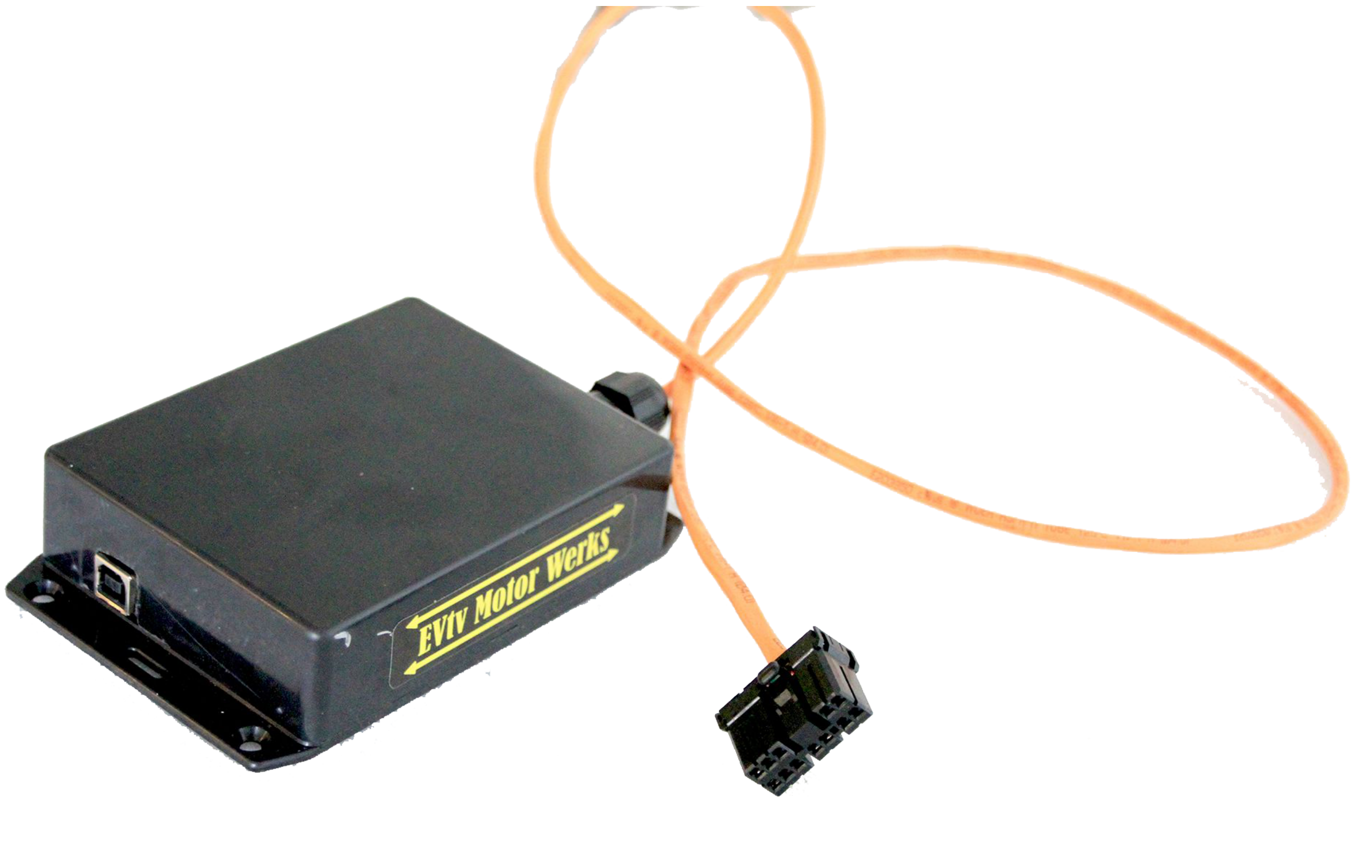
User Manual

SavvyCAN Adapter for Tesla Model S



INTRODUCTION

The EVTV Tesla Model S SavvyCAN Adapter allows you to easily capture and analyze Controller Area Network (CAN) Traffic from the Tesla Model S automobile. It is somewhat focused on the Powertrain CAN Bus 3 but can certainly be used for the other Tesla CAN bus examination.

Analyzing the Tesla CAN bus is challenging in several ways. First, their OBDII connector is essentially useless. All the real CAN buses are on a very non-standard diagnostic port located immediately under the 17inch center console display and behind a small storage compartment there.

Second, they actually have at least six different data busses that could be termed a CAN bus.

Third, these 500kbps busses are just loaded – we've logged frame rates as high as 3500 frames per second on a 500 kbps bus. This is far beyond the abilities of most of the inexpensive OBDII CAN tools available and even beyond the reach of many such commercial devices without dropping frames.

The EVTV Tesla Model S CANdue CAN Bus Kit includes:

1. EVTV Due CAN Multicontroller Adapter adapter with a Tesla Model S diagnostics port harness/connector.

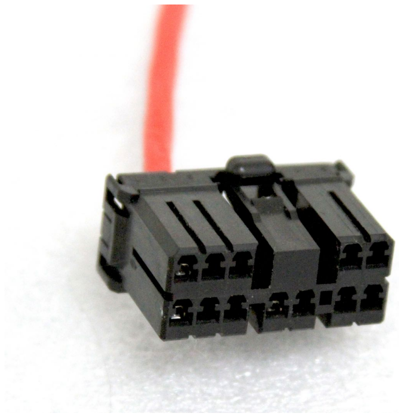
2. Generalized Electric Vehicle Reverse Engineering Tool (GEVRET) software to run on the hardware adapter.

3. SavvyCAN CAN Data Analysis software to run on any laptop or desktop computer – Microsoft Windoze, Linux, or MAC OSX.

4. USB cable with printer B connector.

EVTV Due CAN Multicontroller Adapter

The EVTV Due CAN Muticontroller Adapter hardware is a 84MHz SAM3x ARM Arduino Due compatible multicontroller with integrated CAN bus transceiver. It is housed in a plastic enclosure with a Mini B Universal Serial Bus printer port for connection to a laptop and an external harness to mate with the Tesla Model S diagnostics port connector.



The internal multicontroller board provides a single differential CANbus port accessible through screw terminals on one end. These connections are wired to the Tesla Model S diagnostics port mating connector through a cable that exits the enclosure through a weatherproof gland nut on one end.

A USB B port is provided at one end allowing connection from a laptop computer to the adapter. This is the much more robust printer style USB port.

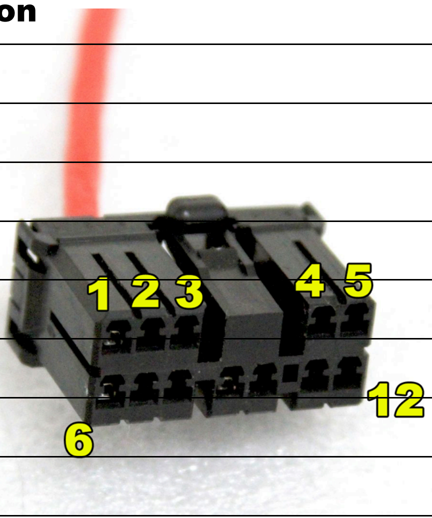
It also features a 6-16vdc power input terminal and ground allowing the board to run independently of a laptop using vehicle 12v power.

