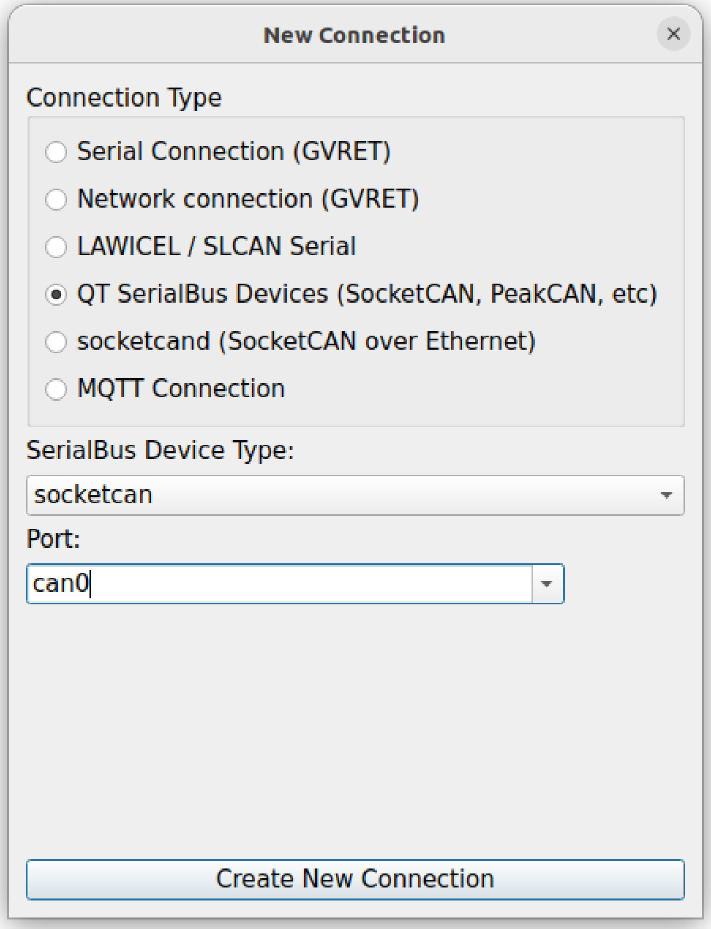


Right Click

Select Connection



Exercise: SavvyCAN (Ubuntu), cont.



New Connection

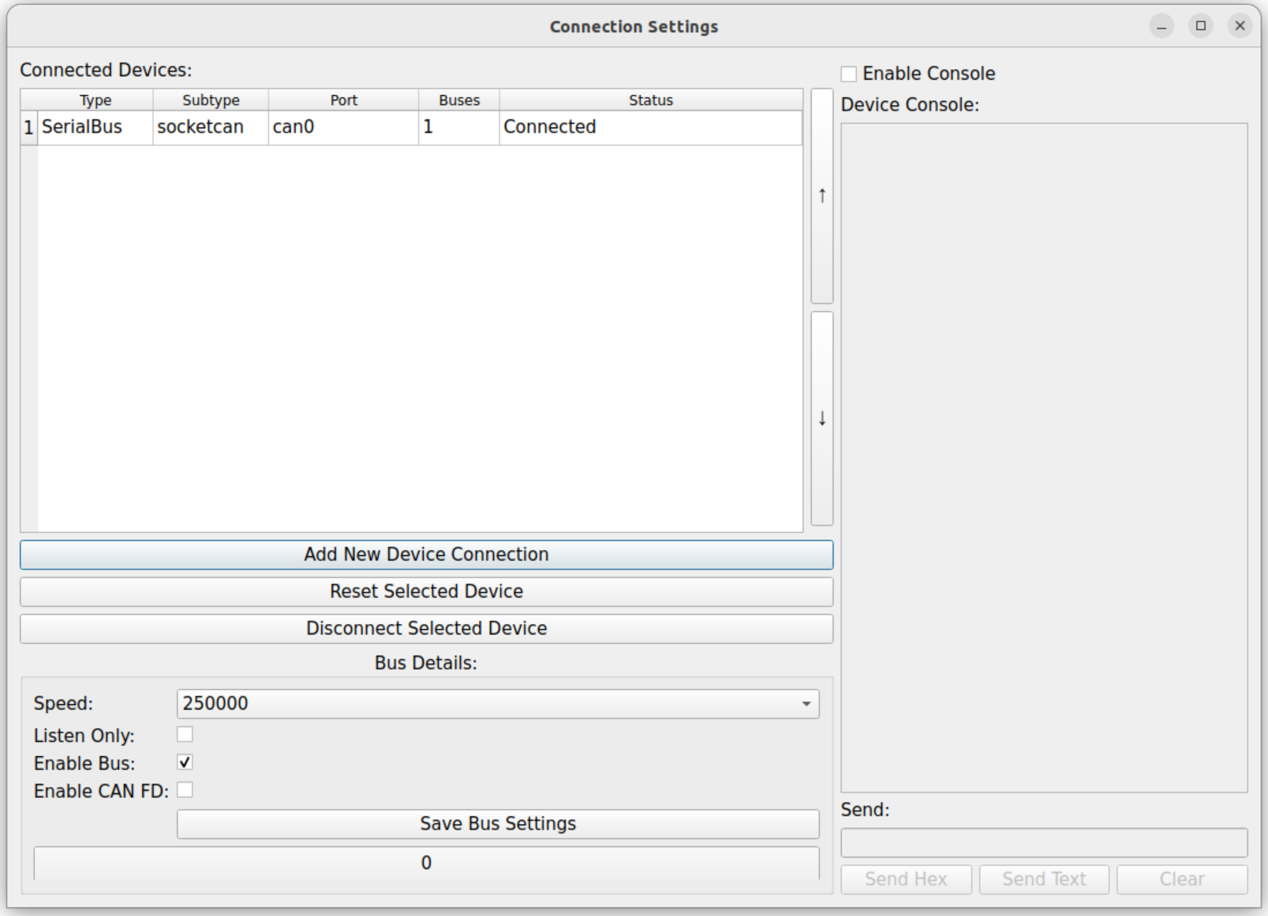
Connection Type

- Serial Connection (GVRET)
- Network connection (GVRET)
- LAWICEL / SLCAN Serial
- [✓ QT SerialBus Devices (SocketCAN, PeakCAN, etc)]
- socketcand (SocketCAN over Ethernet)
- MQTT Connection

SerialBus Device Type: socketcan

Port: can0

Create New Connection



Connection Settings

Connected Devices:

	Type	Subtype	Port	Buses	Status
1	SerialBus	socketcan	can0	1	Connected

Add New Device Connection

Reset Selected Device

Disconnect Selected Device

Bus Details:

Speed: 250000

Listen Only: ☐

Enable Bus: ✓

Enable CAN FD: ☐

Save Bus Settings

0

☐ Enable Console

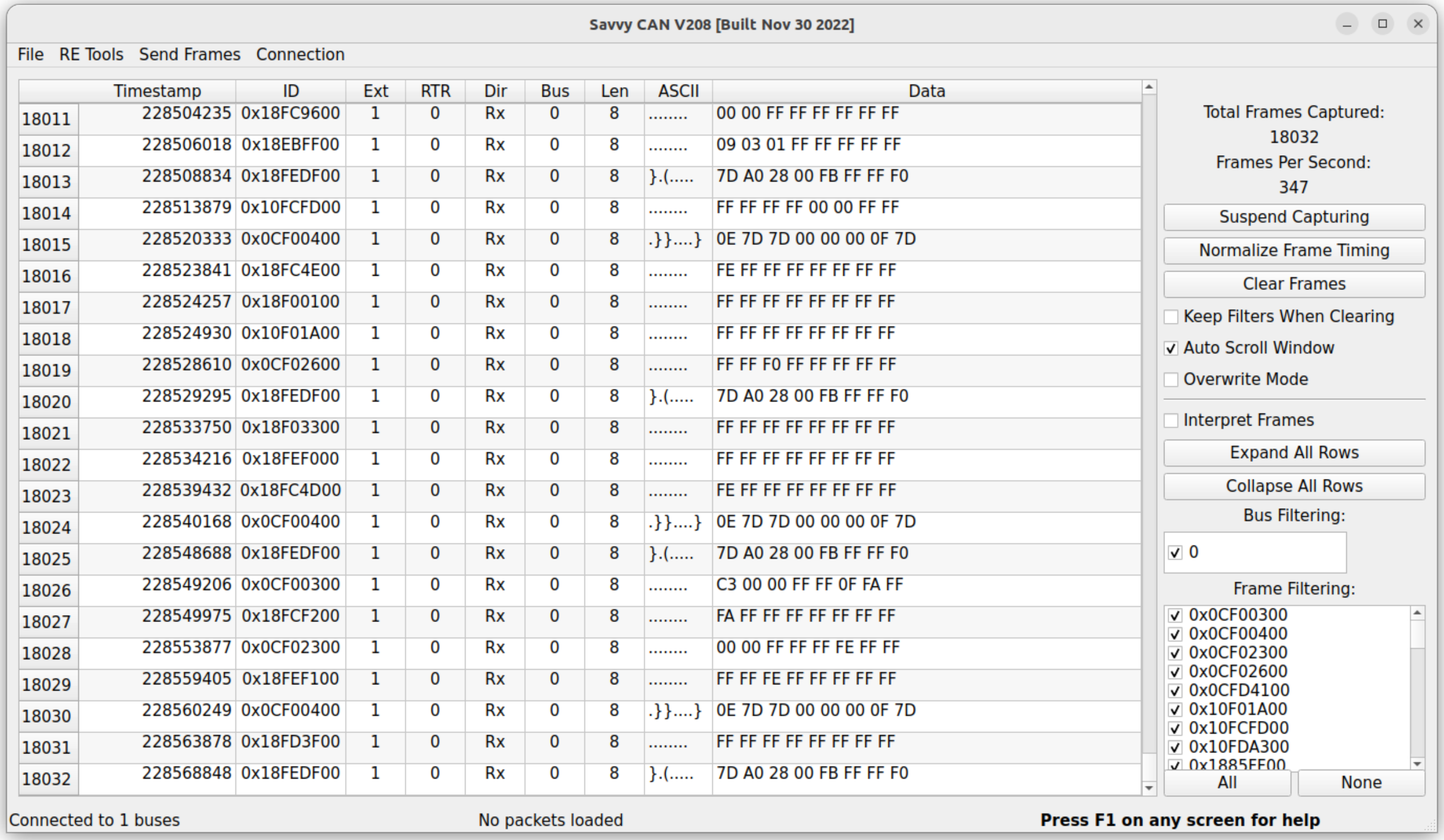
Device Console:

Send:

Send Hex | Send Text | Clear

Exercise: SavvyCAN (Ubuntu), cont.

SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY


Savvy CAN V208 [Built Nov 30 2022]

File RE Tools Send Frames Connection

	Timestamp	ID	Ext	RTR	Dir	Bus	Len	ASCII	Data
18011	228504235	0x18FC9600	1	0	Rx	0	8	....	00 00 FF FF FF FF FF FF
18012	228506018	0x18EBFF00	1	0	Rx	0	8	....	09 03 01 FF FF FF FF FF
18013	228508834	0x18FEDF00	1	0	Rx	0	8	}.(.....	7D A0 28 00 FB FF FF F0
18014	228513879	0x10FCFD00	1	0	Rx	0	8	....	FF FF FF FF 00 00 FF FF
18015	228520333	0x0CF00400	1	0	Rx	0	8	.}}....}	0E 7D 7D 00 00 00 0F 7D
18016	228523841	0x18FC4E00	1	0	Rx	0	8	....	FE FF FF FF FF FF FF FF
18017	228524257	0x18F00100	1	0	Rx	0	8	....	FF FF FF FF FF FF FF FF
18018	228524930	0x10F01A00	1	0	Rx	0	8	....	FF FF FF FF FF FF FF FF
18019	228528610	0x0CF02600	1	0	Rx	0	8	....	FF FF F0 FF FF FF FF FF
18020	228529295	0x18FEDF00	1	0	Rx	0	8	}.(.....	7D A0 28 00 FB FF FF F0
18021	228533750	0x18F03300	1	0	Rx	0	8	....	FF FF FF FF FF FF FF FF
18022	228534216	0x18FEF000	1	0	Rx	0	8	....	FF FF FF FF FF FF FF FF
18023	228539432	0x18FC4D00	1	0	Rx	0	8	....	FE FF FF FF FF FF FF FF
18024	228540168	0x0CF00400	1	0	Rx	0	8	.}}....}	0E 7D 7D 00 00 00 0F 7D
18025	228548688	0x18FEDF00	1	0	Rx	0	8	}.(.....	7D A0 28 00 FB FF FF F0
18026	228549206	0x0CF00300	1	0	Rx	0	8	....	C3 00 00 FF FF 0F FA FF
18027	228549975	0x18FCF200	1	0	Rx	0	8	....	FA FF FF FF FF FF FF FF
18028	228553877	0x0CF02300	1	0	Rx	0	8	....	00 00 FF FF FF FE FF FF
18029	228559405	0x18FEF100	1	0	Rx	0	8	....	FF FF FE FF FF FF FF FF
18030	228560249	0x0CF00400	1	0	Rx	0	8	.}}....}	0E 7D 7D 00 00 00 0F 7D
18031	228563878	0x18FD3F00	1	0	Rx	0	8	....	FF FF FF FF FF FF FF FF
18032	228568848	0x18FEDF00	1	0	Rx	0	8	}.(.....	7D A0 28 00 FB FF FF F0

Total Frames Captured:
18032
Frames Per Second:
347

Suspend Capturing
Normalize Frame Timing
Clear Frames

☐ Keep Filters When Clearing
☑ Auto Scroll Window
☐ Overwrite Mode

☐ Interpret Frames
Expand All Rows
Collapse All Rows

Bus Filtering:
☑ 0

Frame Filtering:
☑ 0x0CF00300
☑ 0x0CF00400
☑ 0x0CF02300
☑ 0x0CF02600
☑ 0x0CFD4100
☑ 0x10F01A00
☑ 0x10FCFD00
☑ 0x10FDA300
☑ 0x1885FF00

All None

Connected to 1 buses   No packets loaded   **Press F1 on any screen for help**

Now we have way to filter and capture data.

But what does this all mean?

Exploring Truck Hardware

LET’S LEARN ABOUT TRUCKS AND TRUCK PARTS.

Interactive Exercise (ask the experts)

In small groups (2-3), go around to the different trucks on this side of the room and…

Identify and locate the following parts:

1. Wheel Speed Sensor
2. Turbo Charger
3. Intake Air Temperature Probe
4. Engine Control Module
5. Diagnostics Port
6. Transmission Selector
7. Air bag (suspension, not restraint)
8. J560 Connector
9. 5th Wheel
10. Aftertreatment system
11. Air Compressor
12. Alternator

Find the answers to these questions:

1. How much fuel do the tanks hold?
2. What is the size of the tires in revolutions per mile?
3. What is the displacement of the engine?
4. How much does the vehicle weigh?
5. What is the class of the vehicle?
6. How many miles are on the odometer?
7. How many batteries are there on the truck?
8. Why do trucks use air brakes?
9. Who makes the engine?
10. Who makes the brakes
11. What are the VINs for the trucks?
12. What is an intercooler for?

Creating Meaning from Messages

HOW DO WE GET ENGINEERING VALUES FROM J1939 PROTOCOL DATA UNITS?

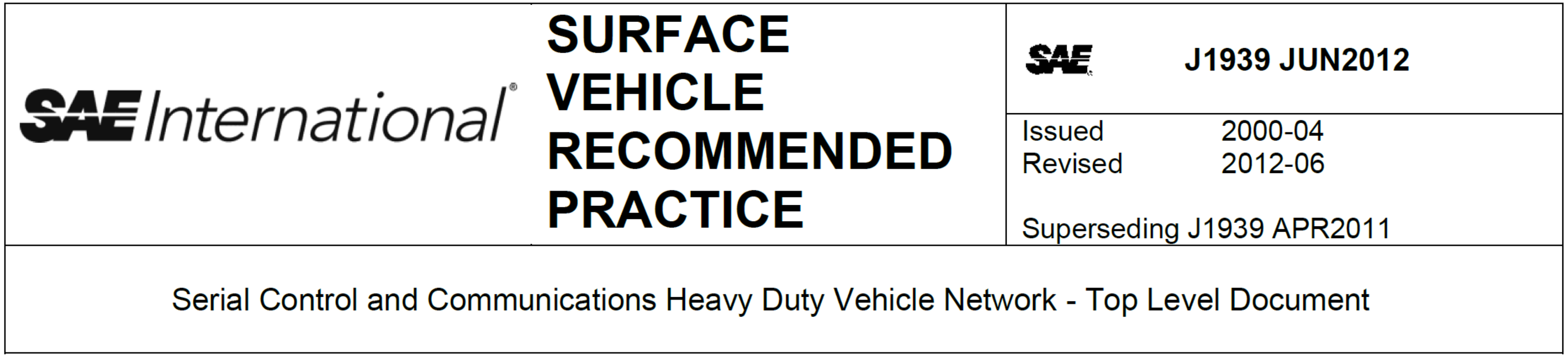
SAE J1939 is Built on CAN

The main features that define J1939 are:

A standardized meaning for 29-bit arbitration identifiers.

A mechanism for sending messages larger than 8 bytes (up to 1785 bytes) using the transport protocol.

The ability for a controller application to negotiate a unique source address.



SAE International

SURFACE VEHICLE RECOMMENDED PRACTICE

SAE J1939 JUN2012

Issued	2000-04
Revised	2012-06

Superseding J1939 APR2011

Serial Control and Communications Heavy Duty Vehicle Network - Top Level Document

J1939 Network Layers

Layer	Name	Standard	Description and Purpose
7	Application	SAE J1939-71 (Applications) SAE J1939-73 (Diagnostics)	Defines how to interpret and compose J1939 messages with engineering values
6	Presentation	Not Used These services are built into the Data Link Layer.	
5	Session		
4	Transport		
3	Network	J1939-31	Clarifies the concept of a gateway between two separate networks.
2	Data Link	J1939-21	Describes how to make a J1939 PDU. Includes details on sending messages up to 1785 bytes long.
1	Physical	J1939-1X	Defined connectors, transceivers, wiring, pinouts, and signaling.

SAE J1939 Standards Organization

Follows the OSI 7-layer model for naming, e.g.:
- J1939-7X are for application layers
- J1939-1X are for physical layers

The standard collection adds much more definition to the CAN communications

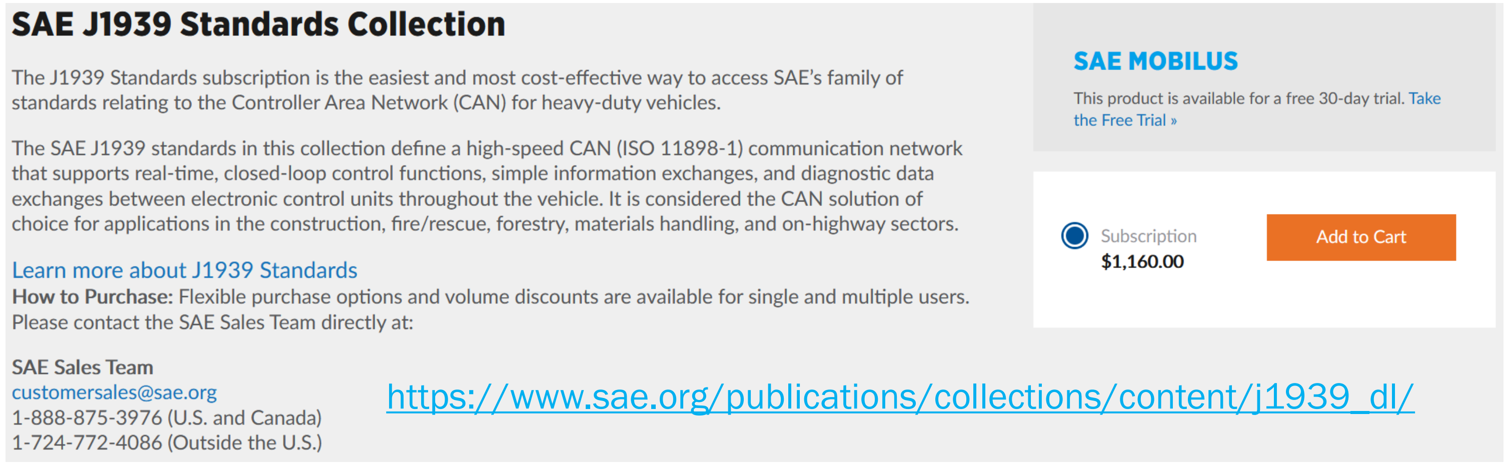
Includes additional "Layers"
- J1939-8X Network Management
- J1939-9X Network Security

J1939 is large and not free

J1939 Accommodates Extensions
- PGN 0xEF00 is Proprietary A
- PGN 0xFFXX is Proprietary B
- PGN 0xDA00 is ISO-15765 (UDS)

A Digital Annex (J1939DA) has the applications defined in an Excel spreadsheet

Recommendation:
- Acquire the Digital Annex first.
- Read J1939-21 for details on the PDU



SAE J1939 Standards Collection

The J1939 Standards subscription is the easiest and most cost-effective way to access SAE's family of standards relating to the Controller Area Network (CAN) for heavy-duty vehicles.

The SAE J1939 standards in this collection define a high-speed CAN (ISO 11898-1) communication network that supports real-time, closed-loop control functions, simple information exchanges, and diagnostic data exchanges between electronic control units throughout the vehicle. It is considered the CAN solution of choice for applications in the construction, fire/rescue, forestry, materials handling, and on-highway sectors.

Learn more about J1939 Standards

How to Purchase: Flexible purchase options and volume discounts are available for single and multiple users. Please contact the SAE Sales Team directly at:

SAE Sales Team
customersales@sae.org
1-888-875-3976 (U.S. and Canada)
1-724-772-4086 (Outside the U.S.)

SAE MOBILUS

This product is available for a free 30-day trial. Take the Free Trial »

● Subscription
$1,160.00

Add to Cart

https://www.sae.org/publications/collections/content/j1939_dl/

Data Decoding and Encoding: Meaning for Bits and Bytes

Common data sizes

- Bit Mapped, like Switch States, (2-bits)
- Single Byte Data (8-bits)
- 2-byte Data (16 bits)
- 4-byte Data (32 bits)
- ASCII data (variable)

Exceptions:

- Field data, engine maps
- Suspect Parameter Numbers (19 bits)
- Failure Mode Indicators (5 bits)
- Others…

Scale, Limits, Offsets, Transfer (SLOTs)

Identifier	SLOT Name	SLOT Type	Scaling	Range	Offset	Length
1	SAEpr11	Pressure	5 kPa/bit	0 to 1,250 kPa	0	1 byte
2	SAEpr13	Pressure	8 kPa/bit	0 to 2,000 kPa	0	1 byte
3	SAEtm11	Time	1 h/bit	0 to 250 h	0	1 byte
4	SAEtm10	Time	1 h/bit	-125 to 125 h	-125 h	1 byte
5	SAEtm12	Time	1 h/bit	-32,127 to 32,128 h	-32,127 h	2 bytes
6	SAEtm06	Time	1 s/bit	0 to 4,211,081,215 s	0	4 bytes
7	SAEad01	Angle/Direction	0.0000001 deg/bit	-210 to 211.1081215 deg	-210 deg	4 bytes
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Bit Transmission Order

Transmission Order: The order in which bits are transmitted over the J1939 Link.

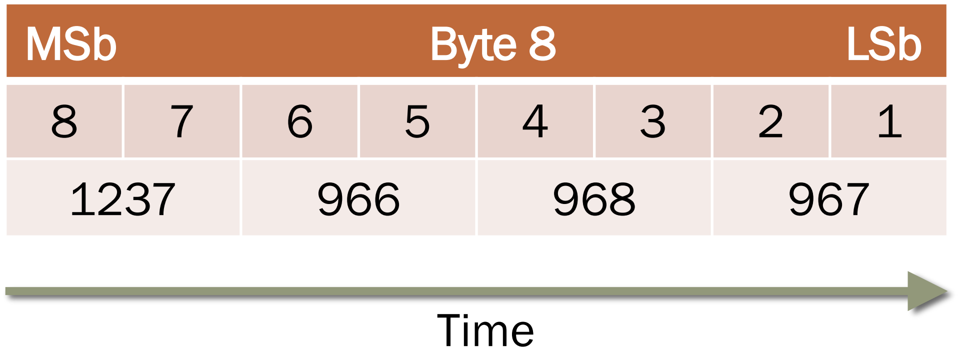
- Data is transmitted in increasing byte order (Byte 1 first, Byte 8 last)
- Bits within the byte are transmitted in decreasing order (Bit 8 first, Bit 1 last)

MSB – Most Significant Byte
MSb – Most Significant Bit

Bit Placement: The location within the byte of the start point of the data

- J1939 uses a convention of Byte.Bit.
- Example from PGN 65265 Cruise Control/Vehicle Speed

SPN	Location	Length	Suspect Parameter
967	8.1	2 bits	Engine Idle Increment Switch
968	8.3	2 bits	Engine Idle Decrement Switch
966	8.5	2 bits	Engine Test Mode Switch
1237	8.7	2 bits	Engine Shutdown Override Switch

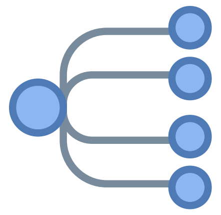


MSb			Byte 8				LSb
8	7	6	5	4	3	2	1
1237		966		968		967	

Types of J1939 Messages

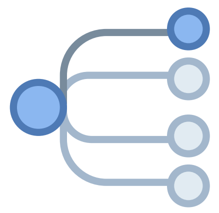
Broadcast

- Messages sent for any controller application (CA) to use
- CAN Arbitration ID specifies a source address (SA) in the last 8 bits of the ID
- Implicitly defined the destination address as 255 (Global)



Point-to-Point

- One controller application sends a message to another
- CAN Arbitration ID specifies a source address (SA) in the last 8 bits of the ID
- A destination address (DA) is in bits 9-16 of the CAN ID.

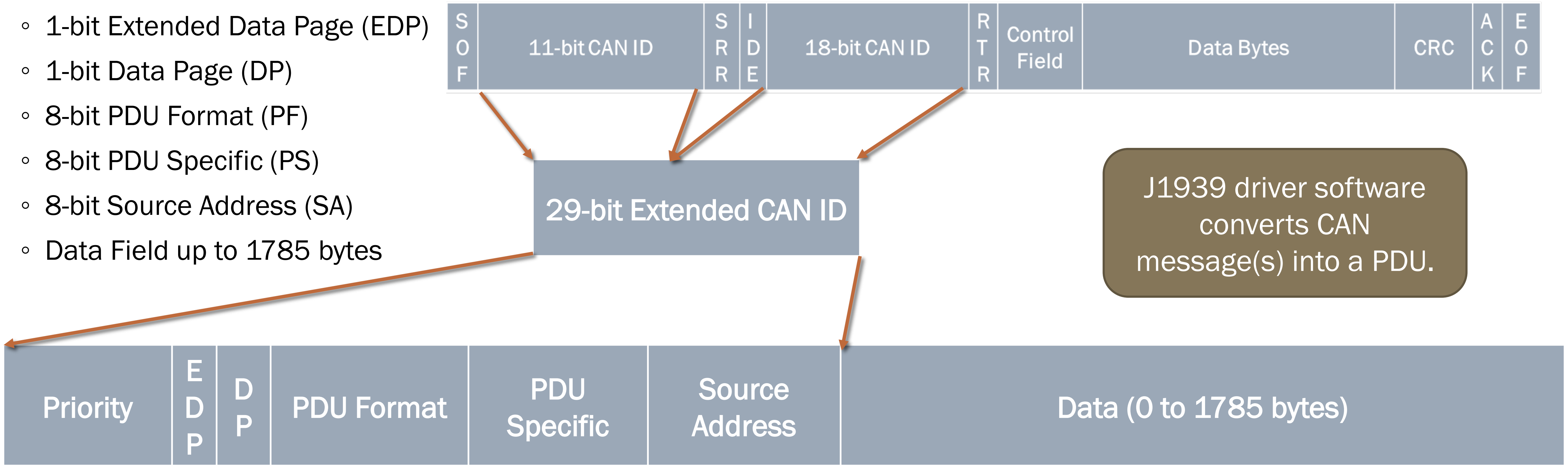


How do some messages implicitly set the destination address to 255?

J1939 Protocol Data Unit

A J1939 message has all the elements in the protocol data unit (PDU)

- 3-bit Priority
- 1-bit Extended Data Page (EDP)
- 1-bit Data Page (DP)
- 8-bit PDU Format (PF)
- 8-bit PDU Specific (PS)
- 8-bit Source Address (SA)
- Data Field up to 1785 bytes



PDU 2 Format Messages

The PDU format #2 is for broadcast messages

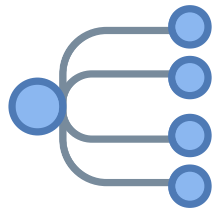
- EDP, DP, PF and PS create the Parameter Group Number (PGN)
- PS becomes the group extension (GE)
- PF value must be greater than or equal to 240 (0xF0)
- Destination address is implied to be 255 (0xFF)

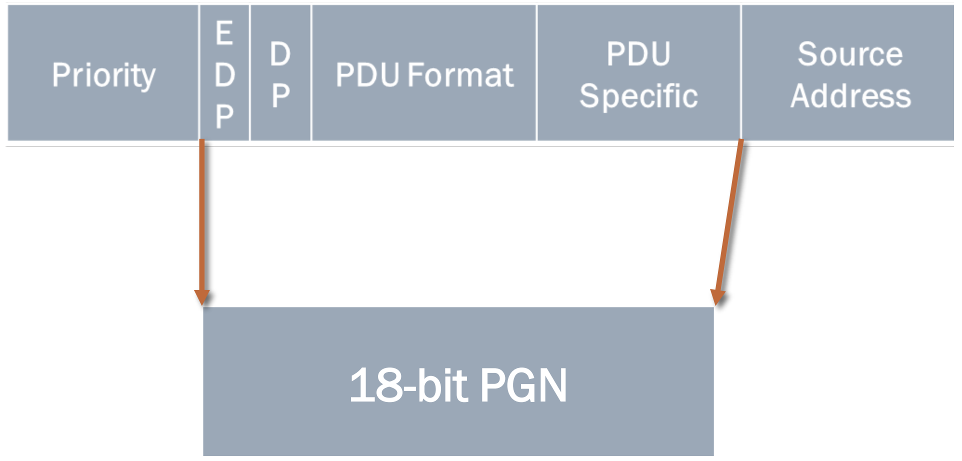
Parameter Groups Numbers are 18 bits.

- Most applications on a truck set the EDP and DP to zero
- Parameter Groups collect similar data for the PDU data field
- PDU 2 messages have a hex values where the leading nibble is F

Examples:

- PGN 65265 (0xFEF1) is for Cruise Control and Vehicle Speed
- PGN 61444 (0xF004) for the Electronic Engine Controller 1 group





PDU 1 Format Messages

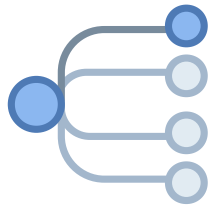
The PDU format #1 is for point-to-point messages

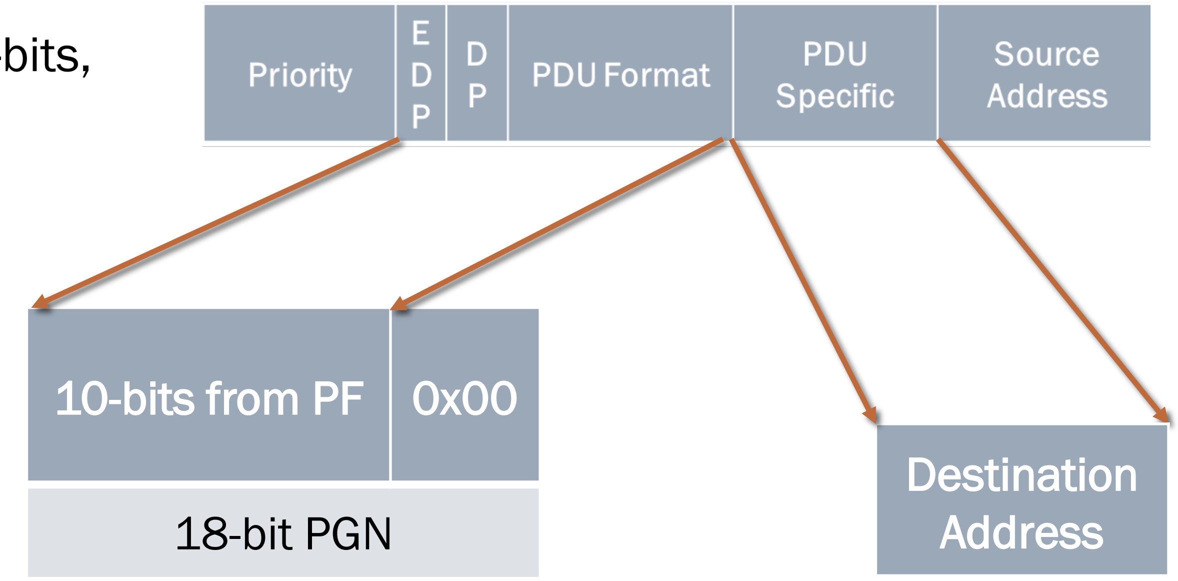
- EDP, DP, PF and 00 create the Parameter Group Number (PGN)
- PS becomes the destination address (DA)

Parameter Group Numbers (PGNs) are still 18-bits, but the last 8 bits are set to zero.

Source and Destination are explicit

PGN values in hex do not have 0xF as the first nibble.





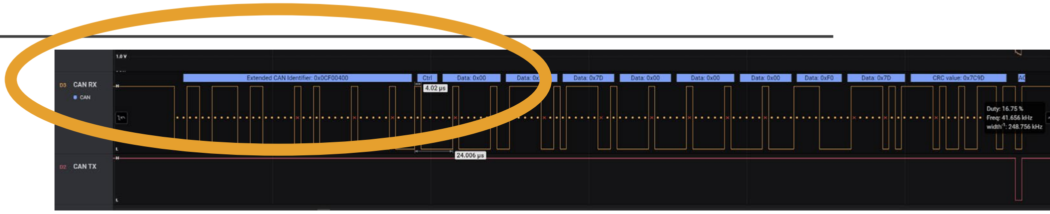
Processing CAN IDs

1. Read the CAN ID as a 32-bit integer
2. Separate the ID into the PDU elements using bit masking and bit shifting
3. Determine if it is a PDU1 or PDU2 message based on the value of PF
    1. If PDU1, PS is the Destination Address
    2. If PDU2, PS is the Group Extension, set Destination Address to 0xFF

```
PRIORITY_MASK = 0x1C000000
EDP_MASK      = 0x02000000
DP_MASK       = 0x01000000
PF_MASK       = 0x00FF0000
PS_MASK       = 0x0000FF00
SA_MASK       = 0x000000FF
PDU1_PGN_MASK = 0x03FF0000
PDU2_PGN_MASK = 0x03FFFF00
```

SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

Bit Masking and Shifting



Example: Determine the PGN from the CAN Frame Capture

ID Hex Nibbles	0	C				F				0				0				4				0				0			
ID Binary	0	1	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
PGN Mask Hex	0	3				F				F				F				F				0				0			
Mask Binary	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
AND Result	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Shift right 8 bits									0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0
Hex after shift									0	0				F				0				0				4			

0x00F004 = 61444 dec

Python Based Parsing

```
def get_j1939_from_id(can_id):
    priority = (PRIORITY_MASK & can_id) >> 26
    edp = (EDP_MASK & can_id) >> 25
    dp = (DP_MASK & can_id) >> 24
    PF = (can_id & PF_MASK) >> 16
    PS = (can_id & PS_MASK) >> 8
    SA = (can_id & SA_MASK)
    if PF >= 0xF0: #240
       DA = 255
       PGN = (can_id & PDU2_PGN_MASK) >> 8
    else:
       DA = PS
       PGN = (can_id & PDU1_PGN_MASK) >> 8
    return priority,PGN,DA,SA
```

```
PRIORITY_MASK = 0x1C000000
EDP_MASK      = 0x02000000
DP_MASK       = 0x01000000
PF_MASK       = 0x00FF0000
PS_MASK       = 0x0000FF00
SA_MASK       = 0x000000FF
PDU1_PGN_MASK = 0x03FF0000
PDU2_PGN_MASK = 0x03FFFF00
```

EDP: Extended Data Page
DP: Data Page
PF: PDU Format
PS: PDU Specific
PDU: Protocol Data Unit
PGN: Parameter Group Number
DA: Destination Address
SA: Source Address

SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

Decoding Example:
Accelerator Pedal Low Idle Switch

Given the following CAN message (hex):

```
0CF00300   [8]  D0 5A 25 FF FF 0F A0 81
```

Break the CAN ID into J1939 values

- 0x0C is priority 3
- F004 is PDU2 format
  - PGN is 0xF003 = 61443, Electronic Engine Control 2
  - Destination Address is 0xFF (implied)
- Source address is 0x00 = 0, Engine #1

Determine <u>some</u> Suspect Parameters in the data

- 558: Accelerator Pedal 1 Low Idle Switch
- 559: Accelerator Pedal 1 Kickdown Switch
- 1437: Road Speed Limit Status
- 2970: Accelerator Pedal 2 Low Idle Switch
- 91: Accelerator Pedal Position 1
- 92: Engine Percent Load At Current Speed
- 974: Remote Accelerator Pedal Position

Bit 8

Bit 1

Byte Position	Byte 1				Byte 2	Byte 3					
Hex Values	0xD0				0x5A	0x25					
Binary	0b 1101 0000										
SPNs	2970	1437	559	558	91	92	974	29	2979		
Engineering	N/A	On	Off	Off			N/A	N/A			

Bit Encoding for SPN 558
00 - Accelerator pedal 1 not in low idle condition
01 - Accelerator pedal 1 in low idle condition
10 - Error
11 - Not available

Decoding Example: Accelerator Pedal Position

Given the following CAN message (hex):

```
0CF00300   [8]  D0 5A 25 FF FF 0F A0 81
```

1. Break the CAN ID into J1939 values
   a) 0x0C is priority 3
   b) F003 is PDU2 format
      i. PGN is 0xF003 = 61443, Electronic Engine Control 2
      ii. Destination Address is 0xFF (implied)
   c) Source address is 0x00 = 0, Engine #1
2. Determine <u>some</u> Suspect Parameters in the data
   a) 558: Accelerator Pedal 1 Low Idle Switch
   b) 559: Accelerator Pedal 1 Kickdown Switch
   c) 91: Accelerator Pedal Position 1
   d) 92: Engine Percent Load At Current Speed
   e) 974: Remote Accelerator Pedal Position
   f) 29: Accelerator Pedal Position 2
   g) 2979: Vehicle Acceleration Rate Limit Status
   h) 3357: Actual Maximum Available Engine % Torque
   i) 5398: Estimated Pumping – Percent Torque

Byte Position	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Hex Values	0xD0	0x5A	0x25	0xFF	0xFF	0x0F	0xA0	0x81
SPNs	558, 559…	91	92	974	29	2979…	3357	5398
Engineering		90*0.4 = 36%		N/A	N/A		160*0.4= 64%	129-125=32%

Byte order for Integers (Endianness)

This Photo by Unknown Author is licensed under CC BY-NC-ND

Post office boxes have mail loaded from the inside and taken out through the front.

This Photo by Unknown Author is licensed under CC BY-SA-NC

This Photo by Unknown Author is licensed under CC BY-NC

Mailboxes have mail loaded and removed from the front.

Byte order for Integers (Endianness)

Let's pretend our mailboxes are containers to hold bytes representing integers.

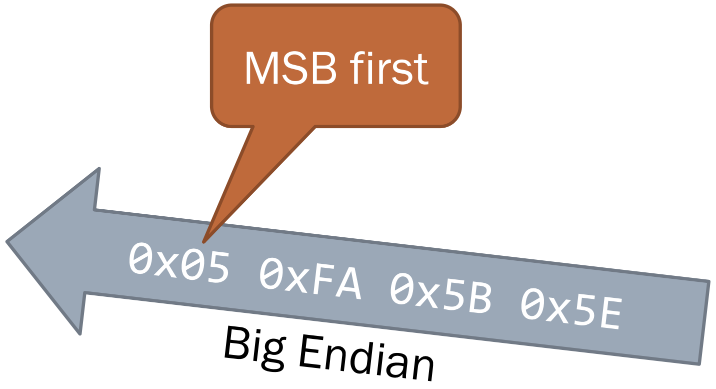
Example: the integer 100,293,486 is 0x05FA5B6E in hex, which is 4 bytes:

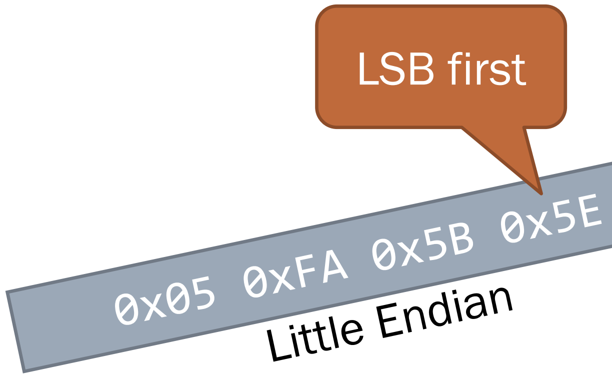
Most Significant Byte

`0x05 0xFA 0x5B 0x5E`

Least Significant Byte

Pretend each byte is a small package. In what order should the postmaster insert the bytes into the mailbox so they are in order when the customer extracts them?





Byte order for Integers (Endianness)

SAE J1939 encodes multi-byte integers in the Little Endian format.

This give the appearance the bytes are reversed and need to be swapped to interpret

Intel format = Little Endian = Least Significant Byte first

Motorola format = Big Endian = Most Significant Byte first (as we typically read and write)

Byte Length	Decimal	Hex	Big Endian	Little Endian (J1939)
1	241	F1	0xF1	0xF1
2	743	2E7	0x02 0xE7	0xE7 0x02
2	25	19	0x00 0x19	0x19 0x00
4	1,890,056,399	70A7F8CF	0x70 0xA7 0xF8 0xCF	0xCF 0xF8 0xA7 0x70

Endianness doesn't affect single byte integers.

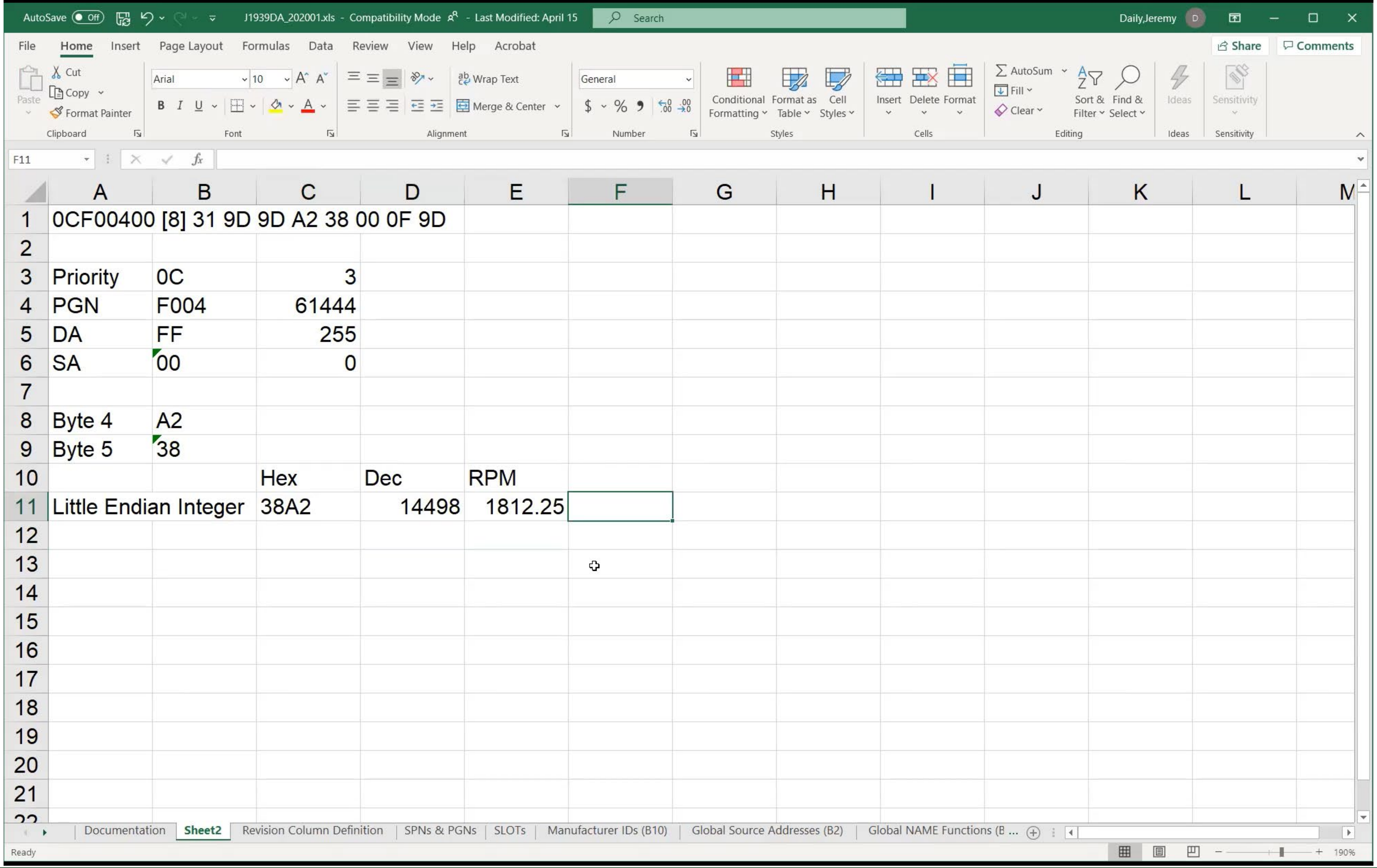
Decoding Example: Engine Speed (RPM)

Given the following CAN message (hex):
`0CF00400 [8] 31 9D 9D A2 38 00 0F 9D`

1. Break the CAN ID into J1939 values
   a) 0x0C is priority 3
   b) F004 is PDU2 format
      i. PGN is 0xF004 = 61444, Electronic Engine Control 1
      ii. Destination Address is 0xFF (implied)
   c) Source address is 0x00 = 0, Engine #1

2. Determine Suspect Parameters in the data
   a) 889: Engine Torque Mode
   b) 4154: Actual Percent Torque
   c) 512: Driver's Demand Engine - Percent Torque
   d) 513: Actual Engine Percent Torque
   e) 190: Engine Speed
   f) 1483: SA of Controlling device
   g) 1675: Engine Starter Mode
   h) 2432: Engine Demand- Percent Torque

Byte Position	Byte 1		Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7		Byte 8
Hex Values	0x31		0x9D	0x9D	0xA2	0x38	0x00	0x0F		0x9D
SPNs	899	4154	512	513	190		1483	1675	Res	2432
Engineering			32%	32%	14,498/8 = 1812.25				N/A	157-125=32%



	A	B	C	D	E
1	0CF00400 [8] 31 9D 9D A2 38 00 0F 9D				
2					
3	Priority	0C	3		
4	PGN	F004	61444		
5	DA	FF	255		
6	SA	00	0		
7					
8	Byte 4	A2			
9	Byte 5	38			
10			Hex	Dec	RPM
11	Little Endian Integer		38A2	14498	1812.25