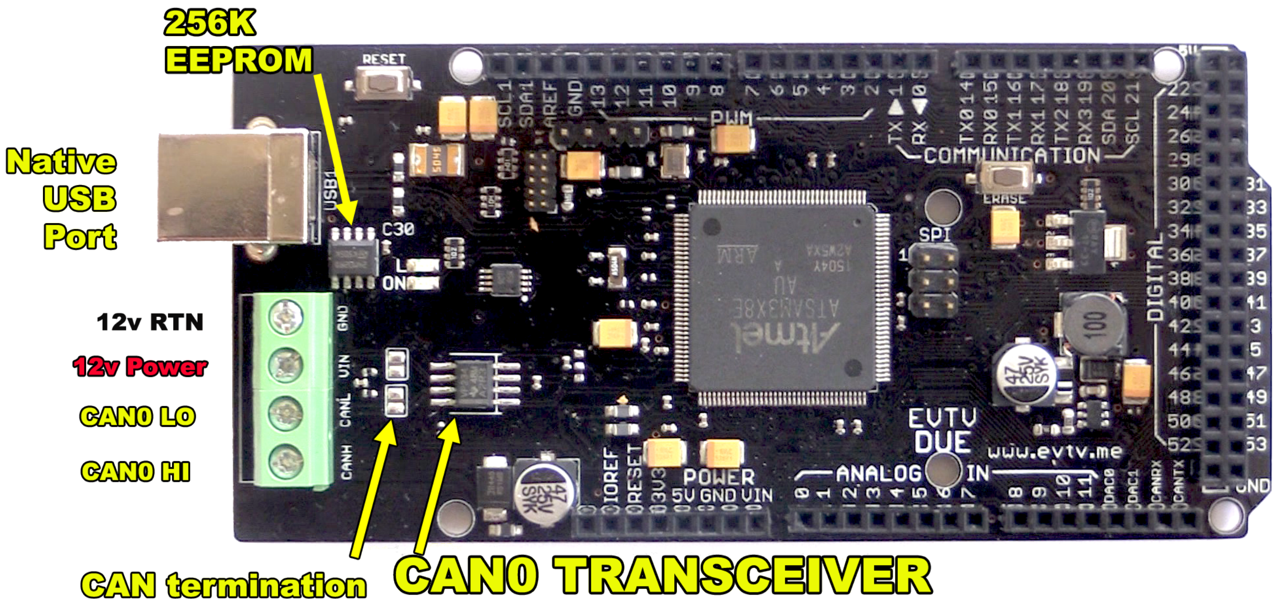
.

The harness comes prewired for pins 1 and 6 of the diagnostics port to CAN0. This is the PowerTrain CAN bus or CAN3 bus on the Tesla Model S. However, additional pins are provided for this connector allowing you to modify which CAN bus is monitored.

Tesla Model S Diagnostics Port Connector X437 Pin Definitions

Pin	Description
1	CAN 3 HI – Powertrain Bus
2	CAN 4 HI – Body Fault Bus
3	K bus ISO9141 single wire LIN
4	CAN 6 HI Chassis Bus
5	CAN 2 HI Body Bus
6	CAN 3 LO Powertrain Bus
7	CAN 4 LO Body Fault Bus
8	Not used
9	Reference Ground
10	12v
11	CAN 6 LO Chassis Bus
12	CAN 2 LO Body Bus





SOFTWARE

The Tesla CAN adapter comes with the General Electric Vehicle Reverse Engineering Tool (GEVRET) software installed. You can connect any laptop with an ordinary serial terminal screen to the USB port of the device to connect directly to the device if you like.

Entering **?** on the command line and pressing ENTER will cause a menu screen to come up. You can use this to manually select CAN bus selections, speeds, etc.

Normally, this will not be used by most users of this product.

A much more capable CAN analysis program is available for using the adapter from a laptop computer via USB port. It is titled SavvyCAN and features a number of advanced CAN analysis functions. But it makes it very easy to log off CAN traffic and save it to files, load them later for analysis, and examine them with a number of views.

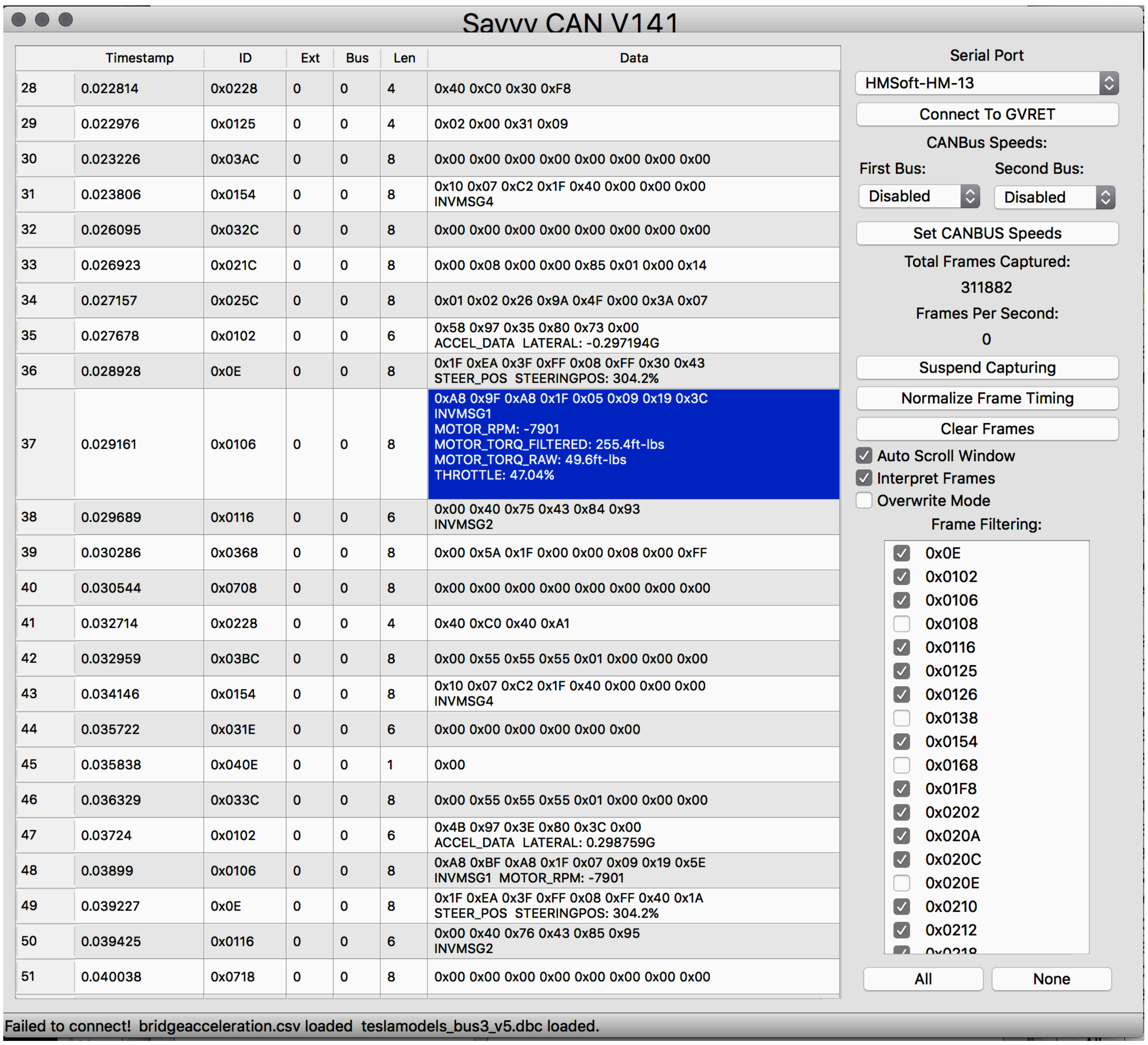
SavvyCAN is available for Windoze, Mac OSX, and Linux. It exchanges messages with the Tesla CAN adapter to send and receive CAN message traffic automatically.