Decoding Example: Vehicle Miles

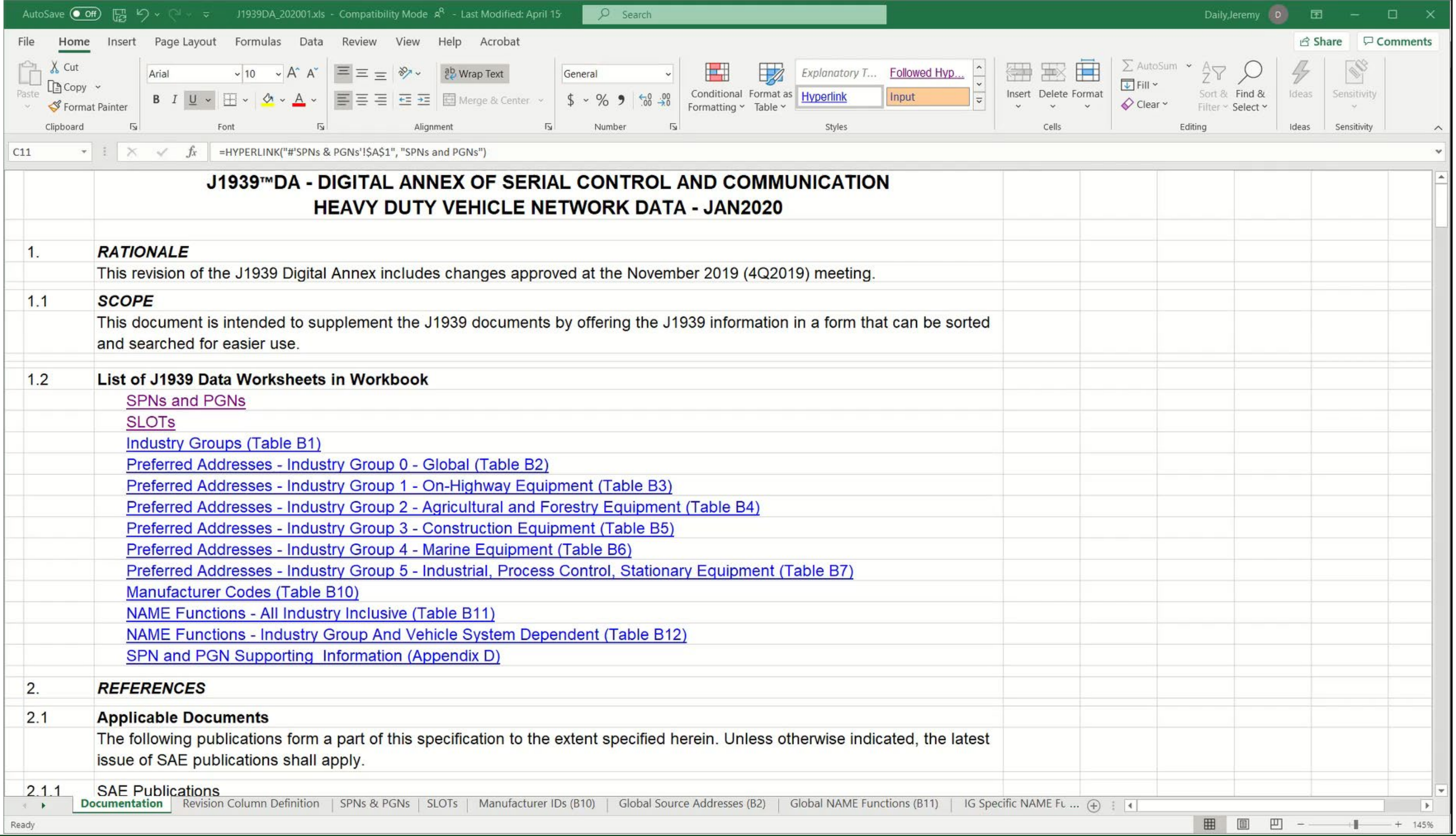
SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

Determine the Odometer reading from the J1939 data.

Not all data defined in J1939 is present on the network.

Strategy:

1. Look up the SPN for distance
2. Search the J1939 Standard for the entry
3. Find the message in a log
4. Decode the logged message



```
=HYPERLINK("#'SPNs & PGNs'!$A$1", "SPNs and PGNs")
```

J1939™DA - DIGITAL ANNEX OF SERIAL CONTROL AND COMMUNICATION
HEAVY DUTY VEHICLE NETWORK DATA - JAN2020

1. ***RATIONALE***

This revision of the J1939 Digital Annex includes changes approved at the November 2019 (4Q2019) meeting.

1.1 ***SCOPE***

This document is intended to supplement the J1939 documents by offering the J1939 information in a form that can be sorted and searched for easier use.

1.2 **List of J1939 Data Worksheets in Workbook**

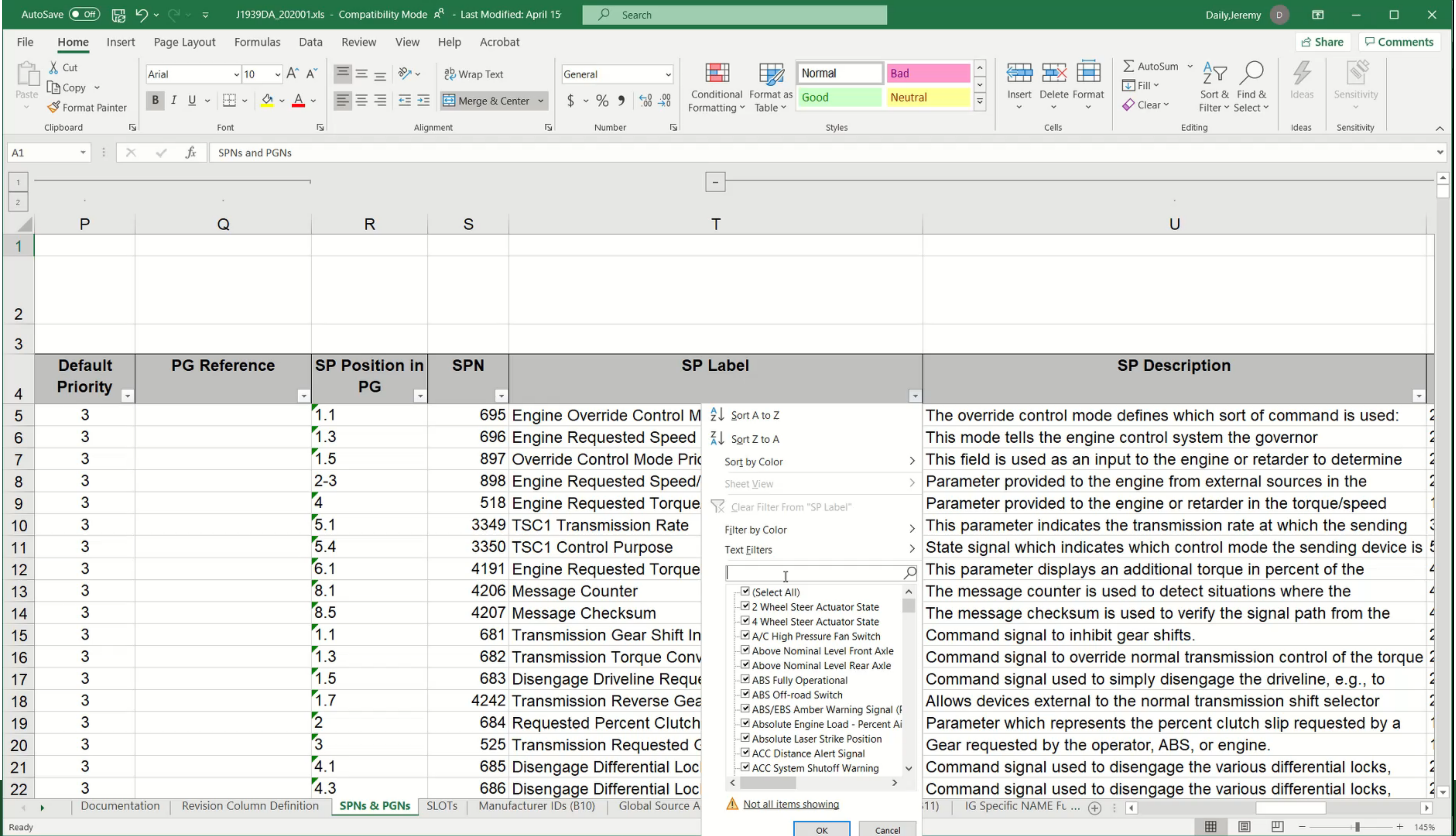
- SPNs and PGNs
- SLOTs
- Industry Groups (Table B1)
- Preferred Addresses - Industry Group 0 - Global (Table B2)
- Preferred Addresses - Industry Group 1 - On-Highway Equipment (Table B3)
- Preferred Addresses - Industry Group 2 - Agricultural and Forestry Equipment (Table B4)
- Preferred Addresses - Industry Group 3 - Construction Equipment (Table B5)
- Preferred Addresses - Industry Group 4 - Marine Equipment (Table B6)
- Preferred Addresses - Industry Group 5 - Industrial, Process Control, Stationary Equipment (Table B7)
- Manufacturer Codes (Table B10)
- NAME Functions - All Industry Inclusive (Table B11)
- NAME Functions - Industry Group And Vehicle System Dependent (Table B12)
- SPN and PGN Supporting Information (Appendix D)

2. ***REFERENCES***

2.1 **Applicable Documents**

The following publications form a part of this specification to the extent specified herein. Unless otherwise indicated, the latest issue of SAE publications shall apply.

2.1.1 SAE Publications



	P	Q	R	S	T	U
4	Default Priority	PG Reference	SP Position in PG	SPN	SP Label	SP Description
5	3		1.1	695	Engine Override Control M	The override control mode defines which sort of command is used:
6	3		1.3	696	Engine Requested Speed	This mode tells the engine control system the governor
7	3		1.5	897	Override Control Mode Pric	This field is used as an input to the engine or retarder to determine
8	3		2-3	898	Engine Requested Speed/	Parameter provided to the engine from external sources in the
9	3		4	518	Engine Requested Torque	Parameter provided to the engine or retarder in the torque/speed
10	3		5.1	3349	TSC1 Transmission Rate	This parameter indicates the transmission rate at which the sending
11	3		5.4	3350	TSC1 Control Purpose	State signal which indicates which control mode the sending device is
12	3		6.1	4191	Engine Requested Torque	This parameter displays an additional torque in percent of the
13	3		8.1	4206	Message Counter	The message counter is used to detect situations where the
14	3		8.5	4207	Message Checksum	The message checksum is used to verify the signal path from the
15	3		1.1	681	Transmission Gear Shift In	Command signal to inhibit gear shifts.
16	3		1.3	682	Transmission Torque Conv	Command signal to override normal transmission control of the torque
17	3		1.5	683	Disengage Driveline Reque	Command signal used to simply disengage the driveline, e.g., to
18	3		1.7	4242	Transmission Reverse Gea	Allows devices external to the normal transmission shift selector
19	3		2	684	Requested Percent Clutch	Parameter which represents the percent clutch slip requested by a
20	3		3	525	Transmission Requested G	Gear requested by the operator, ABS, or engine.
21	3		4.1	685	Disengage Differential Loc	Command signal used to disengage the various differential locks,
22	3		4.3	686	Disengage Differential Loc	Command signal used to disengage the various differential locks,

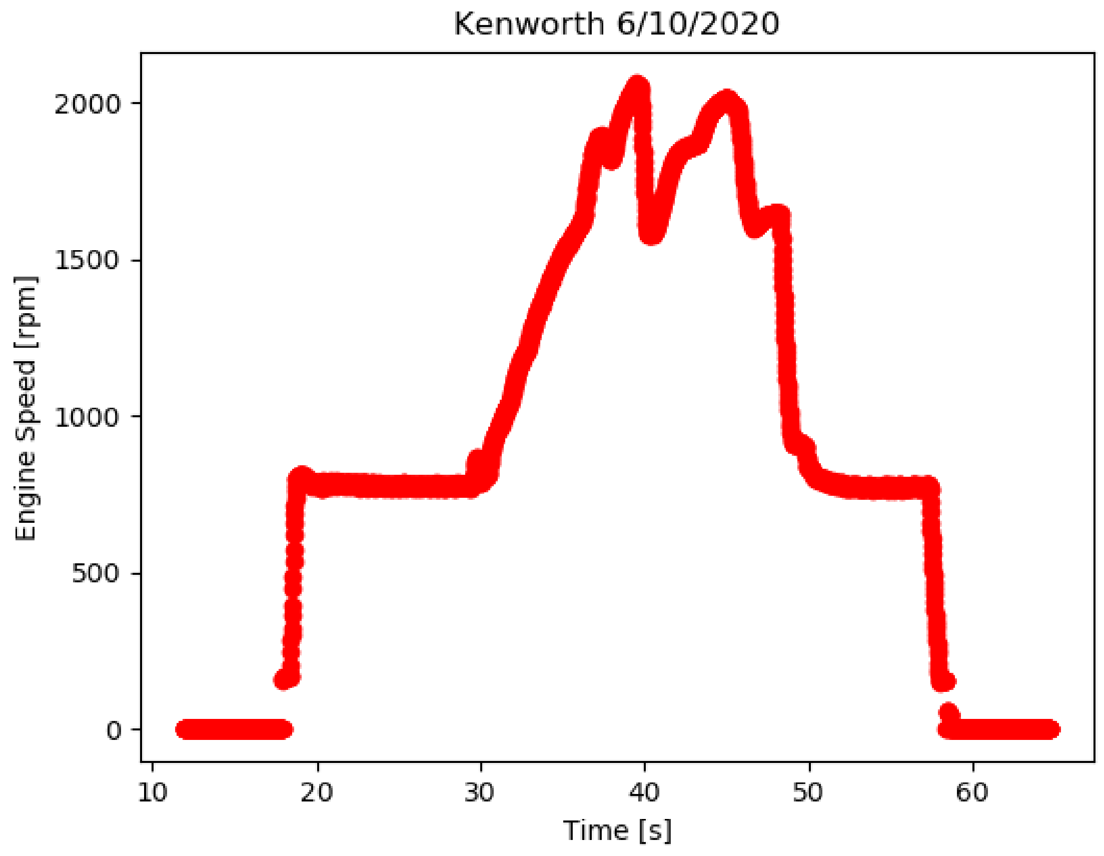
- ✓ (Select All)
- ✓ 2 Wheel Steer Actuator State
- ✓ 4 Wheel Steer Actuator State
- ✓ A/C High Pressure Fan Switch
- ✓ Above Nominal Level Front Axle
- ✓ Above Nominal Level Rear Axle
- ✓ ABS Fully Operational
- ✓ ABS Off-road Switch
- ✓ ABS/EBS Amber Warning Signal (F
- ✓ Absolute Engine Load - Percent Ai
- ✓ Absolute Laser Strike Position
- ✓ ACC Distance Alert Signal
- ✓ ACC System Shutoff Warning

After Class Exercise

Signal Interpretation

A candump data log was capture during a startup seqence for a truck using Linux SocketCAN. The data comes from 2014 Class 6 truck with a box van where the operator started the engine, pressed the accelerator pedal and turned the engine off. The challenge is to determine the highest engine speed in RPM base on the log file.

1. Some other questions for consideration: How many ECUs are on the network?
2. What is the vehicle mileage?
3. Did the vehicle's wheels rotate?



https://www.engr.colostate.edu/~jdaily/cyber/KWTruck.txt

J1939 Transport Protocol

HOW CAN WE SEND MESSAGES LARGER THAN 8 BYTES IN A CAN FRAME?

J1939 Transport Protocol

Data more than 8 bytes in length requires multiple CAN frames to send the data.

Two Approaches that follow PDU formats

- Request to Send/Clear to Send (RTS/CTS) – point-to-point messaging
- Broadcast Announce Message (BAM) – global address
- Approach is determined with the first byte of the Connection Management Message
  - If 32 (0x20), then BAM
  - If 16 (0x10), then RTS
  - If 17 (0x11), then CTS

Three Parameter Groups to track

- Transport Protocol – Connection Management (TP.CM), PGN 60416 (0xEC00)
- Transport Protocol – Data Transfer (TP.DT), PGN 60160 (0xEB00)
- PGN of the data being transported

Details are in SAE J1939-21

J1939 Transport Protocol VIN Example

The following data were on J1939:

```
CAN ID: CAN Data (in hex)
1CECFF00: 20 12 00 03 FF EC FE 00
1CEBFF00: 01 31 58 4B 59 44 50 39
1CEBFF00: 02 58 37 46 4A 34 36 39
1CEBFF00: 03 30 35 38 2A FF FF FF
```

Parse the CAN ID into J1939 parameters:

- 0x1C000000 -> Priority = 7 (lowest)
- 0x00EC0000 -> PGN = 60416 (TP.CM)
- 0x00EB0000 -> PGN = 60160 (TP.DT)
- 0x0000FF00 -> Destination = 255 (Global)
- 0x00000000 -> Source Address = 0 (Engine 1)

J1939 Transport Protocol VIN Example (cont.)

Transport Protocol – Connection Management

```
1CECFF00: 20 12 00 03 FF EC FE 00
```

- `20 – Control Byte = BAM`
- `12 00 – Message size (18 bytes)`
- `03 – Number of packets (3)`
- `EC FE 00 – PGN of message`
  `(0x00FEEC = 65260 Vehicle Identification)`

Connection Mode: BAM

- Byte 1: Control byte = 32 (0x20), Broadcast Announce Message (BAM)
- Bytes 2,3: Total message size, number of bytes (Big Endian or reverse byte order)
- Byte 4: Total number of packets
- Byte 5: Reserved (0xFF)
- Bytes 6,7,8: Parameter Group Number of the packeted message (Big Endian)

Note: The destination on a BAM is often 255 for all the nodes.

1XKYDP9X7FJ469058*

J1939 Transport Protocol VIN Example (cont.)

SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

Transport Protocol – Data Transfer

```
1CEBFF00: 01 31 58 4B 59 44 50 39
1CEBFF00: 02 58 37 46 4A 34 36 39
1CEBFF00: 03 30 35 38 2A FF FF FF
```

- NN – Sequence Number (01 to FF)
- Data totaling the number of bytes in TC.CM
- FF FF FF – Filler for an 8-byte message

A maximum of 255 messages with 7 bytes each means a total of 255*7 = 1785 bytes maximum for each J1939 transport protocol message.

Decoded value from ASCII:

```
 31 58 4B 59 44 50 39 58 37
  46 4A 34 36 39 30 35 38 2A

 1  X  K  Y  D  P  9  X  7
  F  J  4  6  9  0  5  8  *

 Or 1XKYDP9X7FJ469058
```

(VIN is usually 17 characters, so the * is dropped)

J1939 Request Messages

Many data available from and ECU are by request only. Examples include:

- Engine hours
- VIN
- Component Information

PGN 59904 (0xEA00) is for a Request

- Only 3 bytes long
- Data is the PGN being requested
- Should only be used 2-3 times per second

Example:

```
 CAN ID    CAN DATA
18EA0FF9   EC FE 00
```

- 18 – Priority (6 default)
- EA00 – Request PGN (59904)
- 0F – Destination Address (Retarder)
- F9 – Source Address (249: Off-Board Diagnostic Tool)
- EC FE 00 – PGN 65260: VIN (Reverse byte order)

Note: this is a point-to-point request to the retarder from the service tool.

The response may be BAM or RTS/CTS

J1939 Transport Protocol Vulnerabilities

Certain implementations may trust external values for setting up transport protocols, which may lead to issues.

1. Request Overloads
2. Connection Exhaustion
3. BAM Block
4. Malicious Clear to Sent
5. Memory Leak

Exploiting Transport Protocol Vulnerabilities in SAE J1939 Networks

Rik Chatterjee
Colorado State University
rik.chatterjee@colostate.edu

Subhojeet Mukherjee
Colorado State University
subhojeet.mukherjee@colostate.edu

Jeremy Daily
Colorado State University
jeremy.daily@colostate.edu

***Abstract*—Modern vehicles are equipped with embedded computers that utilize standard protocols for internal communication. The SAE J1939 protocols running on top of the Controller Area Network (CAN) protocol is the primary choice of internal communication for embedded computers in medium and heavy-duty vehicles. This paper presents five different cases in which potential shortcomings of the SAE J1939 standards are exploited to launch attacks on in-vehicle computers that constitute SAE J1939 networks.**

In the first two of these scenarios, we validate the previously proposed attack hypothesis on more comprehensive testing setups. In the later three of these scenarios, we present newer attack vectors that can be executed on bench test setups and deployed SAE J1939 networks.

For the purpose of demonstration, we use bench-level test systems with real electronic control units connected to a CAN bus. Additional testing was conducted on a 2014 Kenworth T270 Class 6 truck under both stationary and driving conditions. Test results show how protocol attacks can target specific ECUs. These attacks should be considered by engineers and programmers implementing the J1939 protocol stack in their communications subsystem.

I. INTRODUCTION

Medium and heavy-duty (MHD) vehicles are a part of the US critical infrastructure, transporting goods, supporting emergency services, and so on. Modern MHD vehicles are electronified: most mechanical operations being controlled through embedded computers referred to as Electronic Control Units (ECUs). Within the vehicle, ECUs form networks to communicate mission critical information with each other on a bus topology. For MHD vehicles, the primary choice of communication specifications within these networks is the SAE J1939 standard. SAE J1939 documents [1] are organized in layers much like the ISO/OSI [2] standards for traditional IT networking. At its lowest layers, the SAE J1939 standards utilize the Controller Area Network (CAN) specifications [3] to facilitate the in-vehicle information exchange.

CAN is used widely in automotive networking and aspects of its (in)security has be thoroughly demonstrated. For example, it has been shown, with access to remote and local entry points (vulnerable ECUs) to the CAN network, one can launch attacks on the vehicle to control or disrupt its operations. MHD vehicles also expose similar entry points [4] and, aside from CAN specific attacks, it has been shown that attacks can also be launched on the SAE J1939 protocols. Even so, the number of demonstrated attacks is still limited: Burakova et al. [5] have demonstrated a couple of attacks on the application layer specification of the SAE J1939 standards, Murvay et al. [6] have focused on weaknesses at the network management layer, and Mukherjee et al. [7] have targeted specific protocols at the data-link layer of the specifications.

While the application and network management layers are critical to the cyber-physical operations of the vehicle, important message transportation specifications are made in the data-link layer standards. As such, in this work we demonstrate newer attacks at the data-link layer of the SAE J1939 specifications that broaden the horizon of cyber threats already created by Mukherjee et al. [7]. Moreover, we validate two attacks that Mukherjee et al. demonstrated to work on laboratory test benches. For our validations we use more comprehensive testing setups, as well as a 2014 Kenworth T270 truck; the goal being to demonstrate the applicability and impact of the attacks on different platforms.

The overarching goal of this paper is to enhance the threatscape for in-vehicle networking applications in MHD vehicles. To that end, the rest of the paper is organized as follows. In section II we present a brief overview of SAE J1939, as required to clearly comprehend the contributions made in this paper. In section III we briefly cover the related work in this area. In section IV, we present a description of the testing setup used in this work. In section V, we describe the attack experimentation carried out during the course of the work. Finally, in section VI, we finish with concluding remarks and a brief introspection of the future work.

II. BACKGROUND ON SAE J1939

In-vehicle communication in medium and heavy-duty vehicles is mostly guided by the SAE J1939 standards. SAE J1939 messages carry operational parameters like engine speed, vehicle speed, switch status, etc. These parameters are bundled into logical groups referred to as Parameter Groups (PG). Each PG is identified by a unique number called a Parameter Group Number (PGN), which is also embedded in the message. Information in the J1939 message is carried in a J1939

Symposium on Vehicle Security and Privacy (VehicleSec) 2023
27 February 2023, San Diego, CA, USA
ISBN 1-891562-88-6
https://dx.doi.org/10.14722/vehiclesec.2023.23053
www.ndss-symposium.org

https://www.engr.colostate.edu/~jdaily/presentations/2023%20Transport%20Protocol%20Vulnerabilities%20on%20SAE%20J1939%20Networks.pdf

https://dx.doi.org/10.14722/vehiclesec.2023.23053

J1939 Address Claim

HOW DOES A NETWORK KEEP TRACK OF THE SOURCE AND DESTINATION ADDRESSES IF IT CHANGES?

J1939 Address Claim

Each controller application (node) on the network should have its own source address.

Some ECUs have multiple controller applications.

- SA 0x00: Engine #1
- SA 0x0F: Engine Retarder

Address Claims happen

- On Boot
- When requested
- In response to other claims for the same address

Address Claim Parameter Group Number

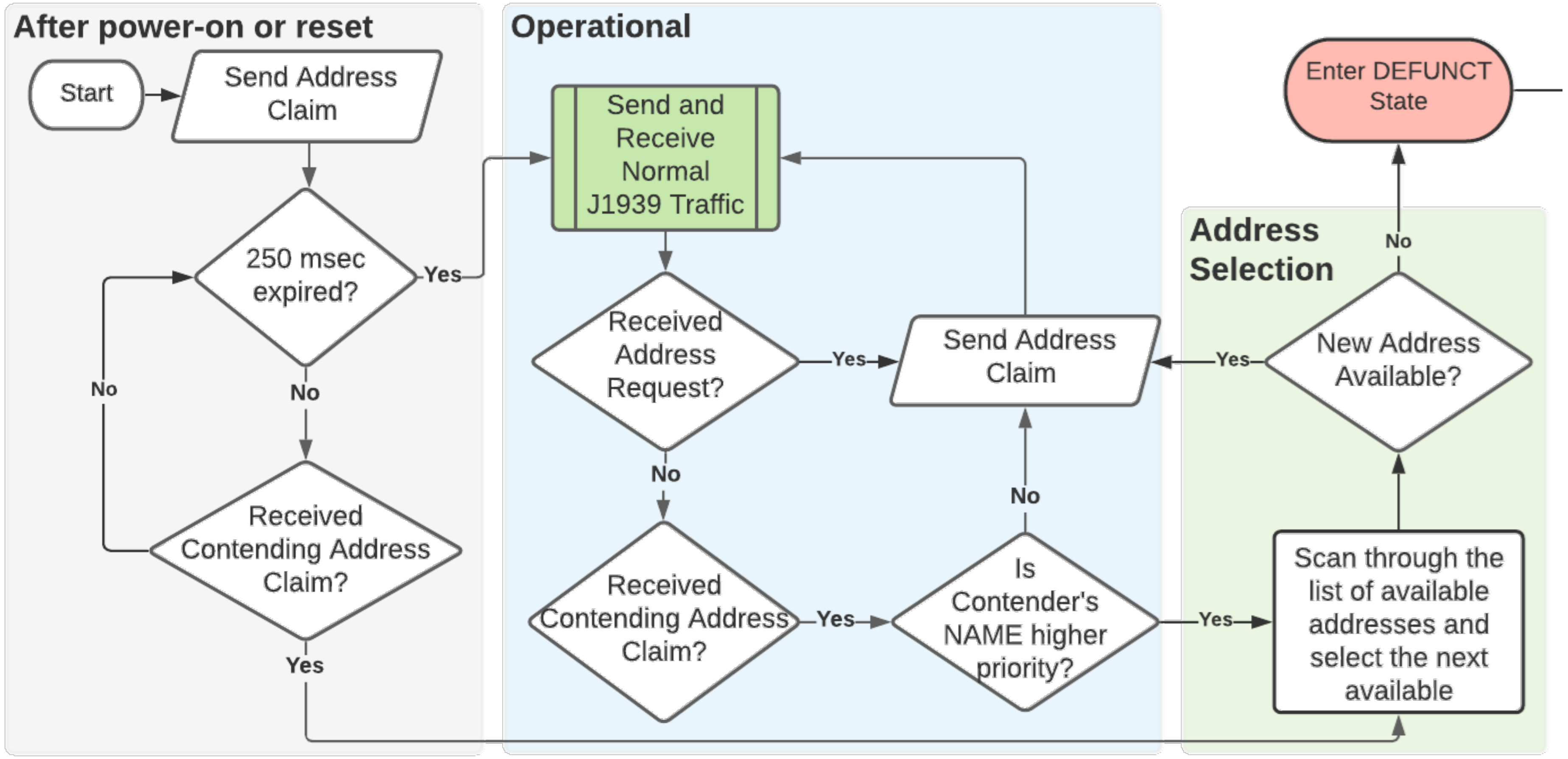
- 60928 (0xEE00)
- Mostly uses the Global destination address (0xFF)
- Source address is the address being claimed

Transmission Address Claim example:

```
18EEFF03: 64 00 40 00 00 03 03 10
```

- `18 – Priority 6 (default)`
- `EE – PGN 60928 = Address Claimed`
- `FF – Global Destination Address`
- `03 – Source address for Transmission #1`
- `64 00 40 00 00 03 03 10 - NAME Field`

How Address Claiming Works


After power-on or reset
Start
Send Address Claim
250 msec expired?
Yes
No
No
Received Contending Address Claim?
Yes
Operational
Send and Receive Normal J1939 Traffic
Received Address Request?
Yes
No
Send Address Claim
No
Received Contending Address Claim?
Yes
Is Contender's NAME higher priority?
Yes
Address Selection
Enter DEFUNCT State
No
New Address Available?
Yes
Scan through the list of available addresses and select the next available

See SAE J1939-81 Network Management

Address NAME Field

Arbitrary Address Capable	Industry Group	Vehicle System Instance	Vehicle System	Reserved	Function	Function Instance	ECU Instance	Manufacturer Code	Identity Number
	SAE		SAE	SAE	SAE			SAE	
1 bit	3 bits	4 bits	7 bits	1 bit	8 bits	5 bits	3 bits	11 bits	21 bits

- From SAE J1939-81, the following NAME field is 64 Bits (8 bytes) long.
- Value is translated with little endian format (Intel), so the least significant byte is first.
- Example 1: Caterpillar C15 with ADEM4 ECU

```
can1  18EEFF00   [8]  D0 6B 01 01 00 00 00 80
```

- Example 2: Detroit Diesel CPC3Evo

```
can1  18EEFF00   [8]  00 00 C0 01 00 00 00 00
```

- Additional Examples

CAN ID has:
- Priority = 6,
- Parameter Group Number = 0xEE00,
- Destination Address = 0xFF (Global),
- Claimed Source Address = 0x00 (Engine #1)

Example 1: Caterpillar

```
can1  18EEFF00   [8]  D0 6B 01 01 00 00 00 80
```

Byte 8 (0x80) = 0b1000 0000, which means:

- it is arbitrary address capable,
- the industry group is 0 (global), and
- the vehicle system instance is zero.

Byte 5 -7 (00 00 00), which means:

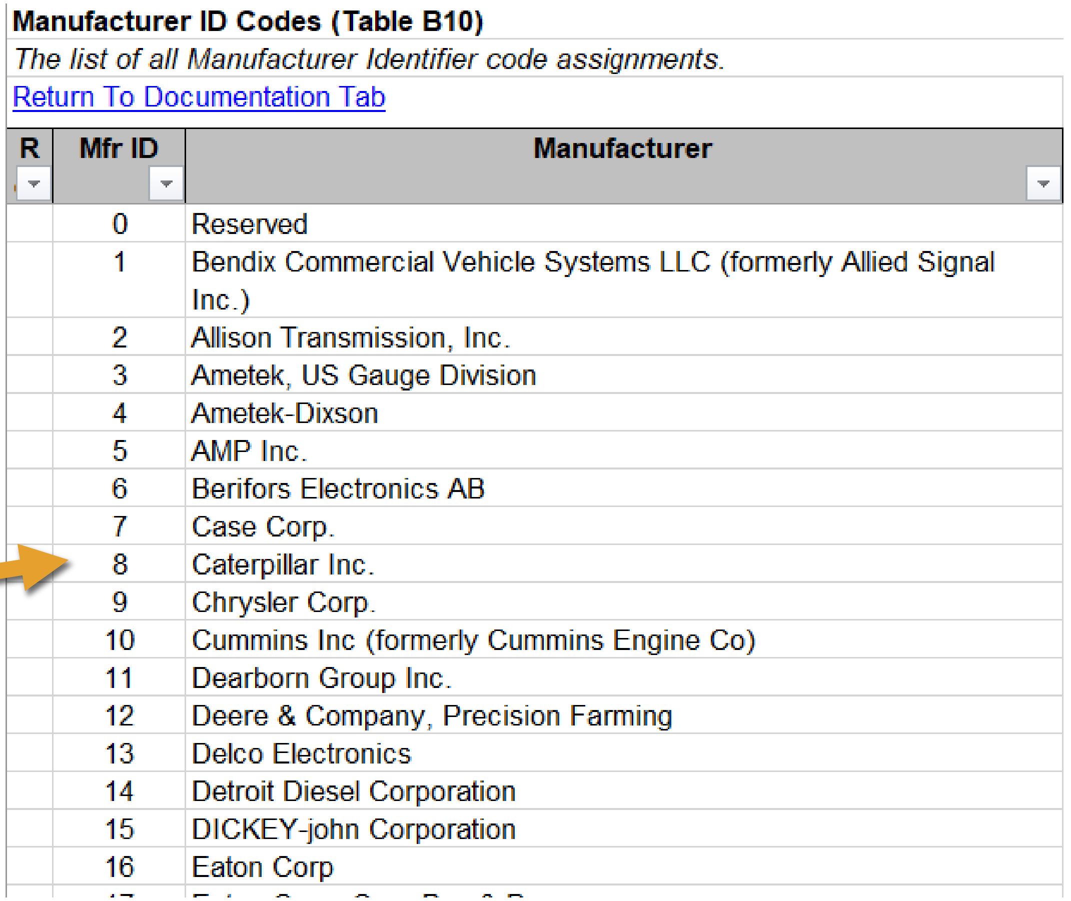
- the vehicle system, function, and function instance are all zero, which is consistent with an engine controller

Byte 4 (0x01), Bits 1-8 = MSB of Mfg Code
Byte 3 (0x01), Bits 8-6 = LSB of Mfg Code

- 0b0000 0001 000~~0 0001~~ = 0b1000 = 8 (dec)

Byte 3 (0x01), bits 1-5 = MSB of Identity Field
Byte 2 (0x6B) = 2nd byte of identity field
Byte 1 (0xD0) = LSB of identity field

- 0b0 0001 0110 1011 1101 0000 = 93,136 (dec)



Manufacturer ID Codes (Table B10)

The list of all Manufacturer Identifier code assignments.

Return To Documentation Tab

R	Mfr ID	Manufacturer
	0	Reserved
	1	Bendix Commercial Vehicle Systems LLC (formerly Allied Signal Inc.)
	2	Allison Transmission, Inc.
	3	Ametek, US Gauge Division
	4	Ametek-Dixson
	5	AMP Inc.
	6	Berifors Electronics AB
	7	Case Corp.
	8	Caterpillar Inc.
	9	Chrysler Corp.
	10	Cummins Inc (formerly Cummins Engine Co)
	11	Dearborn Group Inc.
	12	Deere & Company, Precision Farming
	13	Delco Electronics
	14	Detroit Diesel Corporation
	15	DICKEY-john Corporation
	16	Eaton Corp

Example 2: Detroit Diesel

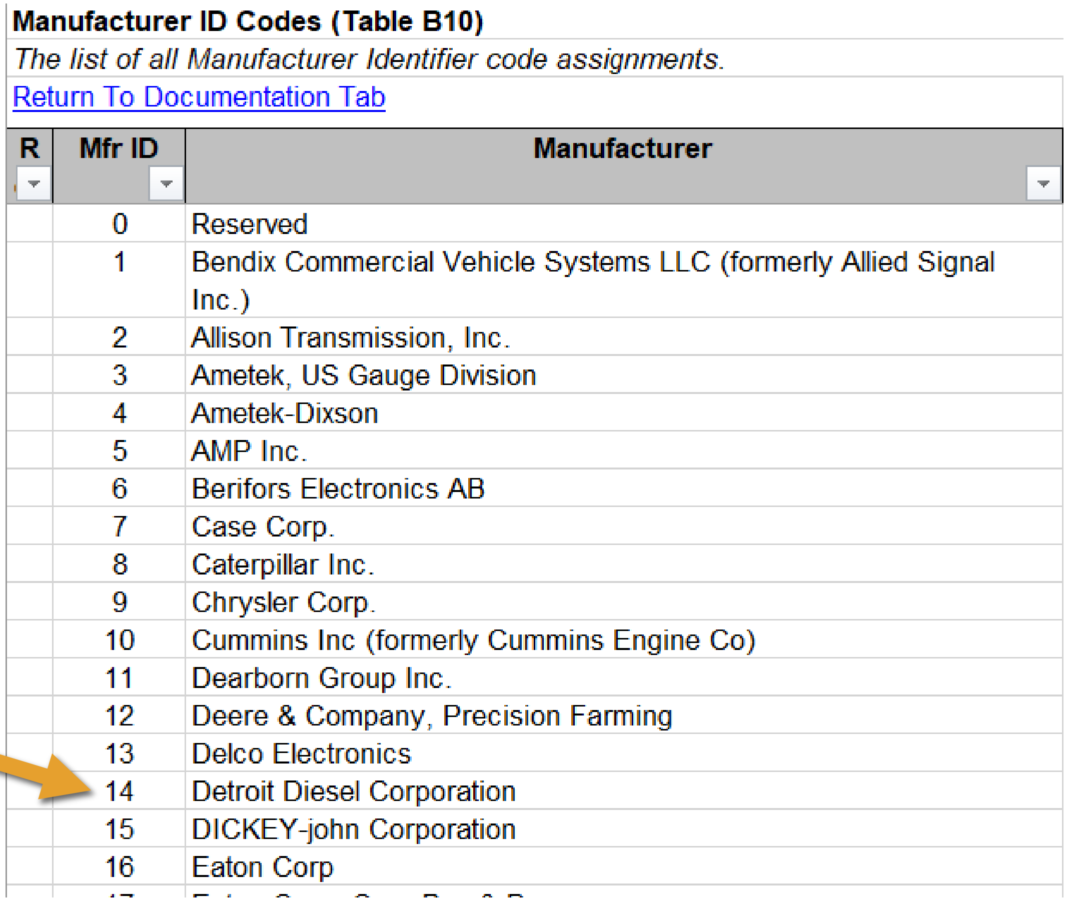
```
can1  18EEFF00   [8] 00 00 C0 01 00 00 00 00
```

Byte 8 (0x00) = 0b0000 0000, which means:
- it is NOT arbitrary address capable,
- the industry group is 0 (global), and
- the vehicle system instance is zero.

Byte 5 -7 (00 00 00), which means:
- the vehicle system, function, and function instance are all zero, which is consistent with an engine controller

Byte 4 (0x01), Bits 1-8 = MSB of Mfg Code
Byte 3 (0xC0), Bits 8-6 = LSB of Mfg Code
- 0b0000 0001 110~~0 0000~~ = 0b1110 = 14 (dec)

Byte 3 (0x01), bits 1-5 = MSB of Identity Field
Byte 2 (0x00) = 2nd byte of identity field
Byte 1 (0x00) = LSB of identity field
- 0b0 0000 0000 0000 0000 0000 (likely not used)


Manufacturer ID Codes (Table B10)
The list of all Manufacturer Identifier code assignments.
Return To Documentation Tab

R	Mfr ID	Manufacturer
	0	Reserved
	1	Bendix Commercial Vehicle Systems LLC (formerly Allied Signal Inc.)
	2	Allison Transmission, Inc.
	3	Ametek, US Gauge Division
	4	Ametek-Dixson
	5	AMP Inc.
	6	Berifors Electronics AB
	7	Case Corp.
	8	Caterpillar Inc.
	9	Chrysler Corp.
	10	Cummins Inc (formerly Cummins Engine Co)
	11	Dearborn Group Inc.
	12	Deere & Company, Precision Farming
	13	Delco Electronics
	14	Detroit Diesel Corporation
	15	DICKEY-john Corporation
	16	Eaton Corp

Example 3: Allison Transmission

```
can1  18EEFF03 [8] 64 00 40 00 00 03 03 10
```

Byte 8 (0x10) = 0b0001 0000, which means:

- it is NOT arbitrary address capable,
- the industry group is 1 (on-highway), and
- the vehicle system instance is zero.

Byte 7 (0x03), the vehicle system is the transmission

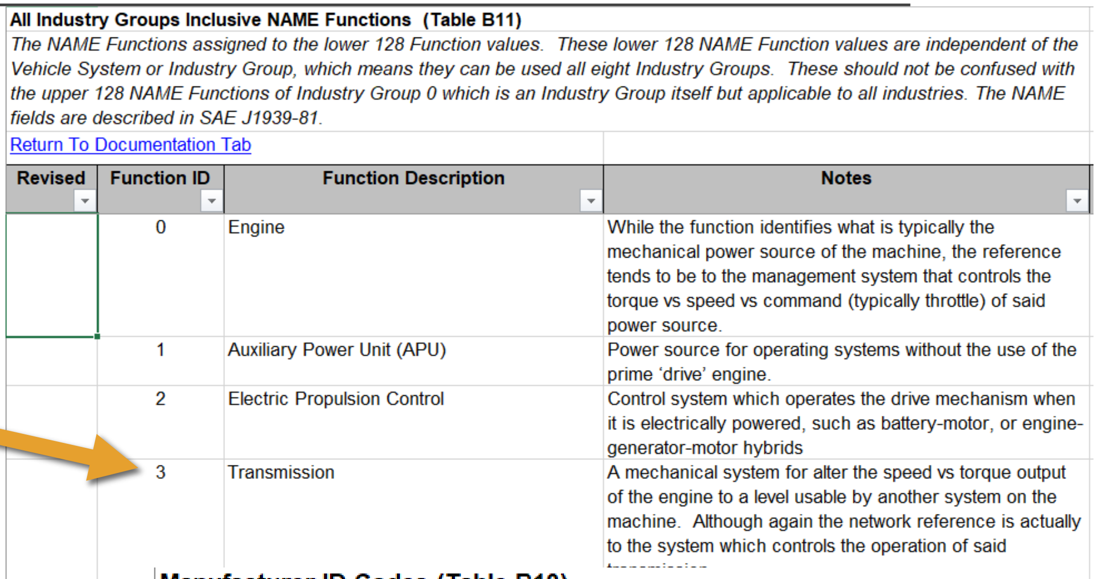
Byte 6 (0x03), function is the transmission

Byte 5 (0x00), the function and ECU instance is zero, which means it's the first instance.

Byte 4 (0x00), Bits 1-8 = MSB of Mfg Code
Byte 3 (0x40), Bits 8-6 = LSB of Mfg Code

- 0b0000 0000 010~~0 0000~~ = 0b0010 = 2 (dec)

Bytes 3-1 (0x00064) comprise the identity field

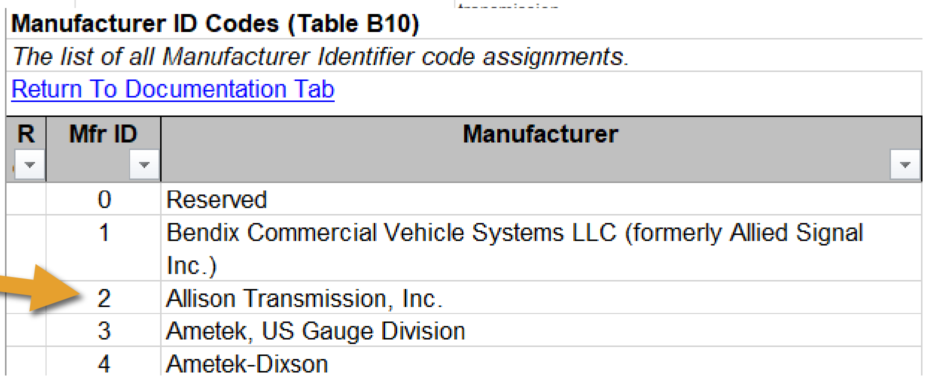


All Industry Groups Inclusive NAME Functions (Table B11)

The NAME Functions assigned to the lower 128 Function values. These lower 128 NAME Function values are independent of the Vehicle System or Industry Group, which means they can be used all eight Industry Groups. These should not be confused with the upper 128 NAME Functions of Industry Group 0 which is an Industry Group itself but applicable to all industries. The NAME fields are described in SAE J1939-81.

Return To Documentation Tab

Revised	Function ID	Function Description	Notes
	0	Engine	While the function identifies what is typically the mechanical power source of the machine, the reference tends to be to the management system that controls the torque vs speed vs command (typically throttle) of said power source.
	1	Auxiliary Power Unit (APU)	Power source for operating systems without the use of the prime 'drive' engine.
	2	Electric Propulsion Control	Control system which operates the drive mechanism when it is electrically powered, such as battery-motor, or engine-generator-motor hybrids
	3	Transmission	A mechanical system for alter the speed vs torque output of the engine to a level usable by another system on the machine. Although again the network reference is actually to the system which controls the operation of said



Manufacturer ID Codes (Table B10)

The list of all Manufacturer Identifier code assignments.

Return To Documentation Tab

R	Mfr ID	Manufacturer
	0	Reserved
	1	Bendix Commercial Vehicle Systems LLC (formerly Allied Signal Inc.)
	2	Allison Transmission, Inc.
	3	Ametek, US Gauge Division
	4	Ametek-Dixson

Address Claim Attack

Idea: Claim someone else's address with a higher priority address (All Zeros).

Keep claiming addresses as they are dynamically claimed.