Links for both SavvyCAN and GEVRET are available at the EVTV store website under the detailed view of the Tesla CAN bus adapter product. These programs are under continuous development so for best results ensure you have the latest versions installed.

Either program can also be found at **`http://github.com/collin80`**.

The basic process of capturing CAN data is pretty straightforward:

1. Push down on the small shelf below the Tesla Model S center screen until it unlatches.
2. Reach in and locate the small wiring harness with diagnostics connector and pull out to gain access.
3. Connect the EVTV adapter to this connector until it clicks
4. Connect your laptop to the USB port on the EVTV CAN adapter
5. Start the SavvyCAN software program on your laptop.
6. In the upper right hand corner of the screen, select your Serial Port used to connect the adapter from the dropdown list provided.
7. Click on the CONNECT TO GVRET button. GVRET is the Generalized Vehicle Reverse Engineering Tool software installed on the adapter.
8. Under FIRST BUS, select 500000 as the desired speed and click SET CANBUS SPEEDS. The Tesla Model S Drive Train CAN bus operates at 500 kbps.



Savvy CAN V141

	Timestamp	ID	Ext	Bus	Len	Data
28	0.022814	0x0228	0	0	4	0x40 0xC0 0x30 0xF8
29	0.022976	0x0125	0	0	4	0x02 0x00 0x31 0x09
30	0.023226	0x03AC	0	0	8	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
31	0.023806	0x0154	0	0	8	0x10 0x07 0xC2 0x1F 0x40 0x00 0x00 0x00 INVMSG4
32	0.026095	0x032C	0	0	8	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
33	0.026923	0x021C	0	0	8	0x00 0x08 0x00 0x00 0x85 0x01 0x00 0x14
34	0.027157	0x025C	0	0	8	0x01 0x02 0x26 0x9A 0x4F 0x00 0x3A 0x07
35	0.027678	0x0102	0	0	6	0x58 0x97 0x35 0x80 0x73 0x00 ACCEL_DATA LATERAL: -0.297194G
36	0.028928	0x0E	0	0	8	0x1F 0xEA 0x3F 0xFF 0x08 0xFF 0x30 0x43 STEER_POS STEERINGPOS: 304.2%
37	0.029161	0x0106	0	0	8	0xA8 0x9F 0xA8 0x1F 0x05 0x09 0x19 0x3C INVMSG1 MOTOR_RPM: -7901 MOTOR_TORQ_FILTERED: 255.4ft-lbs MOTOR_TORQ_RAW: 49.6ft-lbs THROTTLE: 47.04%
38	0.029689	0x0116	0	0	6	0x00 0x40 0x75 0x43 0x84 0x93 INVMSG2
39	0.030286	0x0368	0	0	8	0x00 0x5A 0x1F 0x00 0x00 0x08 0x00 0xFF
40	0.030544	0x0708	0	0	8	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
41	0.032714	0x0228	0	0	4	0x40 0xC0 0x40 0xA1
42	0.032959	0x03BC	0	0	8	0x00 0x55 0x55 0x55 0x01 0x00 0x00 0x00
43	0.034146	0x0154	0	0	8	0x10 0x07 0xC2 0x1F 0x40 0x00 0x00 0x00 INVMSG4
44	0.035722	0x031E	0	0	6	0x00 0x00 0x00 0x00 0x00 0x00
45	0.035838	0x040E	0	0	1	0x00
46	0.036329	0x033C	0	0	8	0x00 0x55 0x55 0x55 0x01 0x00 0x00 0x00
47	0.03724	0x0102	0	0	6	0x4B 0x97 0x3E 0x80 0x3C 0x00 ACCEL_DATA LATERAL: 0.298759G
48	0.03899	0x0106	0	0	8	0xA8 0xBF 0xA8 0x1F 0x07 0x09 0x19 0x5E INVMSG1 MOTOR_RPM: -7901
49	0.039227	0x0E	0	0	8	0x1F 0xEA 0x3F 0xFF 0x08 0xFF 0x40 0x1A STEER_POS STEERINGPOS: 304.2%
50	0.039425	0x0116	0	0	6	0x00 0x40 0x76 0x43 0x85 0x95 INVMSG2
51	0.040038	0x0718	0	0	8	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Serial Port: HMSoft-HM-13

Connect To GVRET

CANBus Speeds:

First Bus: Disabled Second Bus: Disabled

Set CANBUS Speeds

Total Frames Captured: 311882

Frames Per Second: 0

Suspend Capturing

Normalize Frame Timing

Clear Frames

[✓] Auto Scroll Window

[✓] Interpret Frames

[ ] Overwrite Mode

Frame Filtering:

[✓] 0x0E
[✓] 0x0102
[✓] 0x0106
[ ] 0x0108
[✓] 0x0116
[✓] 0x0125
[✓] 0x0126
[ ] 0x0138
[✓] 0x0154
[ ] 0x0168
[✓] 0x01F8
[✓] 0x0202
[✓] 0x020A
[✓] 0x020C
[ ] 0x020E
[✓] 0x0210
[✓] 0x0212

All None

Failed to connect! bridgeacceleration.csv loaded teslamodels_bus3_v5.dbc loaded.

At this point the SavvyCAN display should start showing CAN traffic in the DATA field. Make sure your Model S is “started” and the dash instrument displays are live to ensure flow of CAN traffic. You are now “recording” CAN traffic from the Tesla Model S drive train to memory.

TOTAL FRAMES CAPTURED

This displays the total number of message frames received from the adapter and held in memory.

FRAMES PER SECOND

This is the rate of frames being received per second.

SUSPEND CAPTURING

Pressing this button simply discontinues capturing frames from the port to the memory. It also changes the button to **RESUME CAPTURING**. In this way, you can simply stop and start frame captures at will.

NORMALIZE FRAME TIMING

Normally, the frames are displayed with the time stamp applied by the GVRET device on receipt. This is the number of microseconds since the device was first powered up. But you might want something more in line with the capture. Click NORMALIZE FRAME TIMING to set the FIRST frame received as time 000. All subsequent frame times will be in relation to that frame.