If a system can't find a claimable address, then it should stop broadcasting (Denial of Service)

The following example shows how to conduct an address claim attack:

https://github.com/SystemsCyber/CyberTruckResources/blob/master/05_J1939/06%20J1939%20Address%20Claim.ipynb

- Note: This runs on Linux Socket CAN
- Try it on the CM2350

Run these commands in Ubuntu:

```
git clone https://github.com/SystemsCyber/CyberTruckResources.git
conda activate base
jupyter notebook
```

J1939 Diagnostic Messages

UNDERSTANDING MESSAGES RELATED TO FAULT CODES

J1939-73 Application Layer - Diagnostics

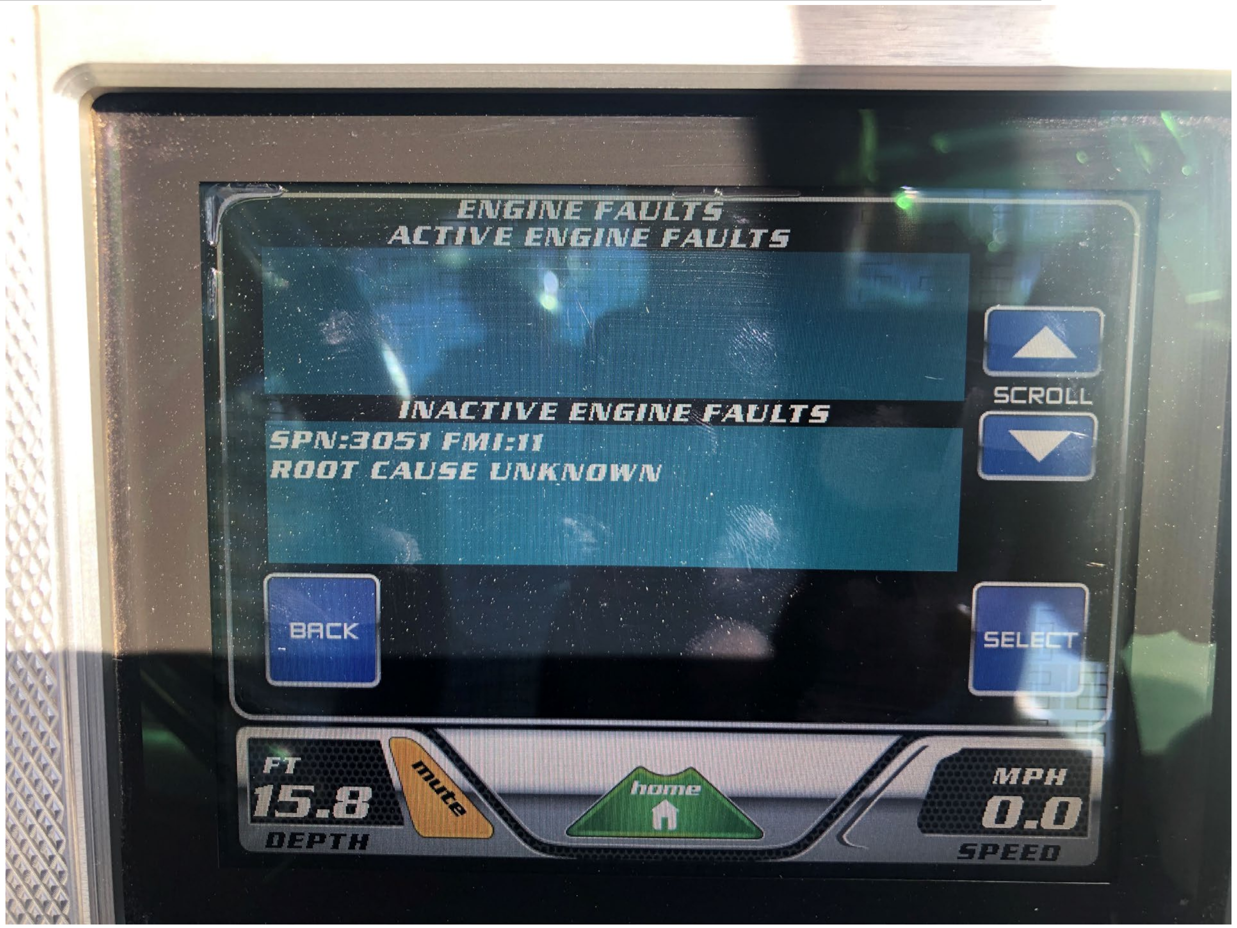
Defines close to 60 Diagnostic Messages related to troubleshooting and monitoring components on a truck.

Defines lamp status
- Check Engine Lamp
- Malfunction Indicator Lamp MIL

Defines Failure Mode Indicators (FMI)

All trucks use some parts of J1939-73
- Diagnostic Message 1

Many parts of J1939-73 are not used
- Components use UDS or proprietary messaging
- Most concepts are implemented in ECUs in some fashion


ENGINE FAULTS
ACTIVE ENGINE FAULTS
INACTIVE ENGINE FAULTS
SPN:3051 FMI:11
ROOT CAUSE UNKNOWN

Diagnostic Message 1 Example (No Fault Codes)

Broadcast once per second

Diagnostic trouble codes (DTCs) have four fields that use 32-bits:

- Suspect Parameter Number (SPN): 19 bits
- Failure Mode Identifier (FMI): 5 bits
- Occurrence Count (OC): 7 bits
- SPN Conversion Method (CM): 1 bit

PGN for DM1 is 65226 (0xFECA)

- Broadcast message with global destination
- Source Address tell which controller application is broadcasting

Unused bytes should be set to 0xFF

```
18FECA03 03 FF 00 00 00 00 FF FF
```

```
18 – Priority (6 default)
FECA – DM1 PGN (65226)
03 – Source Address (Transmission)

03 – Lamp Status (0000 0011)
FF – Lamp Flash Status (1111 1111)
00 00 0 – Suspect Parameter Number
0 – Failure Mode Indicator (FMI)
00 – Conversion and Occurrence Count
```

Diagnostic Trouble Code																															
Least Significant Byte of SPN								Second Byte of SPN								3 most significant bits for SPN, 5 bits for FMI								Conversion Method and Occurance Count							
Suspect Parameter Number (19 bits)																			FMI					CM	OC						
8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1

Lamp Status Bytes in DM1

DM1 Byte 1:
- Bits 8-7: Malfunction indicator lamp
- Bits 6-5: Red stop lamp
- Bits 4-3: Amber warning lamp
- Bits 2-1: Protect lamp

Emissions Related

DM1 Byte 2:
- Bits 8-7: Flash malfunction indicator lamp
- Bits 6-5: Flash red stop lamp
- Bits 4-3: Flash amber warning lamp
- Bits 2-1: Flash protect lamp

Lamp status bytes exist only at the beginning of the DM1 message

Multiple DTCs can be concatenated and broadcast

Lamp Status Bits:
00 = Off
01 = On
10 = Error
11 = Not Available (unused)

Flash Status Bits:
00 = Slow Flash (1/sec)
01 = Fast Flash (2/sec)
10 = Reserved
11 = Unavailable (don't flash)

Diagnostic Message 1
Ex: Multiple Fault Codes

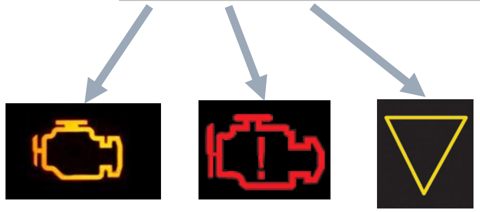
SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

TRANSPORT PROTOCOL MESSAGES

```
1CECFF00 20 CA 00 1D FF CA FE 00
1CEBFF00 01 57 FF 9D 00 03 01 FB
1CEBFF00 02 06 0B 32 4A 00 0E 31
 …(182 more bytes in 26 frames)…
1CEBFF00 1D 09 01 84 06 09 01 FF
```

Last Frame

```
57 FF – Lamp Status
(0101 0111 1111 1111)
```



- All 3 lamps are on
- Protect is not used
- No lamps are flashing

J1939 PROTOCOL DATA UNIT

```
PGN: 0x00FECA = 65226 (DM1)
Dest. Address: 0xFF (Global)
Source Address: 0x00 (Engine #1)


Data: (0x00CA = 202 bytes)

57 FF 9D 00 03 01 FB 06 0B 32 4A
00 0E 31 … 09 01 84 06 09 01
```

Diagnostic Message 1
Ex: Multiple Fault Codes

57 FF 9D 00 03 01 FB 06 0B 32 4A 00 0E 31 … 09 01 84 06 09 01

Use SAE J1939-73 Appendix A for FMIs

DTC 1: 9D 00 03 01

Diagnostic Trouble Code																															
Least Significant Byte of SPN								Second Byte of SPN								3 most significant bits for SPN, 5 bits for FMI								Conversion Method and Occurrence Count							
9D								00								03								01							
Suspect Parameter Number (19 bits)																			FMI					CM	OC						
8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
1	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1
Reverse Byte order: 0x0009D = 157																			3					0	1						

- SPN = 157 (Engine Fuel 1 Injector Metering Rail 1 Pressure)
- FMI = 3 (Voltage Above Normal, or Shorted to High Source)
- Count = 1

DTC 2: FB 06 0B 32

Diagnostic Trouble Code																															
Least Significant Byte of SPN								Second Byte of SPN								3 most significant bits for SPN, 5 bits for FMI								Conversion Method and Occurrence Count							
FB								06								0B								32							
Suspect Parameter Number (19 bits)																			FMI					CM	OC						
8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
1	1	1	1	1	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	1	0
Reverse Byte order: 0x006FB = 1787																			11					0	50						

- SPN = 1787 (Engine Torque Limit Request - Maximum Continuos)
- FMI = 11 (Root Cause Not Known)
- Count = 50

Most diagnostic service tools interpret these codes.

J1939 Diagnostics Summary

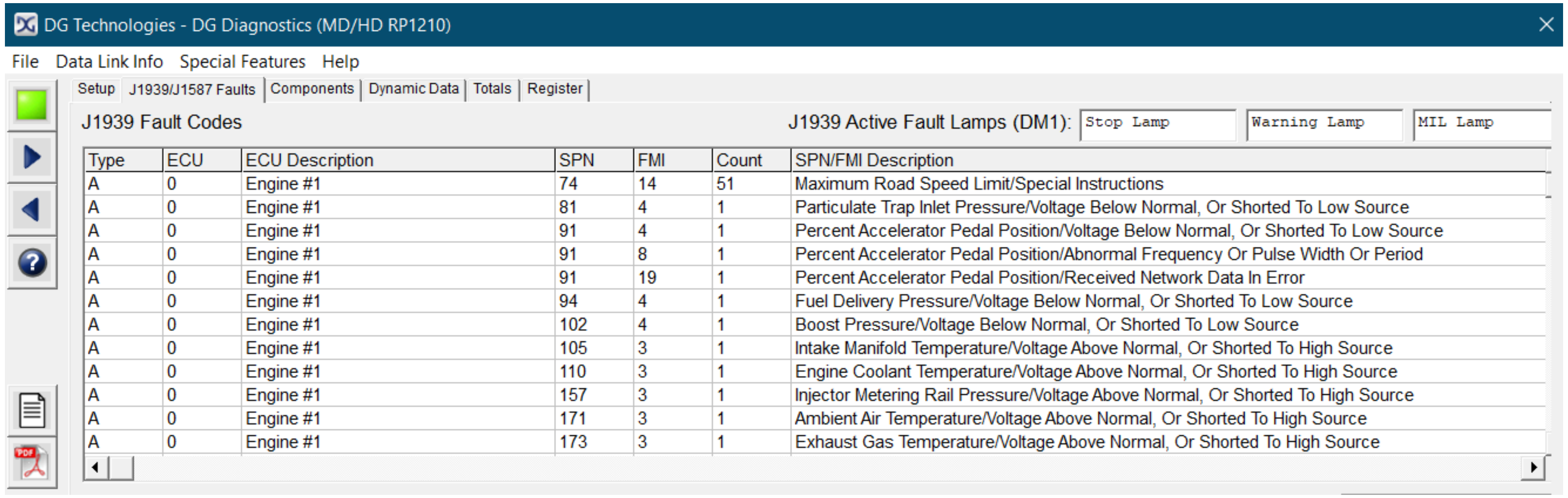
Diagnostic Trouble Codes are defined with

- Suspect Parameter Number (i.e. the potential part that may be broken)
- Failure Mode Indicator (i.e. the symptom of the broken part)
- Occurrence Count (how many times the system sees the indication)

J1939-73 also discusses

- Firmware updates
- Memory access
- Emissions Compliance
- Freeze Frame Data
- Data Security

Many vehicles implement only a small portion of J1939-73



DG Technologies - DG Diagnostics (MD/HD RP1210)
File Data Link Info Special Features Help
Setup J1939/J1587 Faults Components Dynamic Data Totals Register

J1939 Fault Codes

J1939 Active Fault Lamps (DM1): Stop Lamp Warning Lamp MIL Lamp

Type	ECU	ECU Description	SPN	FMI	Count	SPN/FMI Description
A	0	Engine #1	74	14	51	Maximum Road Speed Limit/Special Instructions
A	0	Engine #1	81	4	1	Particulate Trap Inlet Pressure/Voltage Below Normal, Or Shorted To Low Source
A	0	Engine #1	91	4	1	Percent Accelerator Pedal Position/Voltage Below Normal, Or Shorted To Low Source
A	0	Engine #1	91	8	1	Percent Accelerator Pedal Position/Abnormal Frequency Or Pulse Width Or Period
A	0	Engine #1	91	19	1	Percent Accelerator Pedal Position/Received Network Data In Error
A	0	Engine #1	94	4	1	Fuel Delivery Pressure/Voltage Below Normal, Or Shorted To Low Source
A	0	Engine #1	102	4	1	Boost Pressure/Voltage Below Normal, Or Shorted To Low Source
A	0	Engine #1	105	3	1	Intake Manifold Temperature/Voltage Above Normal, Or Shorted To High Source
A	0	Engine #1	110	3	1	Engine Coolant Temperature/Voltage Above Normal, Or Shorted To High Source
A	0	Engine #1	157	3	1	Injector Metering Rail Pressure/Voltage Above Normal, Or Shorted To High Source
A	0	Engine #1	171	3	1	Ambient Air Temperature/Voltage Above Normal, Or Shorted To High Source
A	0	Engine #1	173	3	1	Exhaust Gas Temperature/Voltage Above Normal, Or Shorted To High Source

RP1210 Programming

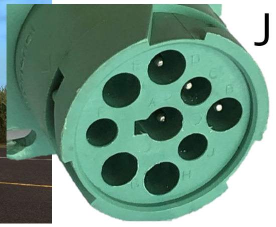
USING WINDOWS DIAGNOSTIC SERVICE TOOLS TO CONNECT TO THE NETWORK

(ONE PERSON PER TABLE SHOULD BOOT THEIR COMPUTER TO WINDOWS NOW)

RP1210 Vehicle Diagnostics Adapters

Truck owners want only one hardware device to work with all their ECUs and diagnostics software

American Trucking Association (ATA) and their Technology Maintenance Council (TMC) published Recommended Practice (RP) number 1210 to define a Windows API for vehicle diagnostic adapters (VDAs)



J1939

Nexiq USB Link 2

DG DPA 5 Pro

Noregon DLA +

USB
Bluetooth
Wifi

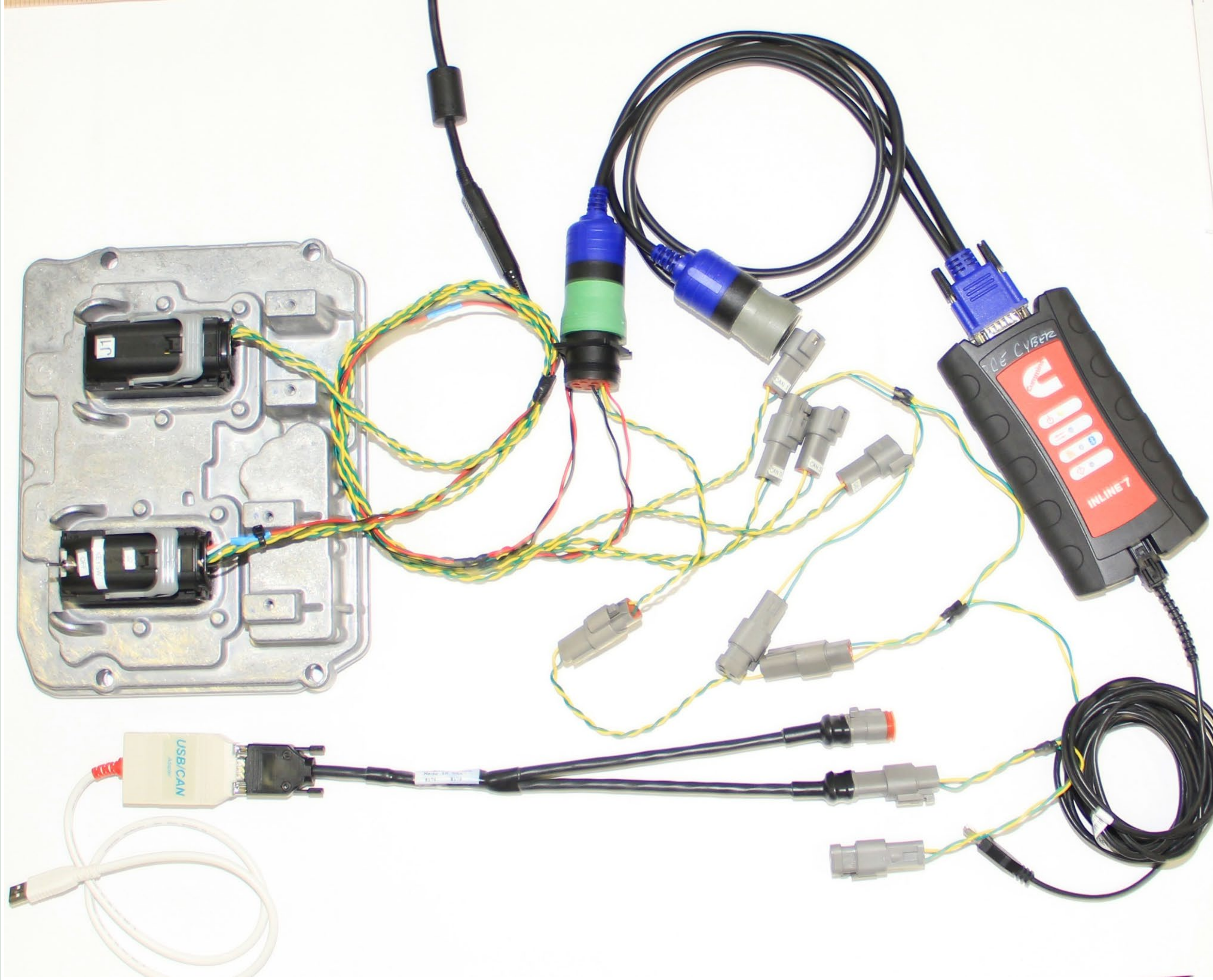
Service Computer

Setup the ECM Kit for RP1210

Connect CAN0 to the Diagnostic Port.

Be sure to connect the terminating resistors.

Either the PeakCAN Adapter or the USB2CAN devices will work.



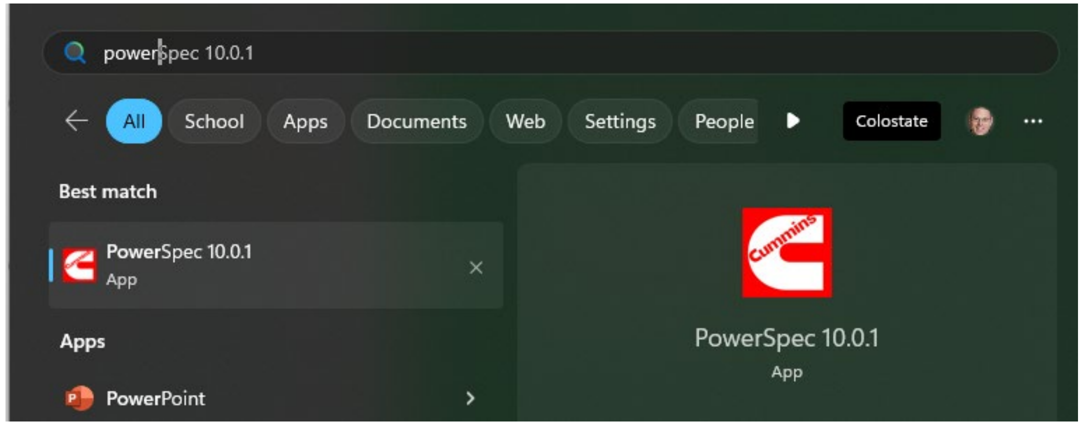
In Class Exercise: Cummins PowerSpec

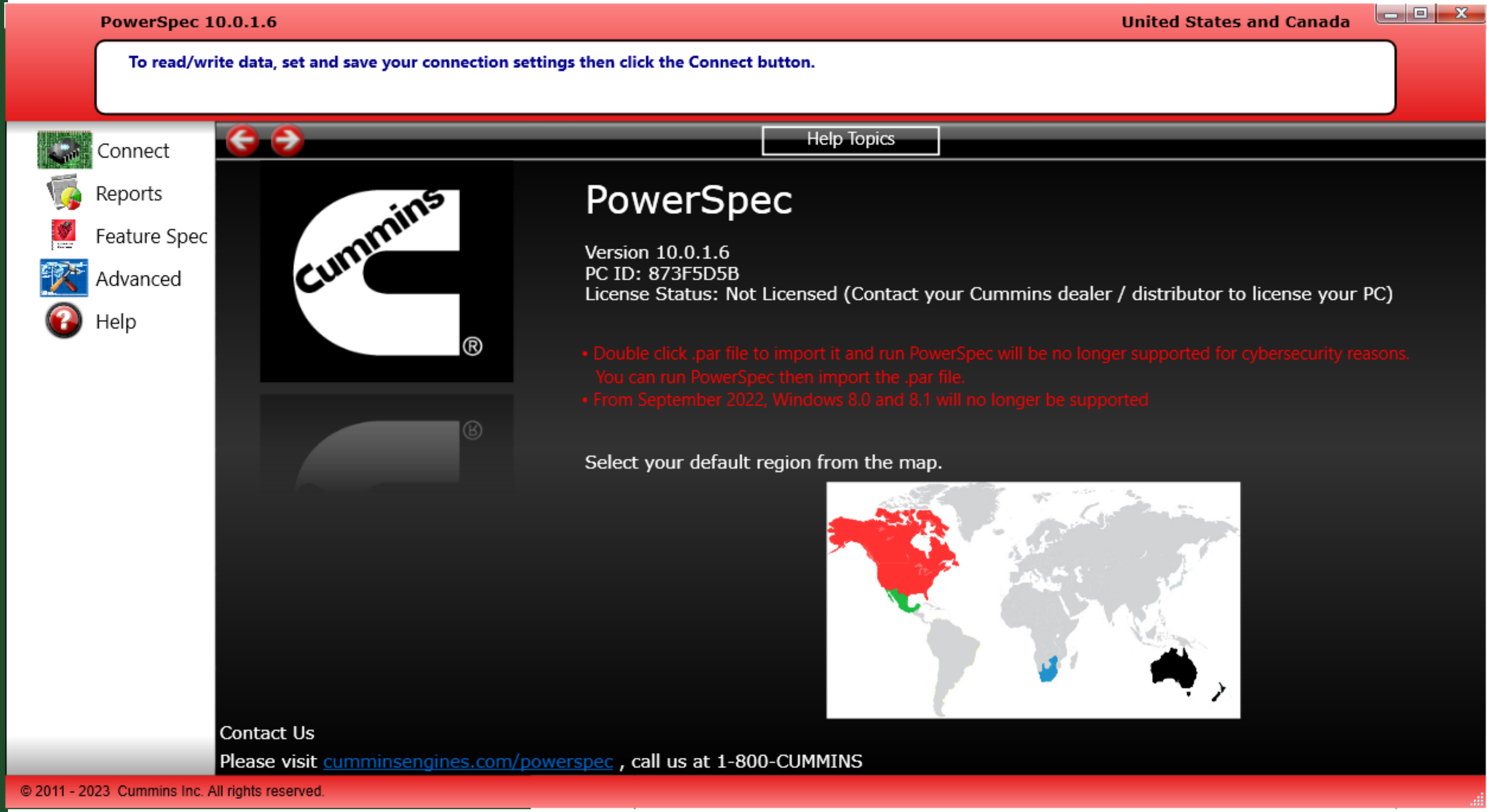
Open Cummins PowerSpec

Connect to the Engine Controller

Read the Dataplate and determine the VIN

Look for the VIN in the J1939 traffic




PowerSpec 10.0.1.6

United States and Canada

To read/write data, set and save your connection settings then click the Connect button.

PowerSpec

Version 10.0.1.6
PC ID: 873F5D5B
License Status: Not Licensed (Contact your Cummins dealer / distributor to license your PC)

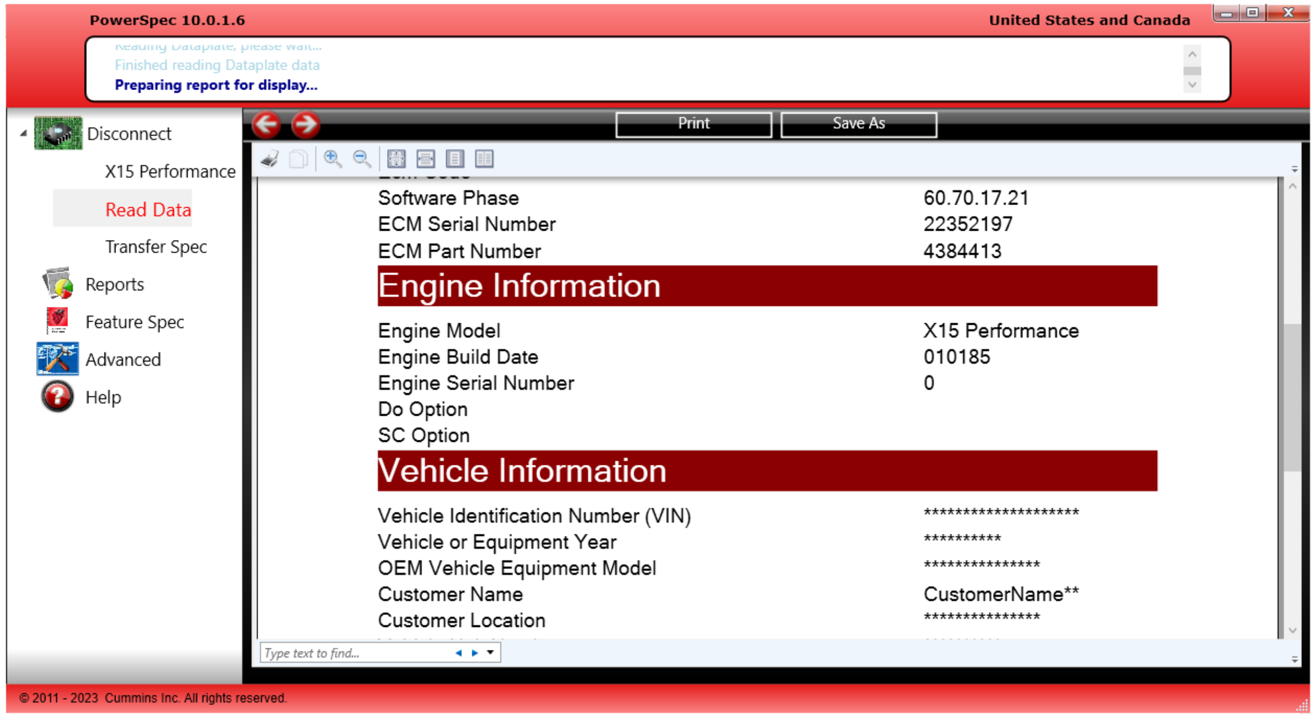
- Double click .par file to import it and run PowerSpec will be no longer supported for cybersecurity reasons. You can run PowerSpec then import the .par file.
- From September 2022, Windows 8.0 and 8.1 will no longer be supported

Select your default region from the map.

Contact Us
Please visit cumminsengines.com/powerspec , call us at 1-800-CUMMINS

© 2011 - 2023 Cummins Inc. All rights reserved.

PowerSpec Dataplate



Reading Dataplate, please wait...
Finished reading Dataplate data
Preparing report for display...

Software Phase	60.70.17.21
ECM Serial Number	22352197
ECM Part Number	4384413

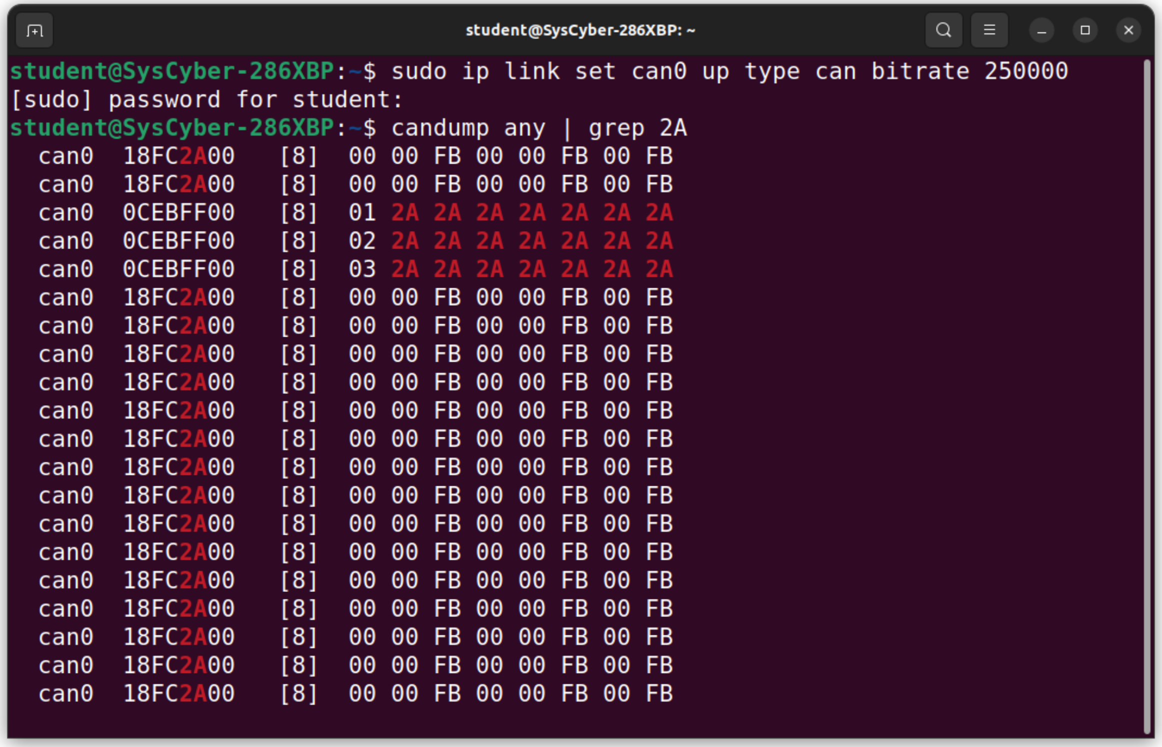
Engine Information

Engine Model	X15 Performance
Engine Build Date	010185
Engine Serial Number	0
Do Option	
SC Option	

Vehicle Information

Vehicle Identification Number (VIN)	********************
Vehicle or Equipment Year	**********
OEM Vehicle Equipment Model	***************
Customer Name	CustomerName**
Customer Location	***************

© 2011 - 2023 Cummins Inc. All rights reserved.



student@SysCyber-286XBP: ~

```
student@SysCyber-286XBP:~$ sudo ip link set can0 up type can bitrate 250000
[sudo] password for student:
student@SysCyber-286XBP:~$ candump any | grep 2A
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
  can0  0CEBFF00   [8]  01 2A 2A 2A 2A 2A 2A 2A
  can0  0CEBFF00   [8]  02 2A 2A 2A 2A 2A 2A 2A
  can0  0CEBFF00   [8]  03 2A 2A 2A 2A 2A 2A 2A
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
  can0  18FC2A00   [8]  00 00 FB 00 00 FB 00 FB
```

ASCII for * is 0x2A

SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

Writing RP1210 Applications – SimpleRP1210

SimpleRP1210 written in C and compiled with Visual Studio Tools

Clone this repository in Windows:

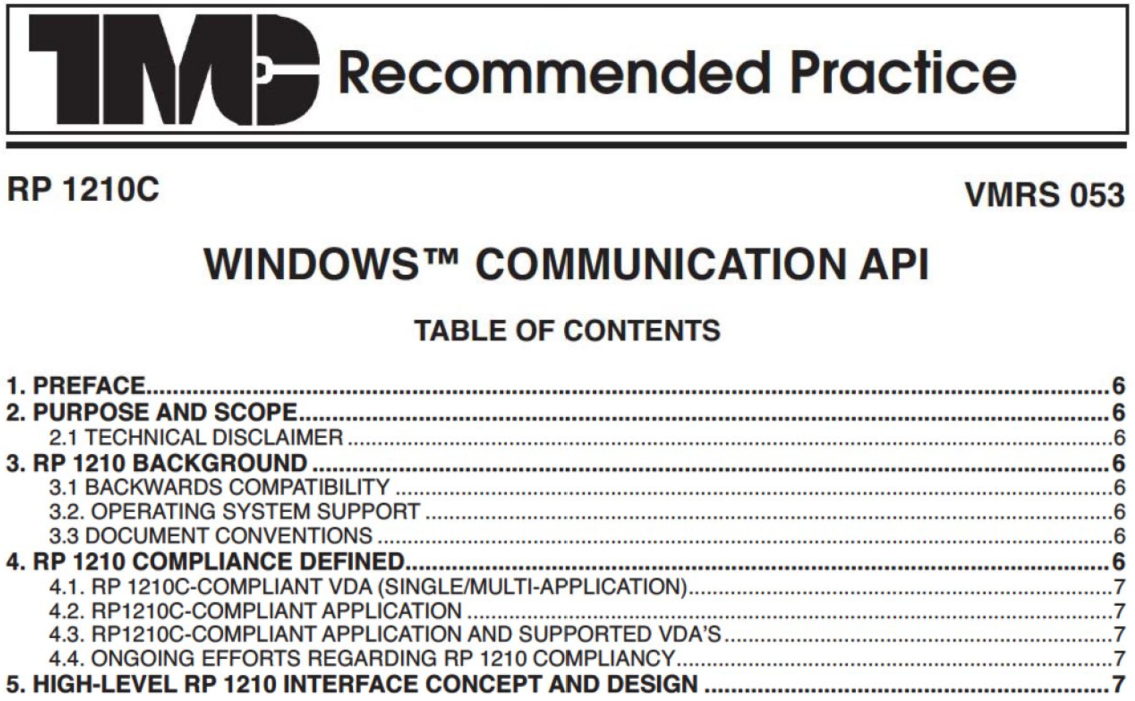
https://github.com/SystemsCyber/ShimDLL

Examine

- simpleRP1210.c
- simpleRP1210.h

Follow the instructions to compile the C code

- From the developer command prompt:
  `cl.exe simpleRP1210.c`
- Execute:
  `simpleRP1210.exe CIL7R32.dll 1`



TMC Recommended Practice

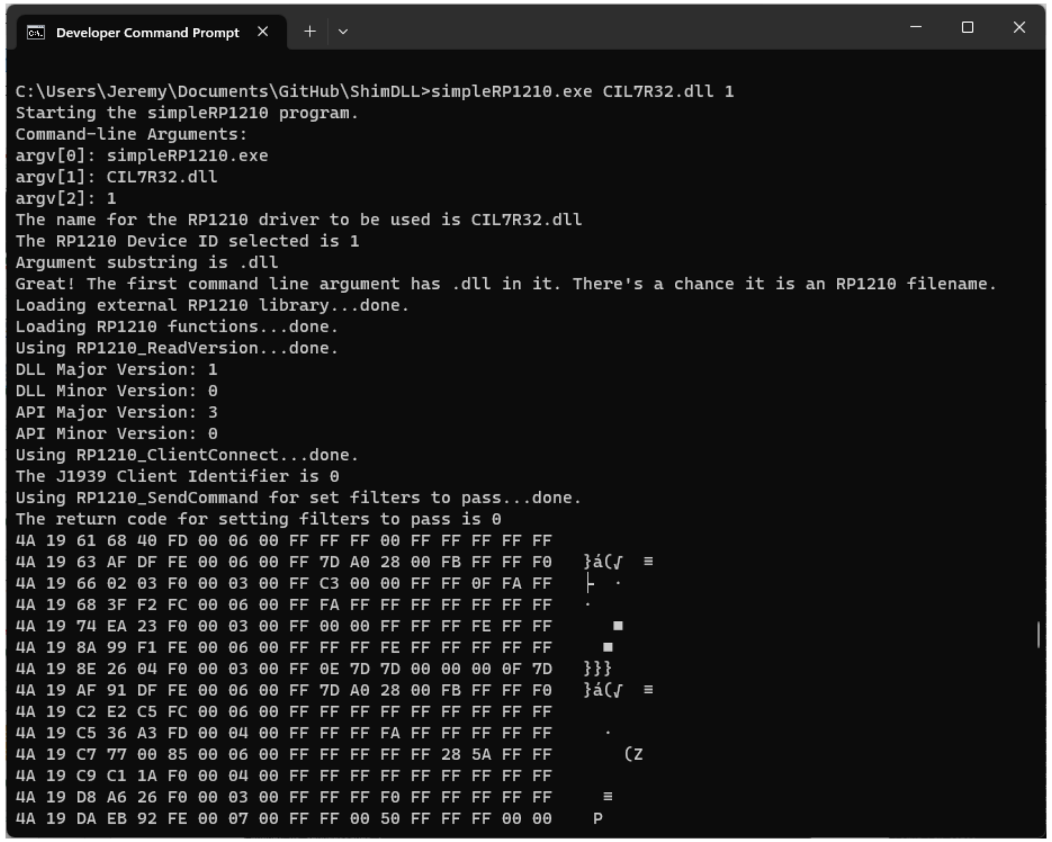
RP 1210C VMRS 053

WINDOWS™ COMMUNICATION API

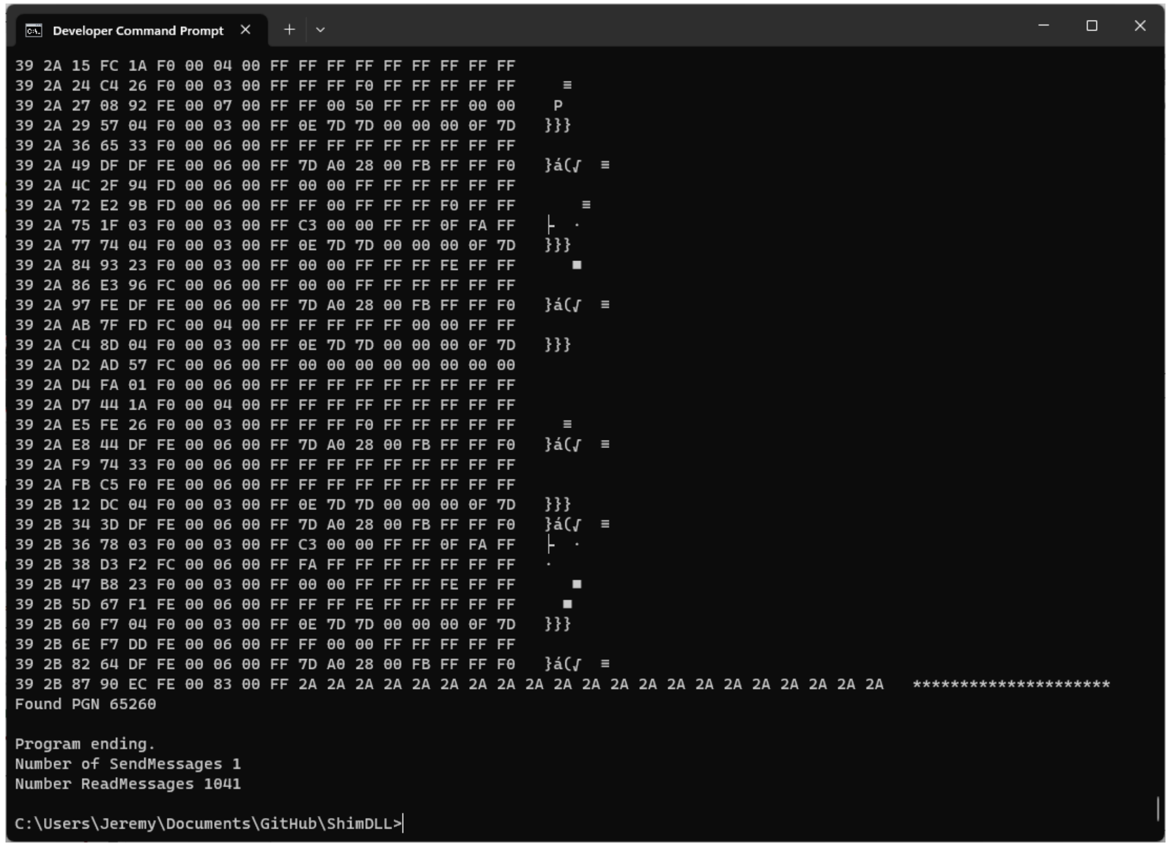
TABLE OF CONTENTS

Exercise: Run SimpleRP1210.exe

`simpleRP1210.exe CIL7R32.dll 1`



```
C:\Users\Jeremy\Documents\GitHub\ShimDLL>simpleRP1210.exe CIL7R32.dll 1
Starting the simpleRP1210 program.
Command-line Arguments:
argv[0]: simpleRP1210.exe
argv[1]: CIL7R32.dll
argv[2]: 1
The name for the RP1210 driver to be used is CIL7R32.dll
The RP1210 Device ID selected is 1
Argument substring is .dll
Great! The first command line argument has .dll in it. There's a chance it is an RP1210 filename.
Loading external RP1210 library...done.
Loading RP1210 functions...done.
Using RP1210_ReadVersion...done.
DLL Major Version: 1
DLL Minor Version: 0
API Major Version: 3
API Minor Version: 0
Using RP1210_ClientConnect...done.
The J1939 Client Identifier is 0
Using RP1210_SendCommand for set filters to pass...done.
The return code for setting filters to pass is 0
4A 19 61 68 40 FD 00 06 00 FF FF FF 00 FF FF FF FF FF
4A 19 63 AF DF FE 00 06 00 FF 7D A0 28 00 FB FF FF F0   }á(√  ≡
4A 19 66 02 03 F0 00 03 00 FF C3 00 00 FF FF 0F FA FF   ├  ·
4A 19 68 3F F2 FC 00 06 00 FF FA FF FF FF FF FF FF FF   ·
4A 19 74 EA 23 F0 00 03 00 FF 00 00 FF FF FF FE FF FF      ■
4A 19 8A 99 F1 FE 00 06 00 FF FF FF FE FF FF FF FF FF     ■
4A 19 8E 26 04 F0 00 03 00 FF 0E 7D 7D 00 00 00 0F 7D   }}}
4A 19 AF 91 DF FE 00 06 00 FF 7D A0 28 00 FB FF FF F0   }á(√  ≡
4A 19 C2 E2 C5 FC 00 06 00 FF FF FF FF FF FF FF FF FF
4A 19 C5 36 A3 FD 00 04 00 FF FF FF FA FF FF FF FF FF     ·
4A 19 C7 77 00 85 00 06 00 FF FF FF FF FF 28 5A FF FF       (Z
4A 19 C9 C1 1A F0 00 04 00 FF FF FF FF FF FF FF FF FF
4A 19 D8 A6 26 F0 00 03 00 FF FF FF F0 FF FF FF FF FF     ≡
4A 19 DA EB 92 FE 00 07 00 FF FF 00 50 FF FF FF 00 00    P
```



```
39 2A 15 FC 1A F0 00 04 00 FF FF FF FF FF FF FF FF FF
39 2A 24 C4 26 F0 00 03 00 FF FF FF F0 FF FF FF FF FF     ≡
39 2A 27 08 92 FE 00 07 00 FF FF 00 50 FF FF FF 00 00    P
39 2A 29 57 04 F0 00 03 00 FF 0E 7D 7D 00 00 00 0F 7D   }}}
39 2A 36 65 33 F0 00 06 00 FF FF FF FF FF FF FF FF FF
39 2A 49 DF DF FE 00 06 00 FF 7D A0 28 00 FB FF FF F0   }á(√  ≡
39 2A 4C 2F 94 FD 00 06 00 FF 00 00 FF FF FF FF FF FF
39 2A 72 E2 9B FD 00 06 00 FF FF 00 FF FF FF F0 FF FF       ≡
39 2A 75 1F 03 F0 00 03 00 FF C3 00 00 FF FF 0F FA FF   ├  ·
39 2A 77 74 04 F0 00 03 00 FF 0E 7D 7D 00 00 00 0F 7D   }}}
39 2A 84 93 23 F0 00 03 00 FF 00 00 FF FF FF FE FF FF      ■
39 2A 86 E3 96 FC 00 06 00 FF 00 00 FF FF FF FF FF FF
39 2A 97 FE DF FE 00 06 00 FF 7D A0 28 00 FB FF FF F0   }á(√  ≡
39 2A AB 7F FD FC 00 04 00 FF FF FF FF FF 00 00 FF FF
39 2A C4 8D 04 F0 00 03 00 FF 0E 7D 7D 00 00 00 0F 7D   }}}
39 2A D2 AD 57 FC 00 06 00 FF 00 00 00 00 00 00 00 00
39 2A D4 FA 01 F0 00 06 00 FF FF FF FF FF FF FF FF FF
39 2A D7 44 1A F0 00 04 00 FF FF FF FF FF FF FF FF FF
39 2A E5 FE 26 F0 00 03 00 FF FF FF F0 FF FF FF FF FF     ≡
39 2A E8 44 DF FE 00 06 00 FF 7D A0 28 00 FB FF FF F0   }á(√  ≡
39 2A F9 74 33 F0 00 06 00 FF FF FF FF FF FF FF FF FF
39 2A FB C5 F0 FE 00 06 00 FF FF FF FF FF FF FF FF FF
39 2B 12 DC 04 F0 00 03 00 FF 0E 7D 7D 00 00 00 0F 7D   }}}
39 2B 34 3D DF FE 00 06 00 FF 7D A0 28 00 FB FF FF F0   }á(√  ≡
39 2B 36 78 03 F0 00 03 00 FF C3 00 00 FF FF 0F FA FF   ├  ·
39 2B 38 D3 F2 FC 00 06 00 FF FA FF FF FF FF FF FF FF   ·
39 2B 47 B8 23 F0 00 03 00 FF 00 00 FF FF FF FE FF FF      ■
39 2B 5D 67 F1 FE 00 06 00 FF FF FF FE FF FF FF FF FF     ■
39 2B 60 F7 04 F0 00 03 00 FF 0E 7D 7D 00 00 00 0F 7D   }}}
39 2B 6E F7 DD FE 00 06 00 FF FF 00 00 FF FF FF FF FF
39 2B 82 64 DF FE 00 06 00 FF 7D A0 28 00 FB FF FF F0   }á(√  ≡
39 2B 87 90 EC FE 00 83 00 FF 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A   *********************
Found PGN 65260

Program ending.
Number of SendMessages 1
Number ReadMessages 1041

C:\Users\Jeremy\Documents\GitHub\ShimDLL>
```

Function Prototypes

Function Name	Description
RP1210_ClientConnect (…)	Load the routines for a particular protocol on the correct channel
RP1210_SendCommand(…)	Send command to change the behavior or property of the VDA
RP1210_SendMessage (…)	Send a message through the VDA to the vehicle network
RP1210_ReadMessage (…)	Read a message from the vehicle network
RP1210_ClientDisconnect (…)	Disconnect the client and close the driver

The message structure depends on the type of client

- J1939
- CAN
- J1708

RP1210 Log files are helpful to understand diagnostic communication.

https://www.atabusinesssolutions.com/Shopping/Product/viewproduct/2675472/TMC%20Individual%20RPs

RP1210 Log File Example DG Technologies DPA5

```
FT:0011,AT:0030  XX,CC,00,02,001,J1939:Baud=Auto,0,0,0
Exe Name: C:\PROGRAM FILES (X86)\CUMMINS\POWERSPEC5\POWERSPEC.EXE :Thu Jan 13 13:05:10 2022
FT:0000,AT:0000  02,SC,00,17,45,35,30,30,30,30,30,00,00,00,00,00,00,00,00,00,00,00
FT:0000,AT:0000  02,SC,00,1,18,00
FT:0001,AT:0001  02,SC,00,7,4,0d,00,ef,00,ff,00,fa
FT:0000,AT:0000  02,SC,00,1,18,01
FT:0252,AT:0252  02,SC,00,10,19,fa,d6,eb,56,01,00,81,00,00,00
FT:0000,AT:0000  02,SM,00,15,0,0,00,ef,00,06,fa,00,81,02,01,01,ff,01,ff,00,00,
FT:0265,AT:0013  02,RM,19,4104,1,00,01,9b,da,00,ef,00,00,00,fa,81,01,02,00,01,70,01,05,30,
FT:0000,AT:0000  02,SM,00,15,0,0,00,ef,00,06,fa,00,81,00,03,00,00,01,01,00,00,
FT:0027,AT:0027  02,RM,23,4104,1,00,01,9b,f3,00,ef,00,00,00,fa,81,00,03,01,01,01,02,01,00,7f,04,28,31,
```