CLEAR FRAMES

This button simply resets the logging memory to zero frames – effectively erasing everything you've captured so far. You can use CLEAR FRAMES to clear memory and then RESUME CAPTURING to start a fresh capture at any time.

AUTO SCROLL WINDOW

Normally, the data window simply displays the first 25 frames or so and you can "scroll" down with the mouse to view subsequent frames. This checkbox allows you to autoscroll the window to the left so that the newest data always appears as the bottom line of the screen. In this way you are always viewing the LAST 25 frames received. You can still scroll up and down.

INTERPRET FRAMES

Interpret Frames allows you to examine values contained in CAN frames. It performs any necessary math functions on the data for you to get the real value. For example, Voltage might be in bytes 1 and 2 of a frame in LSB/MSB format, and multiplied by 100. Interpret Frames would then take byte 2 *256, add the value in frame 1, and divide the total by 100 to get voltage.

These functions are defined for each frame in a Vector Graphics format .DBC file. Think of this as a rules file or library for all the different data definitions you have. This file can be LOADED by using the LOAD DBC file on the top bar FILE menu. They can also be SAVED using the SAVE DBC option on the same menu.

DBC files are normally created and edited separately.

OVERWRITE MODE

Overwrite mode shows all the UNIQUE message IDs and overwrites each with the latest data as it is received. Works in capture but not on a loaded data log.

FRAME FILTERING

Frame filtering is a very powerful function allowing you to define specifically which frames are displayed. The window beneath lists all unique frames received so far. The ALL button at the bottom puts a check box next to each message ID assuring that they will be captured. The NONE button removes all checkboxes.

You can of course put check marks next to any message IDs of interest and those will subsequently be displayed.

Note that all frames are still captured to memory. The filter simply determines which frames are displayed.

FRAME NUMBER

Leftmost column on the display lists the number of the message frame received. Each incoming frame is assigned a subsequent number.

TIMESTAMP

The time at which the associated frame is received starting with the beginning of capture. As noted, this can also be “normalized” with the first frame at time 00000.

ID

The 11-bit or 29-bit message identification number from the received frame.

EXT

This will display a 1 if 29-bit extended frame was received or 0 if standard frame

BUS

Some adapters can support two CAN busses and SavvyCAN can receive CAN traffic from two busses simultaneously. This column shows which bus (usually 0 and 1) that the associated message was received on. This will always be 0 for the Tesla Model S capture device.

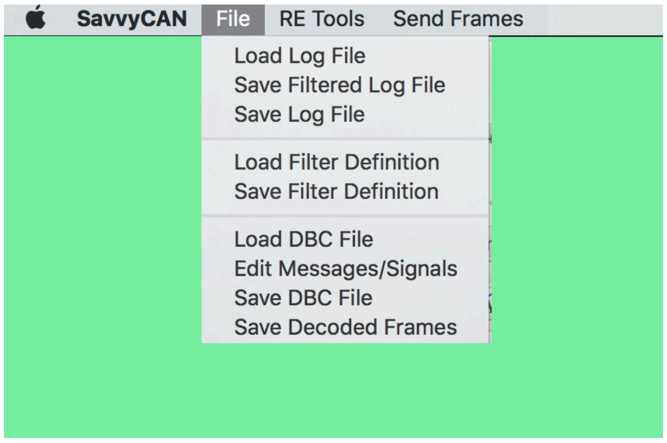
LEN

Number of data payload bytes in this message.

DATA

This shows the actual data payload bytes received in hexadecimal format. If INTERPRET FRAMES is on it may also show additional data showing what is contained in those bytes as interpreted by the DBC file rules.

FILE MANAGEMENT



In addition to the main communications screen, SavvyCAN has a top bar menu allowing access to other functions and indeed other analysis screens.

SAVE LOG FILE

The FILE menu lists a number of options allowing you to save and load data in various files.

SAVE LOG FILE is the most important function of SavvyCAN. It allows you to save all the CAN traffic you have captured to your hard drive in a variety of file formats for later use by SavvyCAN, or by other programs or spreadsheets.

LOAD LOG FILE

Load logfile of course is the other end of this. In addition to CAPTURING CAN data, SavvyCAN provides many analysis tools that can be applied to previously captured log files as well. LOAD LOG FILE simply allows you to reload a previous capture into memory for analysis.

SAVE FILTERED LOG FILE

As previously described, the filter window on the main capture screen simply determines what messages are DISPLAYED – all messages being retained in memory. But you may truly only be interested in a few messages. SAVE FILTERED LOG FILE allows you to save a data log of JUST the frames of interest. In this way, the next time you load that file, it will contain only the messages of interest and will of course be a much smaller file as well.

SAVE AND LOAD FILTER DEFINITION

Similarly, there may be quite a bit of work developing your message ID filter where you have hundreds of possible messages, and several dozen that you ARE in fact interested in. This function allows you to save a filter definition, and later reload it without having to manually check each box again.

LOAD/SAVE DBC FILES

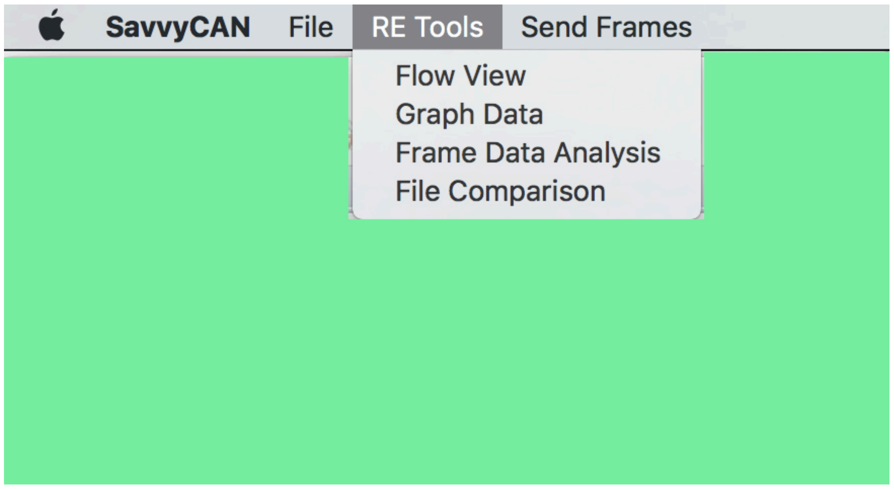
DBC files are simply a file format developed by Vector Graphics to save data definitions of CAN data in a library file allowing you to interpret those frames later using the definitions. The file format simply ends in **`.dbc.`**

SavvyCAN actually supports the Vector data definition files and you can import and export .DBC files from other programs into SavvyCAN. You can also EDIT MESSAGES/SIGNALS and SAVE DECODED FRAMES – that is messages for which the data definition is known.

REVERSE ENGINEERING TOOLS

SavvyCAN goes quite beyond capturing and saving CAN messages from the bus. It provides a variety of data analysis tools that can be of huge assistance in reverse engineering what messages do what on a CAN bus.

These tools are available in the RETOOLS menu of the SavvyCAN top bar menu.



FLOW VIEW

Flow view is provided to allow you to select a single message ID and then follow its flow through an entire CAN capture session, noting the changes from frame to frame for that single message ID.

You basically "play" back the capture recording while focusing on a single frame.

Up to 8 data bytes are displayed and you can easily compare their numeric contents to the startup state or to the previous frame.

Flow View also graphs each of up to 8 data bytes in the payload.

And finally, a 64 bit array of all 8 data bytes and each bit of each byte is graphically represented on screen so you can see individual bits wink in and out as it changes from frame to frame.

FRAME IDs FOUND

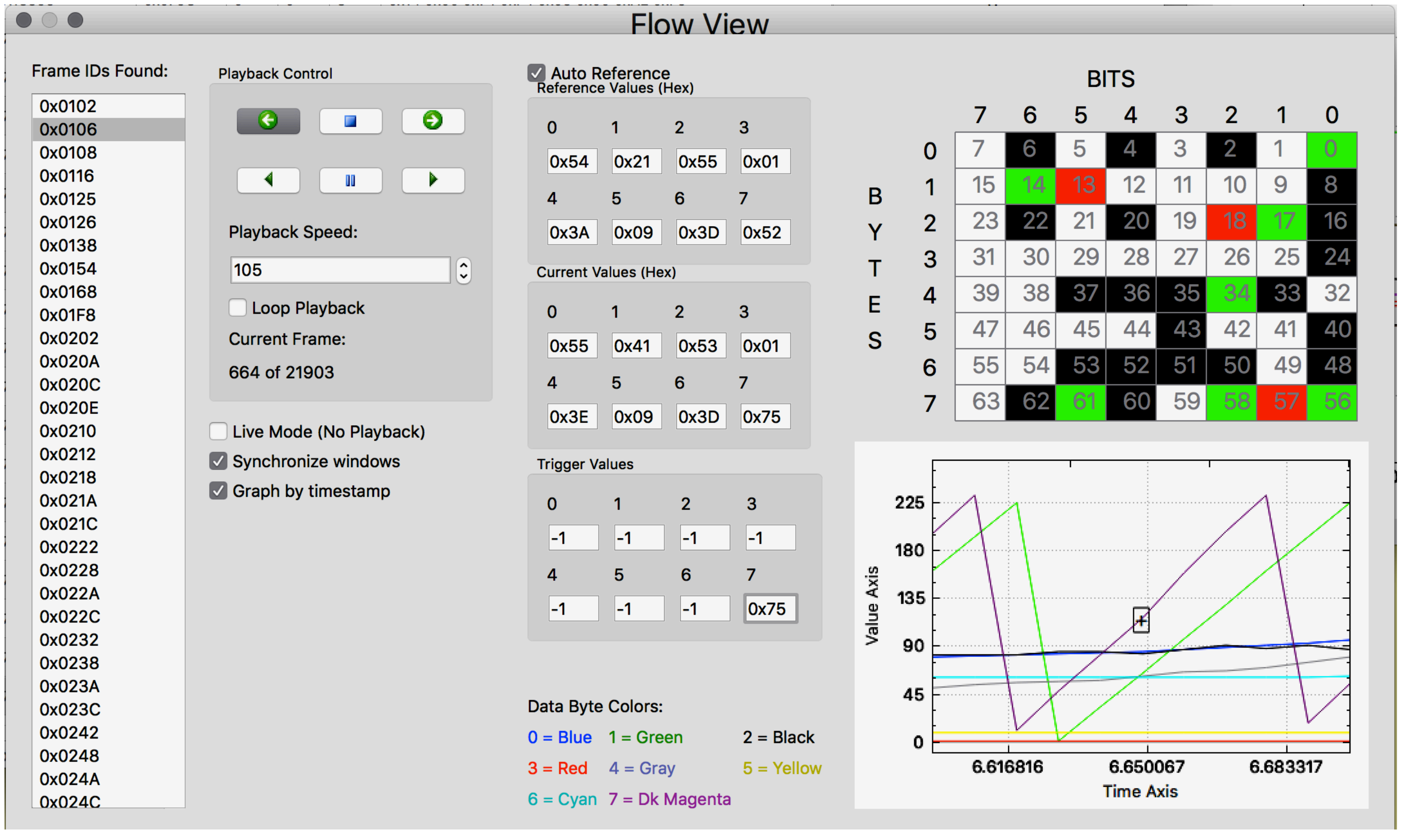
This box lists all unique CAN messages available in the current capture file in memory. You select the frame to examine with Flow View by highlighting one of these message IDS.

PLAYBACK CONTROL

The familiar playback controls allowing you to play through the series of CAN messages of that ID, either automatically or manually stepping through them one at a time.

PLAYBACK SPEED

Allows you to vary the rate at which automatic playback occurs.



Frame IDs Found:

0x0102
0x0106
0x0108
0x0116
0x0125
0x0126
0x0138
0x0154
0x0168
0x01F8
0x0202
0x020A
0x020C
0x020E
0x0210
0x0212
0x0218
0x021A
0x021C
0x0222
0x0228
0x022A
0x022C
0x0232
0x0238
0x023A
0x023C
0x0242
0x0248
0x024A
0x024C

Playback Speed: 105

☐ Loop Playback

Current Frame: 664 of 21903

☐ Live Mode (No Playback)

☑ Synchronize windows

☑ Graph by timestamp

☑ Auto Reference

Reference Values (Hex)

0	1	2	3
0x54	0x21	0x55	0x01
4	**5**	**6**	**7**
0x3A	0x09	0x3D	0x52

Current Values (Hex)

0	1	2	3
0x55	0x41	0x53	0x01
4	**5**	**6**	**7**
0x3E	0x09	0x3D	0x75

Trigger Values

0	1	2	3
-1	-1	-1	-1
4	**5**	**6**	**7**
-1	-1	-1	0x75

Data Byte Colors:

0 = Blue 1 = Green 2 = Black
3 = Red 4 = Gray 5 = Yellow
6 = Cyan 7 = Dk Magenta

BITS

BYTES	7	6	5	4	3	2	1	0
0	7	⬛ 6	5	⬛ 4	3	⬛ 2	1	🟩 0
1	15	🟩 14	🟥 13	12	11	10	9	⬛ 8
2	23	⬛ 22	21	⬛ 20	19	🟥 18	🟩 17	16
3	31	30	29	28	27	26	25	⬛ 24
4	39	38	⬛ 37	⬛ 36	⬛ 35	🟩 34	33	32
5	47	46	45	44	⬛ 43	42	41	⬛ 40
6	55	54	⬛ 53	⬛ 52	⬛ 51	⬛ 50	49	⬛ 48
7	63	⬛ 62	🟩 61	⬛ 60	59	🟩 58	🟥 57	🟩 56

LOOP PLAYBACK

With this checkbox, once end of file is reached, you can simply start over with the first message captured with that message ID.

CURRENT FRAME

Shows you total frames of this message ID in the file and the frame number of the current frame displayed from this file.