CC – Client Connect
SC – Send Command

SM - Send Message
RM – Read Message

RP1210 Client Connect

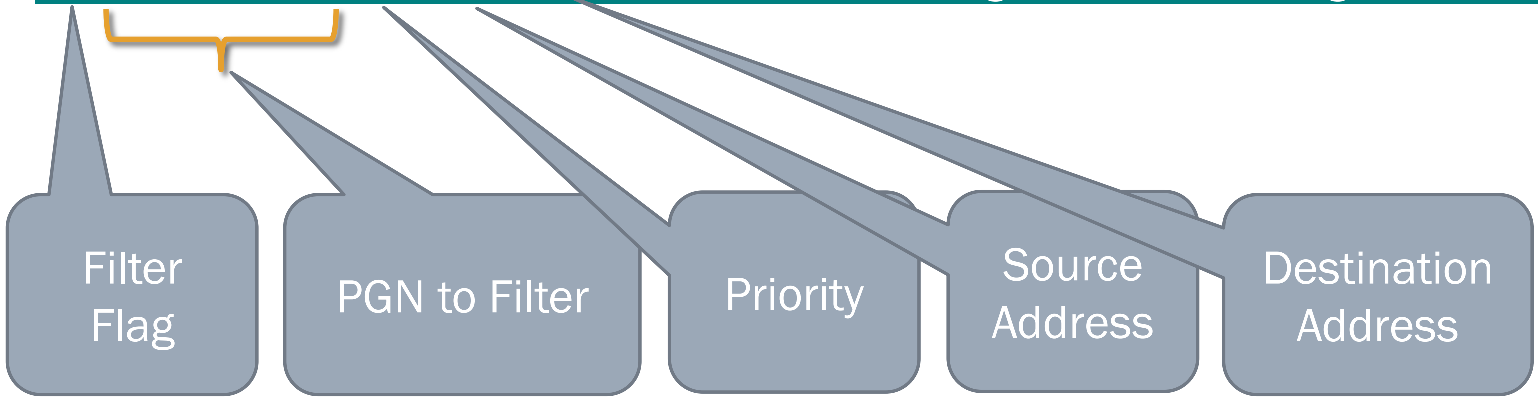
```
FT:0011,AT:0030  XX,CC,00,02,001,J1939:Baud=Auto,0,0,0
Exe Name: C:\PROGRAM FILES (X86)\CUMMINS\POWERSPEC5\POWERSPEC.EXE :Thu Jan 13 13:05:10 2022
```

```
CC – Client Connect
00 – Legacy Window Handle (always 0x00)
02 – Returned Client ID (Used by all the other commands)
001 – Device ID (Selected from the RP1210 INI files for the device connected)
J1939:Baud=Auto – Protocol String
0 – TX Buffer Size (Set to zero to accept default of 8k)
0 – RX Buffer Size (Set to zero to accept default of 8k)
0 – IsAppPacketizingIncomingMsg (set to zero to have the VDA do J1939 transport protocol)
```

RP1210 Send Command

```
FT:0001,AT:0001   02,SC,00,7,4,0d,00,ef,00,ff,00,fa

02 – Connected Client (Result from Client Connect)
SC – Send Command
00 – Return value from Command (00 = Success for Command 4)
4 – Command Number (4 = Set Message Filtering for J1939)
7 – Message Size for Command
0d,00,ef,00,ff,00,fa – Command message for setting J1939 Filter
```



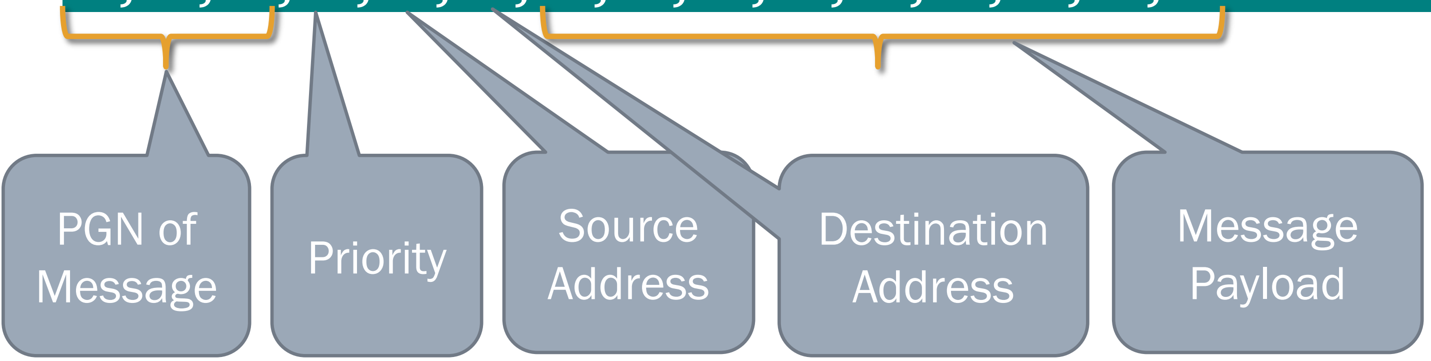
Each command has different meanings for the parameters.

See the RP1210 Document for details.

RP1210 Send Message

```
FT:0000,AT:0000   02,SM,00,15,0,0,00,ef,00,06,fa,00,81,02,01,01,ff,01,ff,00,00,
```

```
02 – Connected Client (Result from Client Connect)
SM - Send Message
00 – Return value (00 = Successfully sent message)
15 – Message Size including identifiers and messages
0 – Legacy Notify Status on Transmit (always set to zero)
0 – Legacy Block on Send (Flag ignored and aways set to zero)
00,ef,00,06,fa,00,81,02,01,01,ff,01,ff,00,00 – Message to be sent (in hex)
```



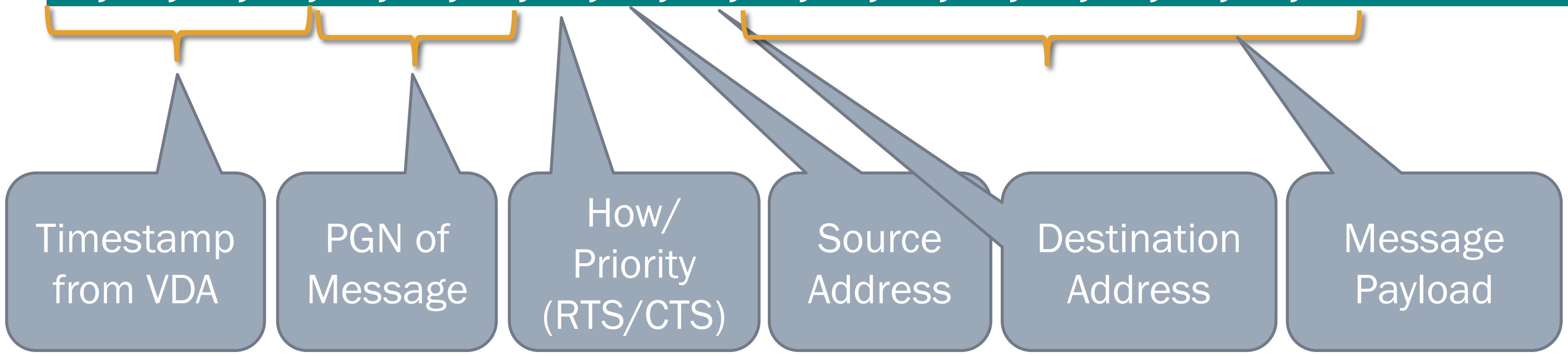
Note: In this case, the message is 9 bytes, so the VDA will encapsulate the message in the J1939 Transport Protocol.

SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

RP1210 Read Message

```
FT:0265,AT:0013   02,RM,19,4104,1,00,01,9b,da,00,ef,00,00,00,fa,81,01,02,00,01,70,01,05,30,

02 – Connected Client (Result from Client Connect)
RM – Read Message
19 – Message size in bytes
4104 – Buffer Size
1 – Block on Read (1 – BLOCKING IO)
00,01,9b,da,00,ef,00,00,00,fa,81,01,02,00,01,70,01,05,30 - Received Message
```



Note: In this case, the message is 9 bytes, so the CAN traffic will show the message in the J1939 Transport Protocol. The VDA provides the result.

Proprietary Diagnostic Protocols

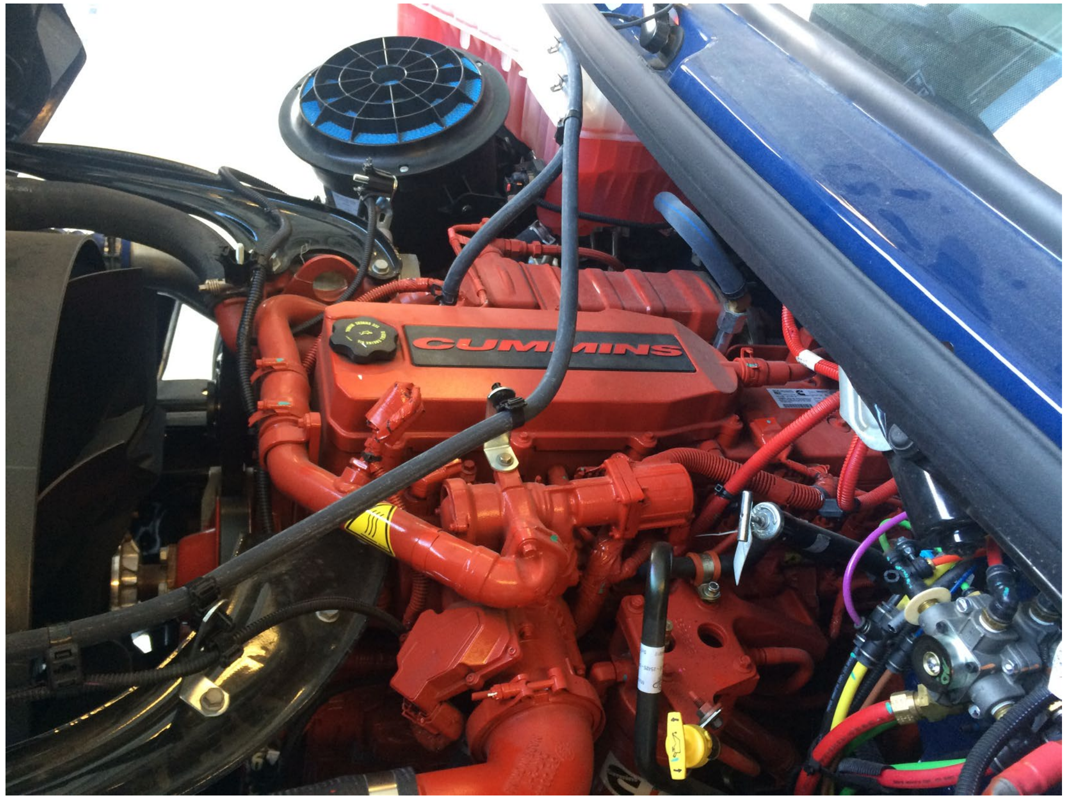
Network traffic from Cummins Insite shows diagnostics over Proprietary A messages: PGN 61184 (0xEF00)

Some J1939 fields are duplicated in proprietary protocols

Data may be richer compared to J1939

- Commands for service routines
- Calibration modifications

Navistar protocols extensively use Proprietary B messages (PGN =0xFFXX)



RP1210 Summary

The RP1210 system has been around since the 1990s

Vendors provide their details using the INI files; Applications parse the INI files

Technicians select their drivers based on the parsed INI files

Multiple drivers and VDAs can co-exist on computers

Function prototypes are common across all VDAs; Logging capabilities are different

RP1210 makes J1939 transport protocols transparent; CAN logs may look different

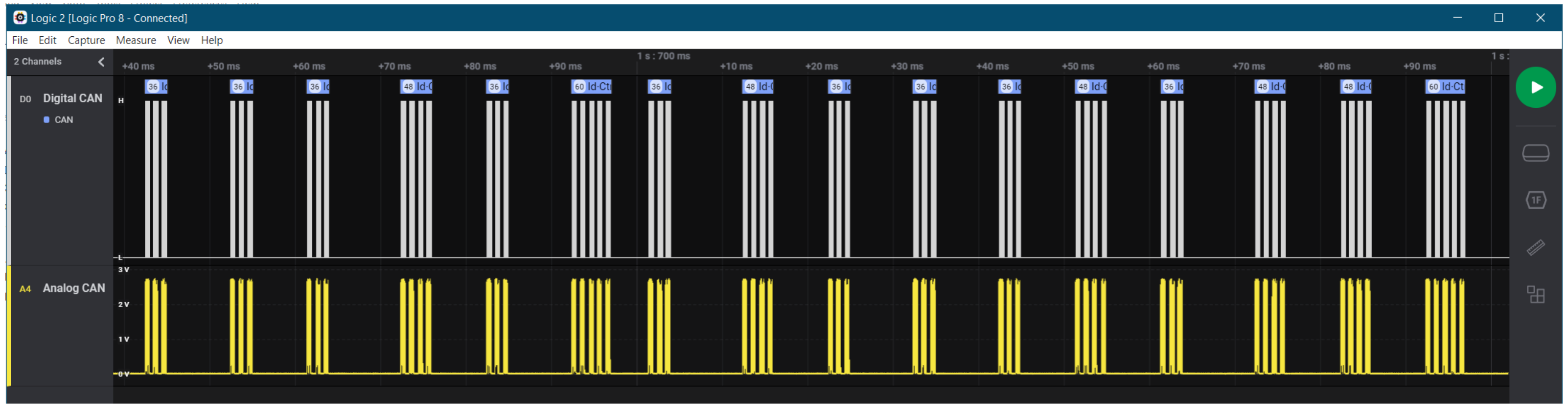
There are many more details in the actual RP1210 document

A minimal console program written with RP1210 function calls is called simpleRP1210
https://github.com/SystemsCyber/ShimDLL

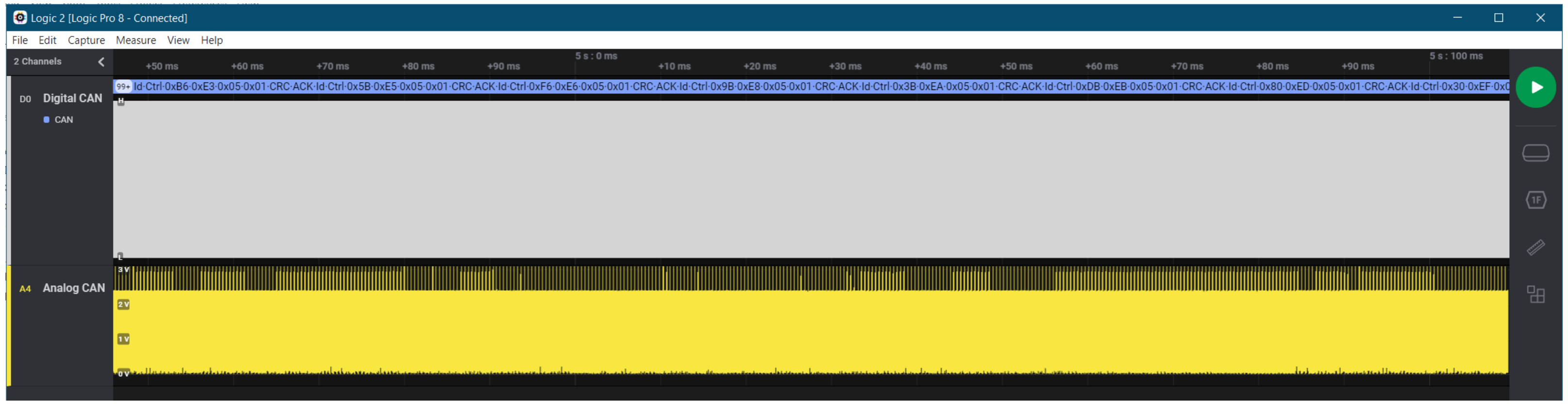
Cybersecurity Considerations for J1939

WHAT CAN DO WRONG IF A HACKER GETS ACCESS TO THE NETWORK?

Denial Of Service


Logic 2 [Logic Pro 8 - Connected]
File Edit Capture Measure View Help
2 Channels
D0 Digital CAN
CAN
A4 Analog CAN
1 s : 700 ms
3 V
2 V
1 V
0 V

Normal J1939

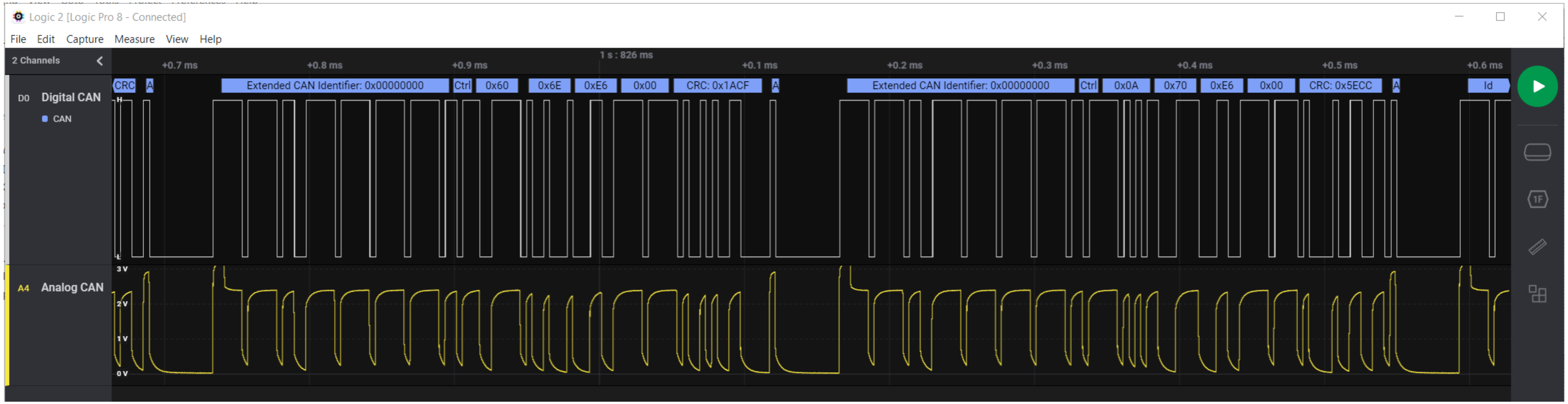

Logic 2 [Logic Pro 8 - Connected]
File Edit Capture Measure View Help
2 Channels
D0 Digital CAN
CAN
A4 Analog CAN
5 s : 0 ms
5 s : 100 ms

```
99+ Id·Ctrl·0xB6·0xE3·0x05·0x01·CRC·ACK·Id·Ctrl·0x5B·0xE5·0x05·0x01·CRC·ACK·Id·Ctrl·0xF6·0xE6·0x05·0x01·CRC·ACK·Id·Ctrl·0x9B·0xE8·0x05·0x01·CRC·ACK·Id·Ctrl·0x3B·0xEA·0x05·0x01·CRC·ACK·Id·Ctrl·0xDB·0xEB·0x05·0x01·CRC·ACK·Id·Ctrl·0x80·0xED·0x05·0x01·CRC·ACK·Id·Ctrl·0x30·0xEF·0x0
```

Flooded J1939

SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

Denial of Service



By repeating high priority messages (ID = 0), no other legitimate message can get access to the network.

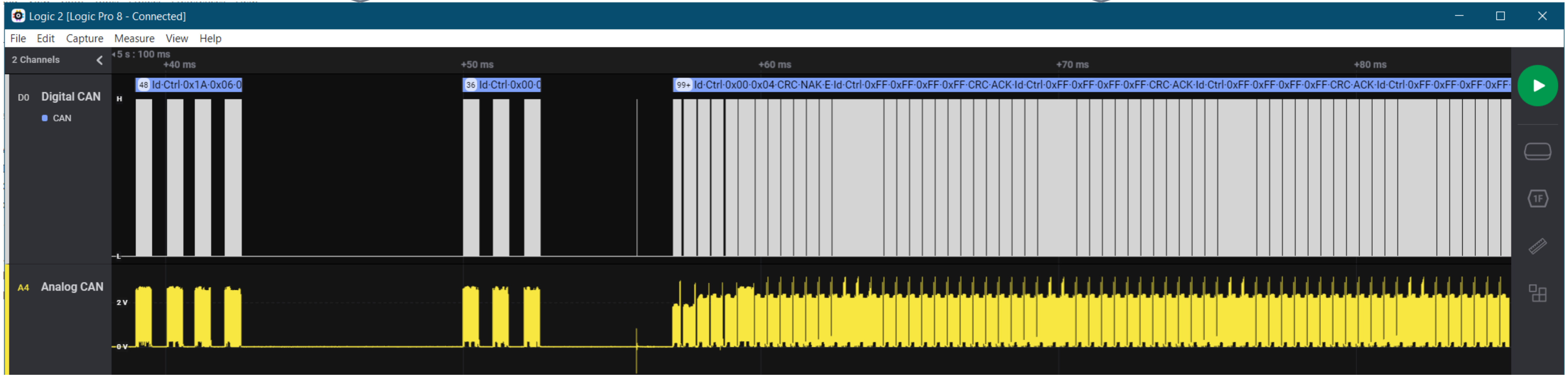
This will shut down communications and potentially stall a truck.

There are no native protections against this in J1939; avoid connecting unknown new devices to J1939.