SYNCHRONIZE WINDOWS

When you “play through” the sequence of occurrences of a byte in flow view, if this checkbox is elected, the play in flow view is synchronized and linked with the action in the main SavvyCAN serial communications panel, AND with the play through in the GRAPH VIEW screen. So that at any particular moment, all three views are displaying the same value. This lets you STOP the action at any particular point in the graph view of the entire capture, and actually see the values of the message ID under scrutiny in flow view.

GRAPH BY TIMESTAMP

Normally, the frames for any particular message ID are numbered for this view and indeed on the left we display the current frame number and total number of message frames of that ID. The small graph in the lower right hand side of the panel normally graphs the 7 data bytes by frame number. By clicking this, the graph will show TIME along the horizontal access instead, making it easier to compare our location with the GRAPH function elsewhere or the main data view. Shows graph of each data byte by time stamp of time message received.

AUTOREFERENCE

Allows you to either list the initial frame from the file or the previous frame displayed as the reference frame which you are comparing current frame to.

The central display area of the screen shows the reference frame and the current frame with hexadecimal values for each current byte. These are updated as the frames are played.

DATA SEEK VALUES

Data seek values allows you to set seek points in the series of messages to stop the playback. The playback will run until that specific value is found in that byte. At that point it will stop. You can then change speeds or make other adjustments and then continue.

DATA BYTE COLORS

Lists the colors used to graph each data byte in the graph display to the right. This running graph updates as the frames are viewed in their "flow".

BITS

Bits displays each data byte from 0 to 7 in the data payload. It further breaks out each bit of those bytes as bit 0-7. This forms a 64-bit display grid.

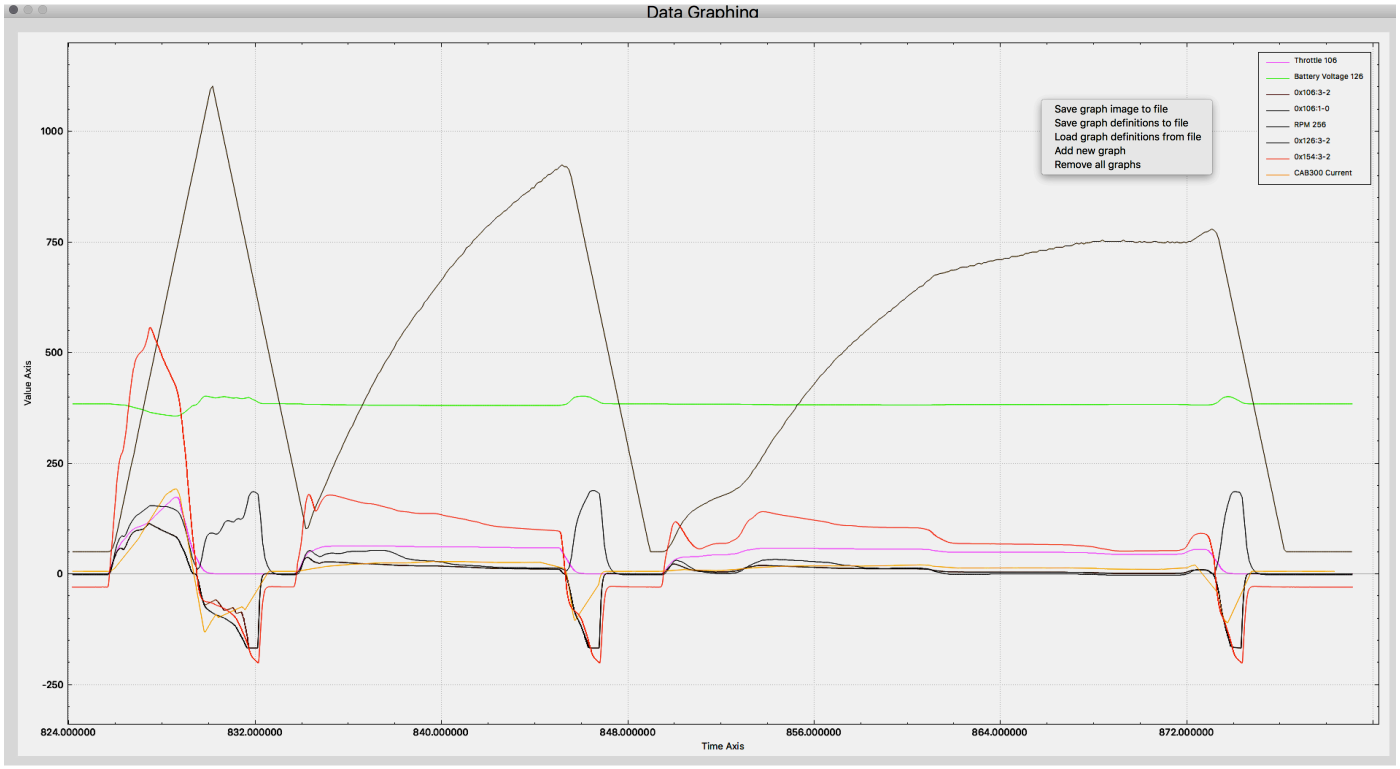
Bits set in the FIRST frame of this message ID are displayed in black.

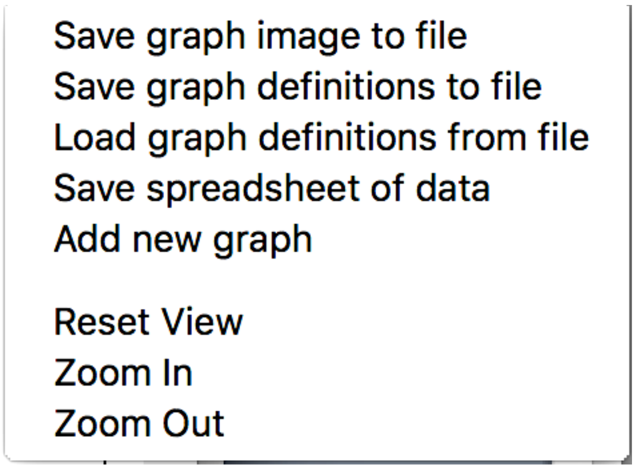
Red denotes bits that WERE black,, but are currently reset (0).

Green denotes bits that WERE NOT set initially, but now ARE.

GRAPH DATA

Data graphing is a powerful analysis tool allowing you to graph data values over time. Better, you can graph several of them over the same time frame. This lets you examine relationships between different data as they interact. For example, as you see your torque increase, indicating a large demand for current, you may notice a smaller dip in another value that is concurrent. That value may turn out to be voltage. But only by seeing the two interact can you make that determination.




Save graph image to file
Save graph definitions to file
Load graph definitions from file
Save spreadsheet of data
Add new graph

Reset View
Zoom In
Zoom Out

To create a graph, press CNTRL and click on the graph screen to bring up the graph menu.

This screen allows you to do a number of things. You can load and save a graph definitions file containing your graph definitions.