Spoofing Messages and Commands

Normal J1939

Spoofed J1939



48 Id·Ctrl·0x1A·0x06·0

36 Id·Ctrl·0x00·0

99+ Id·Ctrl·0x00·0x04·CRC·NAK·E·Id·Ctrl·0xFF·0xFF·0xFF·0xFF·CRC·ACK·Id·Ctrl·0xFF·0xFF·0xFF·0xFF·CRC·ACK·Id·Ctrl·0xFF·0xFF·0xFF·0xFF·CRC·ACK·Id·Ctrl·0xFF·0xFF·0xFF·0xFF·

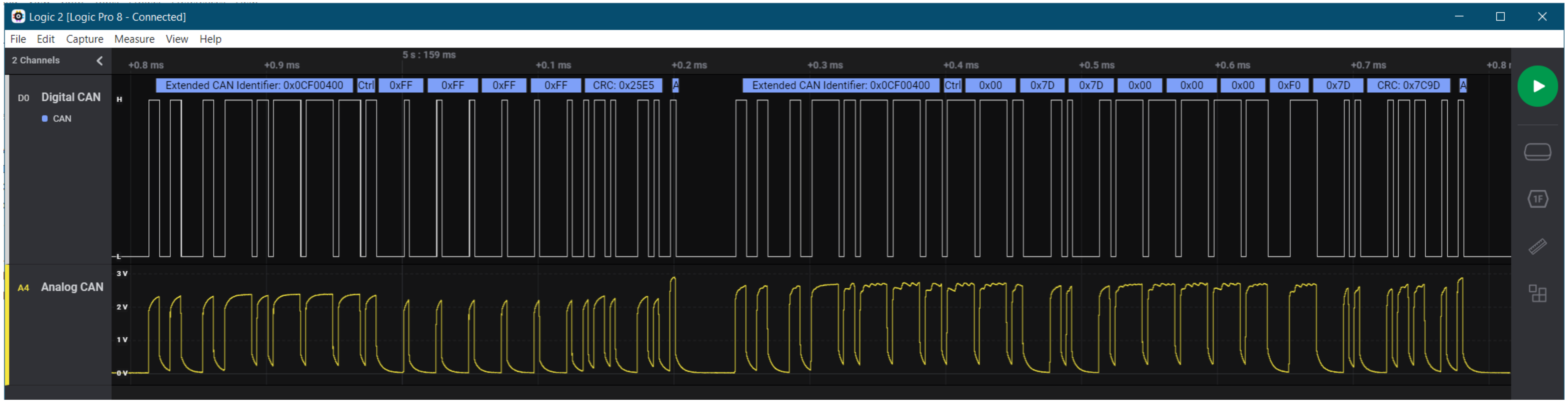
SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

Spoofing Messages and Commands

Two messages with the same IDs will be interpreted the same way

- One message is legitimate (right)
- The other is spoofed (left)

The network doesn't know which message is legitimate

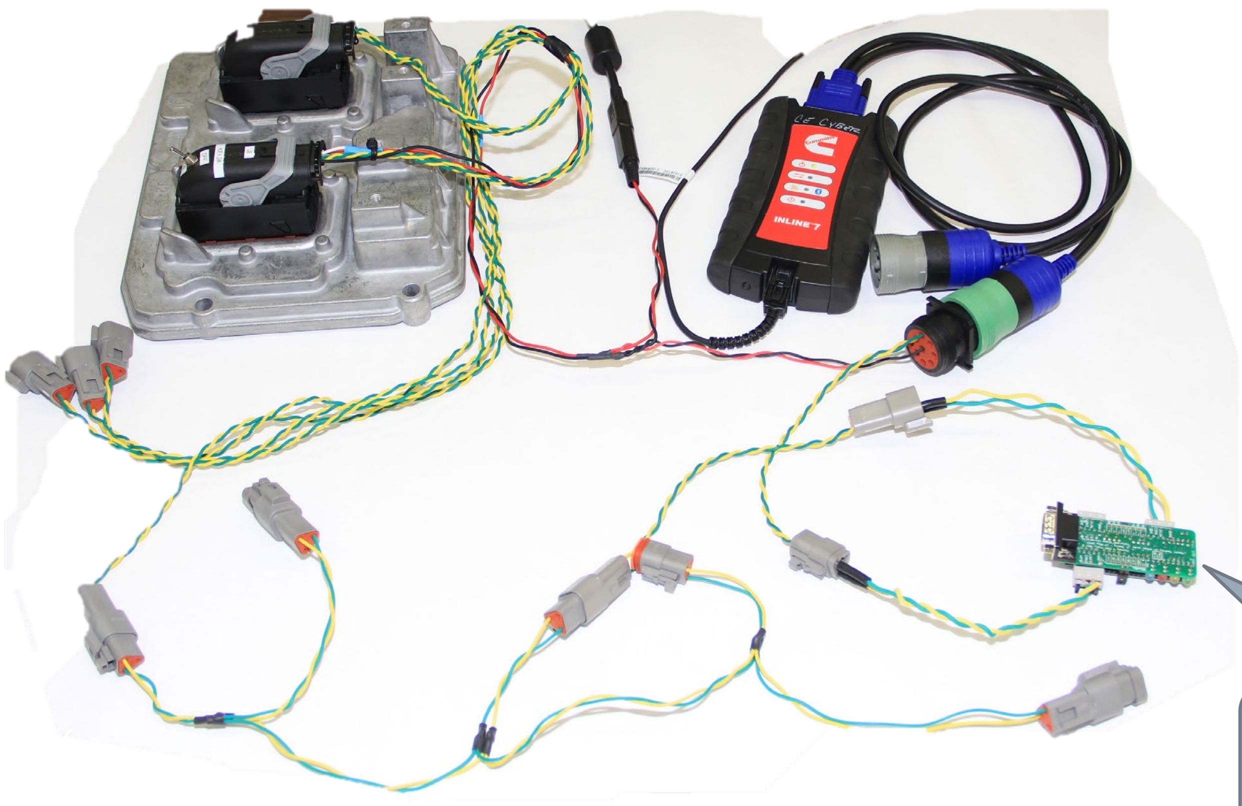


Extended CAN Identifier: 0x0CF00400 | Ctrl | 0xFF | 0xFF | 0xFF | 0xFF | CRC: 0x25E5 | A

Extended CAN Identifier: 0x0CF00400 | Ctrl | 0x00 | 0x7D | 0x7D | 0x00 | 0x00 | 0x00 | 0xF0 | 0x7D | CRC: 0x7C9D | A

The operator reported that the vehicle suddenly stopped working when they were sitting in traffic. There was no sign of overheating or mechanical failure

Machine-in-the-Middle (MITM) Attacks



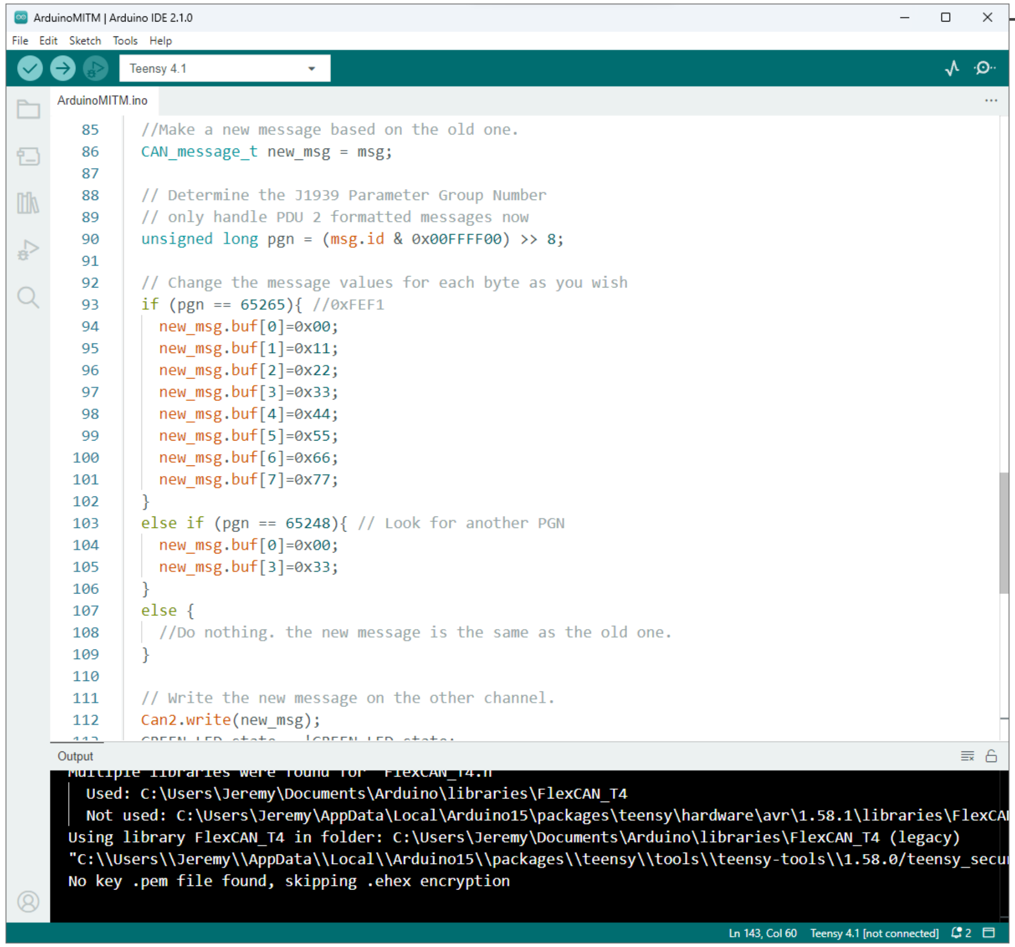
Insert Teensy 4.1 CAN Adapter Here

J1939 and CAN are trusting; there is no built-in authentication for CAN frames.

MITM devices can manipulate network traffic.

Use Arduino to modify the code at https://github.com/SystemsCyber/CAN-Adapter/tree/main/ArduinoMITM and change messages that show up in simpleRP1210.exe.

Exercise: MITM Programing



ArduinoMITM | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Teensy 4.1
ArduinoMITM.ino
Output
Ln 143, Col 60 Teensy 4.1 [not connected] 2

```
//Make a new message based on the old one.
CAN_message_t new_msg = msg;

// Determine the J1939 Parameter Group Number
// only handle PDU 2 formatted messages now
unsigned long pgn = (msg.id & 0x00FFFF00) >> 8;

// Change the message values for each byte as you wish
if (pgn == 65265){ //0xFEF1
  new_msg.buf[0]=0x00;
  new_msg.buf[1]=0x11;
  new_msg.buf[2]=0x22;
  new_msg.buf[3]=0x33;
  new_msg.buf[4]=0x44;
  new_msg.buf[5]=0x55;
  new_msg.buf[6]=0x66;
  new_msg.buf[7]=0x77;
}
else if (pgn == 65248){ // Look for another PGN
  new_msg.buf[0]=0x00;
  new_msg.buf[3]=0x33;
}
else {
  //Do nothing. the new message is the same as the old one.
}

// Write the new message on the other channel.
Can2.write(new_msg);
```

```
Multiple libraries were found for "FlexCAN_T4.h"
  Used: C:\Users\Jeremy\Documents\Arduino\libraries\FlexCAN_T4
  Not used: C:\Users\Jeremy\AppData\Local\Arduino15\packages\teensy\hardware\avr\1.58.1\libraries\FlexCA
Using library FlexCAN_T4 in folder: C:\Users\Jeremy\Documents\Arduino\libraries\FlexCAN_T4 (legacy)
"C:\\Users\\Jeremy\\AppData\\Local\\Arduino15\\packages\\teensy\\tools\\teensy-tools\\1.58.0/teensy_secu
No key .pem file found, skipping .ehex encryption
```

This will change

```
F1 FE 00 06 00 FF FF FF FE FF FF FF FF FF
```

to

```
F1 FE 00 06 00 FF 00 11 22 33 44 55 66 77
```

in the simpleRP1210.exe output.

Summary

The need for in-vehicle communication using CAN and SAE J1939

Connecting to J1939 Networks

Classify the different types of communication over J1939

Interpret J1939 network traffic using the SAE Standard

Recognize SAE J1939 Transport Protocols for larger messages

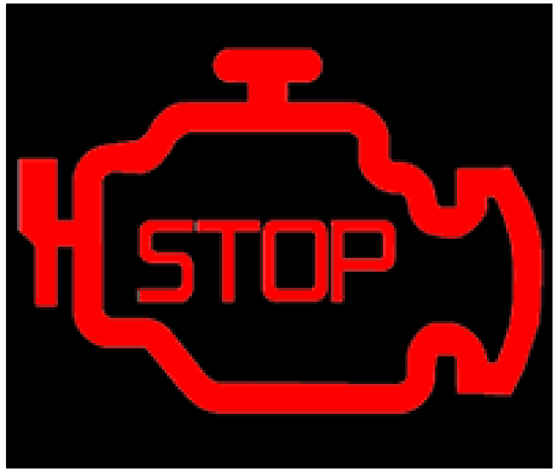
Understand J1939 Diagnostic Messages

Introduction to J1939 Address Claiming

Demonstration of RP1210 functionality for diagnostics

Showed examples of Unified Diagnostic Services (UDS) over J1939

Realize J1939 is inherently an open (and potentially insecure) read-write bus



BACKUP Slides

Unified Diagnostic Services over J1939

UDS COMMUNICATIONS AS DEFINED IN ISO 15765

Unified Diagnostic Services (UDS) over CAN

UDS is used by many truck component makers for diagnostics and maintenance actions

- Bendix Brake Controllers
- Detroit Diesel Electronic Controllers
- Mack and Volvo Diagnostics
- Many others

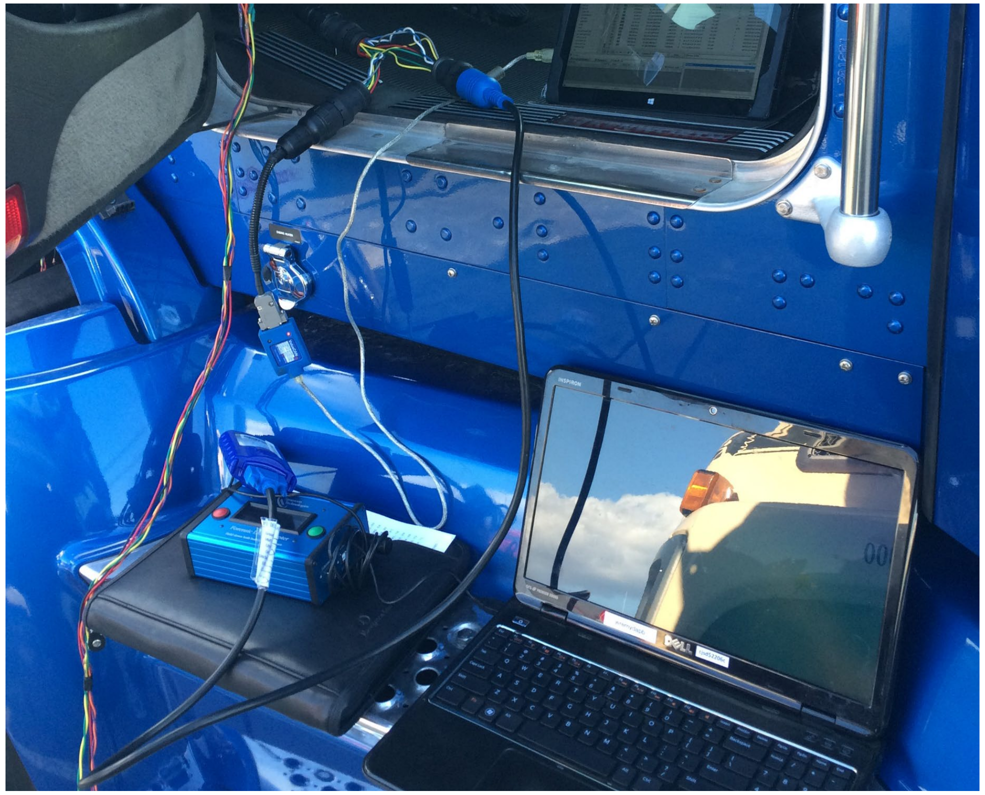
Message meaning is defined in ISO 14229

J1939 uses PGN 55808 (0xDA00) for UDS

- PDU1 format (Point-to-point) uses Destination Address

UDS has its own transport protocol (ISO-TP)

- 4096 bytes maximum per message.



SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

UDS Example: Establish UDS Session

MESSAGE FROM OFF-BOARD TOOL (SA: F1)

CAN ID

```
18DA00F1 02 10 03 00 00 00 00 00
```

```
18 – Priority 6
DA – PF for ISO 15765 (PGN = 0xDA00)
00 – Dest. Address (0x00 = Engine #1)
F1 – Source Address (0xF1 = Service Tool)

0 – Single Frame Message
2 – Message Size (2 bytes)
10 – Service ID (0x10 Establish Session)
03 – Session Type (0x03 = Extended)
```

MESSAGE FROM ENGINE #1 (SA: 00)

```
18DAF100 06 50 03 00 14 00 C8 01
```

Unused

```
0 – Single Frame Message
6 – Message Size (6 bytes)
50 – Service ID Response for Session
03 – Session Type (0x03 = Extended)
00 14 00 C8 – Session Timing Parameters
```

Service Identifier (SID) Responses always add 0x40 to the Request SID.
Example: 0x10 - Request
0x50 - Response

UDS Example: Seed-Key Exchange

MESSAGE FROM OFF-BOARD TOOL (SA: F1)

```
18DA00F1 02 27 09 00 00 00 00 00

18DA00F1 04 27 0A B1 27 00 00 00
```

```
0 – Single Frame Message
2 – Message size (2 bytes)
27 – SID for Security Exchange
09 – Request Seed
```

```
0 – Single Frame Message
4 – Message size (4 bytes)
27 – SID for Security Exchange
0A – Parameter for Key
B1 27 – Value of Key
```

MESSAGE FROM ENGINE #1 (SA: 00)

```
18DAF100 04 67 09 EC 65 21 05 03

18DAF100 02 67 0A EC 65 21 05 03
```

Repeated Buffer Contents Not Used

```
0 – Single Frame Message
4 – Message size (4 bytes)
67 – SID for Security Exchange Response
09 – Parameter for Seed
EC 65 – Value of Seed
```

```
0 – Single Frame Message
2 – Message size (2 bytes)
67 – SID for Security Exchange Response
0A – Key Acknowledge (Accepted)
```

UDS Example: Read Data By Identifier

MESSAGE FROM OFF-BOARD TOOL (SA: F1)

```
18DA00F1 03 22 F1 51 00 00 00 00
18DA00F1 30 08 00 00 00 00 00 00
```

```
0 – Single Frame Message
3 – Message size (3 bytes)
22 – SID (0x22 = Read Data By Identifier)
F1 51 – Data Identifier (Proprietary)

30 – Flow Control Frame
08 – Send up to 8 more frames
```

MESSAGE FROM ENGINE #1 (SA: 00)

```
18DAF100 10 09 62 F1 51 11 29 00
18DAF100 21 11 2A 00 51 11 29 00
```

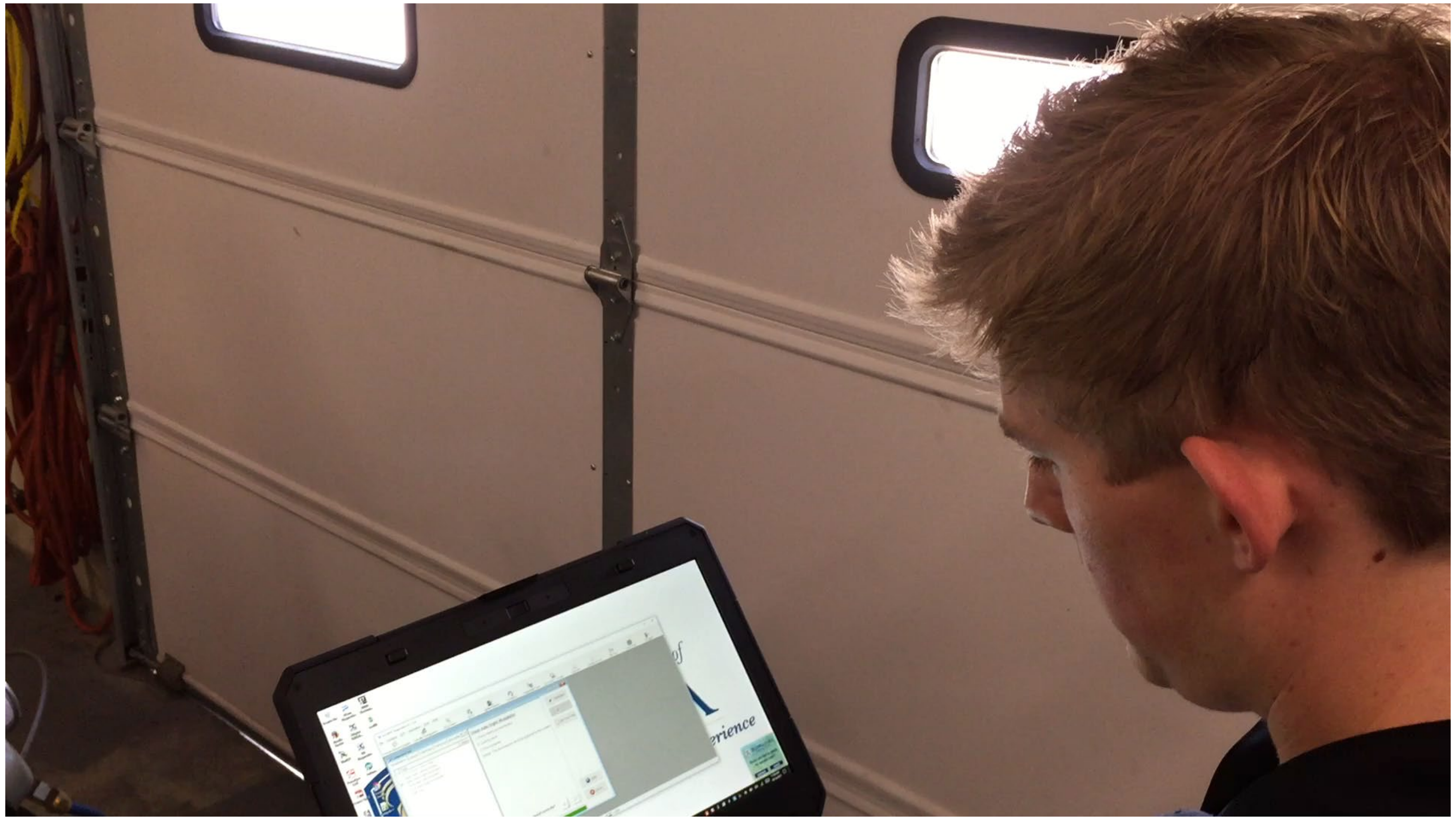
Same memory

```
1 – First Frame of Message
0 09 – Message Size (up to 4096)
62 – SID Response (0x22 + 0x40)
F1 51 – Data Identifier of Response
11 29 00 – First 3 bytes of data

2 – Consecutive Frame
1 – Frame Sequence Number
11 2A 00 – Last 3 Bytes of Data
```

Example UDS Session for Brake Controls

- A session is established for brake controller diagnostics
- Students commanded a brake chuff test
- All communications went over UDS
- The brake controller trusts the UDS commands



Resources for Unified Diagnostic Services

UDS, like J1939, is extensive and has many reference documents

Most UDS communications have proprietary meaning

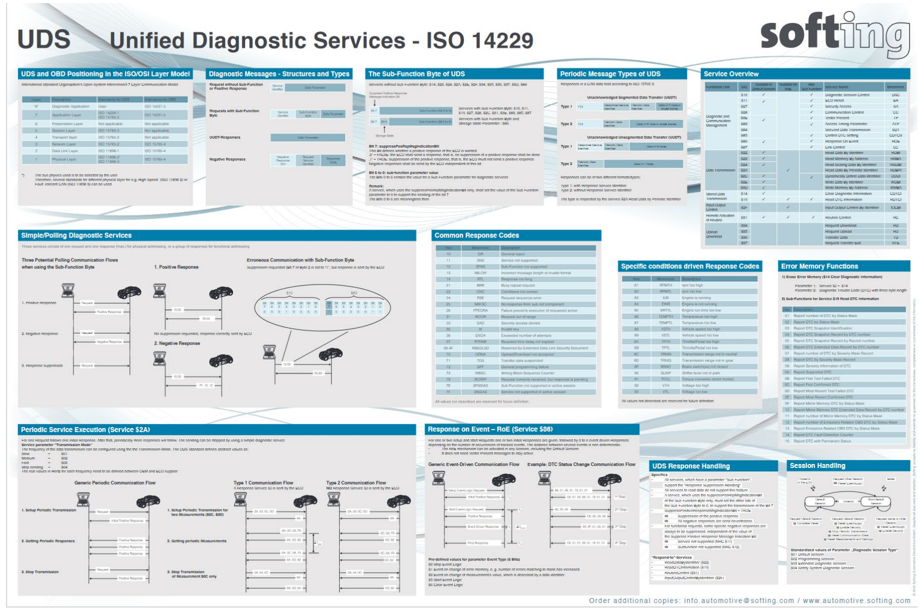
UDS is also used in passenger cars (with different CAN IDs)

UDS uses a server and client model

- Server: on-board ECU
- Client: off-board diagnostic tool

Links for additional information:

- https://en.wikipedia.org/wiki/ISO_15765-2
- https://www.sae.org/publications/books/content/r-474/
- https://automotive.softing.com/fileadmin/sof-files/pdf/de/ae/poster/UDS_Faltposter_softing2016.pdf


UDS Unified Diagnostic Services - ISO 14229 softing

UDS and OBD Positioning in the ISO/OSI Layer Model
Diagnostic Messages - Structures and Types
The Sub-Function Byte of UDS
Periodic Message Types of UDS
Service Overview
Simple/Polling Diagnostic Services
Common Response Codes
Specific conditions driven Response Codes
Error Memory Functions
Periodic Service Execution (Service $2A)
Response on Event – RoE (Service $86)
UDS Response Handling
Session Handling

[illegible]

Order additional copies: info.automotive@softing.com / www.automotive.softing.com