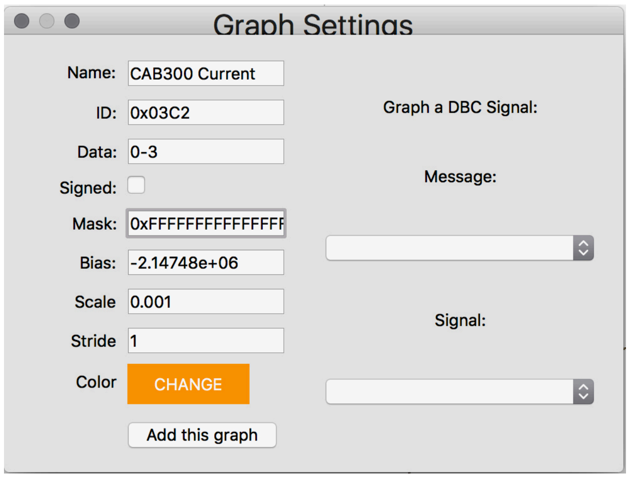
You can also save a graph IMAGE file of the graphic image as it appears on screen.

You can save a spreadsheet of the data used on the graph.

But most importantly it allows you to add a new graph.

This is a new data value which will be graphed on the form on screen. And you can add any number of these to the screen.

When you select ADD NEW GRAPH, you will see a new control box titled GRAPH SETTINGS. This is where you enter values defining what and how to graph data.



Name: CAB300 Current
ID: 0x03C2
Data: 0-3
Signed:
Mask: 0xFFFFFFFFFFFFFFFF
Bias: -2.14748e+06
Scale 0.001
Stride 1
Color

NAME is simply the name of the value shown in the legend.

ID is the message ID of the CAN message you wish to graph.

DATA is the crucial element. It defines the byte or bytes to be graphed. This is a value between 0 and 7 indicating a byte containing a value of interest.

You can enter multibyte values as in 2-3 where bytes two and three are treated as a 16-bit integer. In the case of LSB/MSB values, you can enter 3-2 to indicate that.

SIGNED indicates if you want to treat this as a signed integer or unsigned integer.

MASK allows you to mask off individual bits of the value to ignore.

BIAS allows you to offset the graph vertically from the zero origin line.

SCALE allows you to multiply or divide the value by any number – effectively scaling the size of the waveform.

STRIDE allows you to graph every nth data point instead of each one. For example, enter 10 to graph every 10th message frame of that ID. For some large data sets or messages that occur very frequently, this can declutter the display and improve performance on large data sets.

And **COLOR** will allow you to define what color the graph line will appear.

You can click on the timeline below to expand or contract the graph in time. And you can click on the values to the left of the graph to expand or contract the range of values depicted.

You can also click/drag on the central graph area to zoom BOTH time and value in and out.

In this way, you can add multiple data elements from the same or different CAN messages all to the same graph. You can zoom in and out and examine their relationships in value and time.

And you can save your definitions or even an image of the graph to a file. You can load the definitions later to graph the same data from a different log file.

FRAME DATA ANALYSIS

Select Frame Data Analysis from the RE TOOLS menu to call up the Detailed Frame Information screen.

This screen provides statistical data on each specific CAN message received.

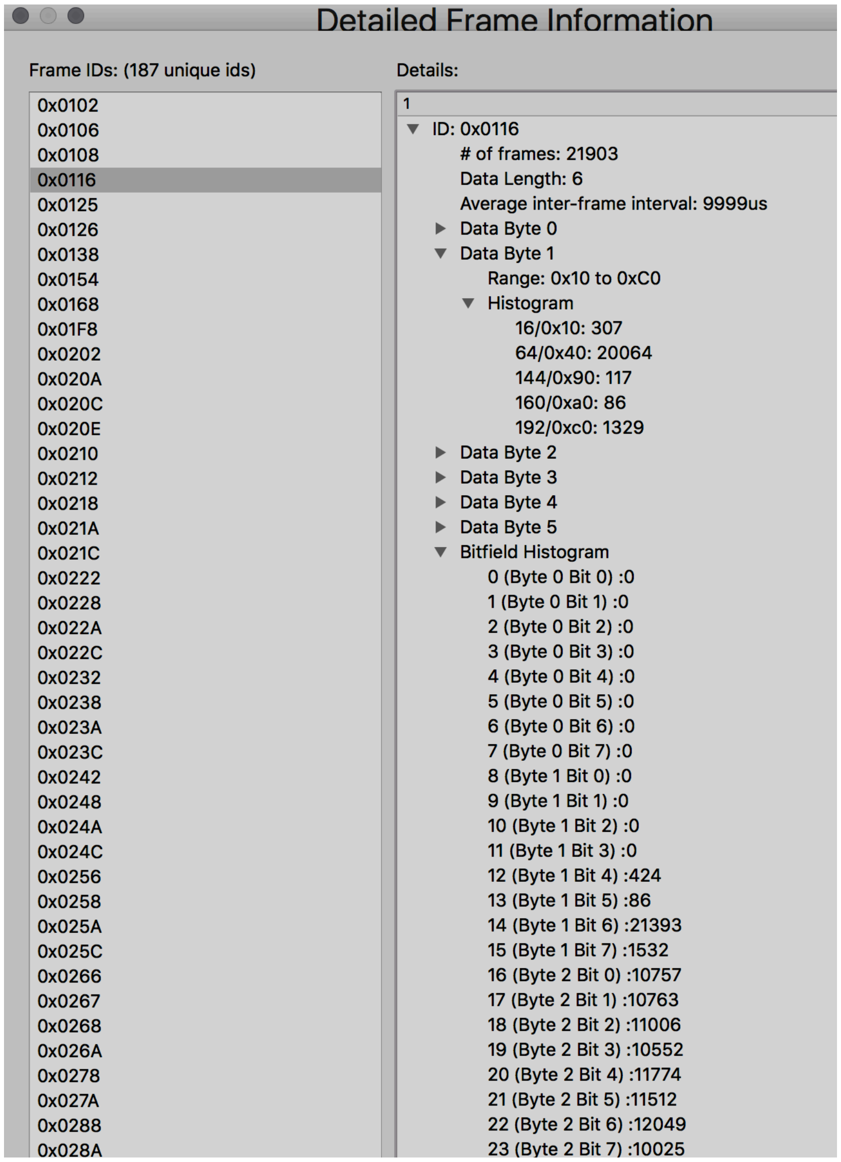
All received message IDS are listed on the left under the Frame IDs and indeed this panel displays the total number of unique message IDS received in this capture or data log.

Highlight any message ID to display statistical details in the right hand panel.

In this example, we highlight message ID 116 and see that the capture log contains 21,903 instances of this message, that the data length in the 116 message is 6 bytes, and that the average time interval between receipts of this message is 9999 microseconds or about one message ever 10 milliseconds.

If we expand Data Byte 1 we learn that the value in byte 1 ranges from 0x10 to 0xC0. And a histogram provides the number of times each value appears in the data log.

The most common value is 0x40 which appears in 20,064 messages.


Detailed Frame Information

Frame IDs: (187 unique ids)

```
0x0102
0x0106
0x0108
0x0116
0x0125
0x0126
0x0138
0x0154
0x0168
0x01F8
0x0202
0x020A
0x020C
0x020E
0x0210
0x0212
0x0218
0x021A
0x021C
0x0222
0x0228
0x022A
0x022C
0x0232
0x0238
0x023A
0x023C
0x0242
0x0248
0x024A
0x024C
0x0256
0x0258
0x025A
0x025C
0x0266
0x0267
0x0268
0x026A
0x0278
0x027A
0x0288
0x028A
```

Details:

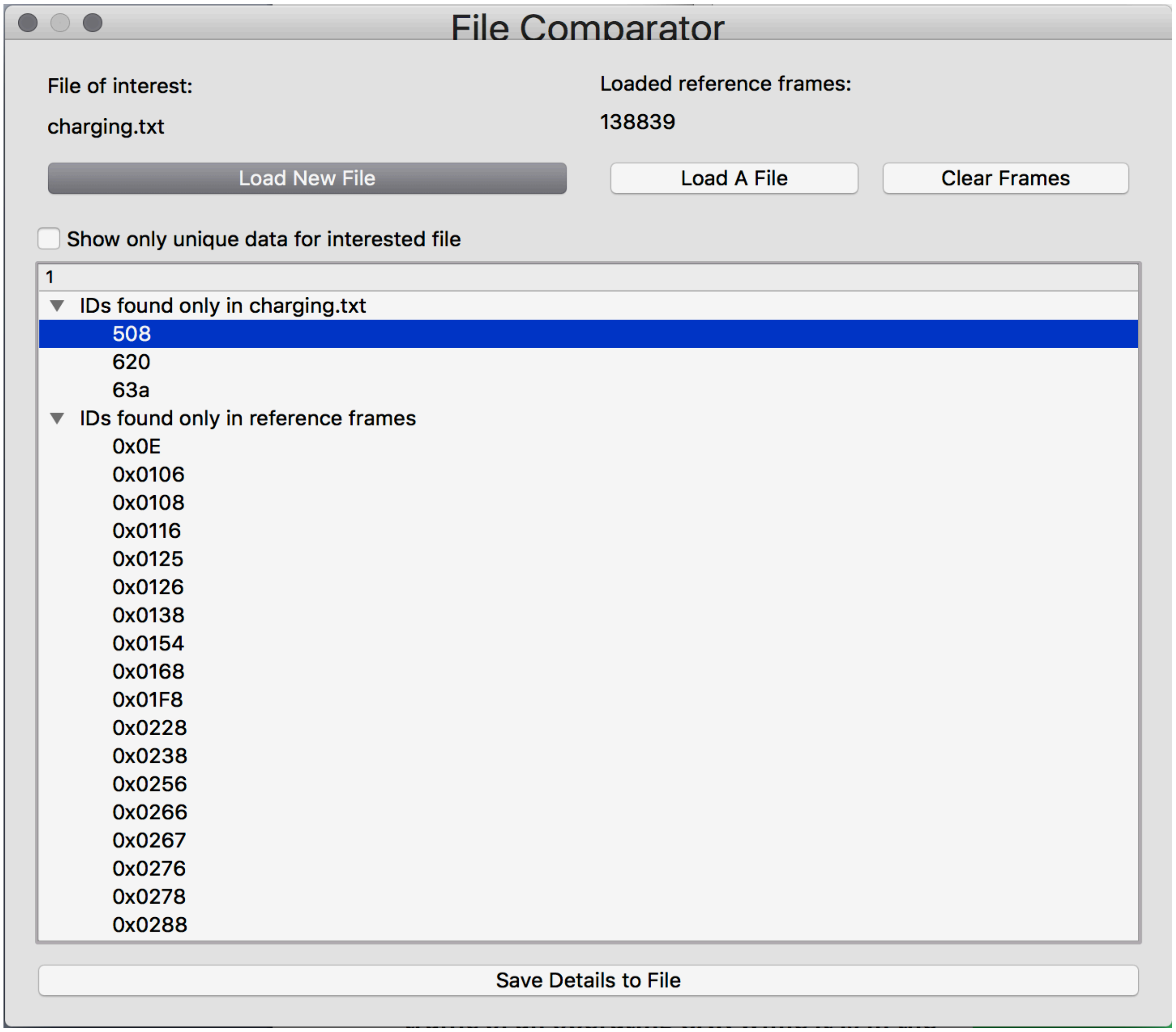
```
1
▼ ID: 0x0116
    # of frames: 21903
    Data Length: 6
    Average inter-frame interval: 9999us
  ▶ Data Byte 0
  ▼ Data Byte 1
      Range: 0x10 to 0xC0
    ▼ Histogram
        16/0x10: 307
        64/0x40: 20064
        144/0x90: 117
        160/0xa0: 86
        192/0xc0: 1329
  ▶ Data Byte 2
  ▶ Data Byte 3
  ▶ Data Byte 4
  ▶ Data Byte 5
  ▼ Bitfield Histogram
      0 (Byte 0 Bit 0) :0
      1 (Byte 0 Bit 1) :0
      2 (Byte 0 Bit 2) :0
      3 (Byte 0 Bit 3) :0
      4 (Byte 0 Bit 4) :0
      5 (Byte 0 Bit 5) :0
      6 (Byte 0 Bit 6) :0
      7 (Byte 0 Bit 7) :0
      8 (Byte 1 Bit 0) :0
      9 (Byte 1 Bit 1) :0
      10 (Byte 1 Bit 2) :0
      11 (Byte 1 Bit 3) :0
      12 (Byte 1 Bit 4) :424
      13 (Byte 1 Bit 5) :86
      14 (Byte 1 Bit 6) :21393
      15 (Byte 1 Bit 7) :1532
      16 (Byte 2 Bit 0) :10757
      17 (Byte 2 Bit 1) :10763
      18 (Byte 2 Bit 2) :11006
      19 (Byte 2 Bit 3) :10552
      20 (Byte 2 Bit 4) :11774
      21 (Byte 2 Bit 5) :11512
      22 (Byte 2 Bit 6) :12049
      23 (Byte 2 Bit 7) :10025
```

A Bitfield Histogram actually shows the number of times each bit of the data payload is true. We see no bits in Byte 0 are ever lit, but bit 22 (byte 2, bit 6) is set 12,049 times in the 21,903 messages received.

FILE COMPARISON

File comparison is a powerful function allowing you to quickly compare two data logs and determine what message IDs they have in common, or conversely, which message IDs are unique to one or the other of the two files.

Lets assume we have done a CAN data capture of a car NOT charging and then the same car charging. We might want to see what new message IDs show up once we go into charge mode.



File of interest: charging.txt

Loaded reference frames: 138839

Show only unique data for interested file

```
1
▼ IDs found only in charging.txt
      508
      620
      63a
▼ IDs found only in reference frames
      0x0E
      0x0106
      0x0108
      0x0116
      0x0125
      0x0126
      0x0138
      0x0154
      0x0168
      0x01F8
      0x0228
      0x0238
      0x0256
      0x0266
      0x0267
      0x0276
      0x0278
      0x0288
```

Our first order of business is to load our reference file. Click LOAD A FILE.

At this point LOADED REFERENCE FRAMES should give us an indication of the total number of reference frames in our file.

Click LOAD A NEW FILE and select a file to compare. The file name will be listed as the File of Interest.

At this point, we will see three expandable functions listed in the main screen.

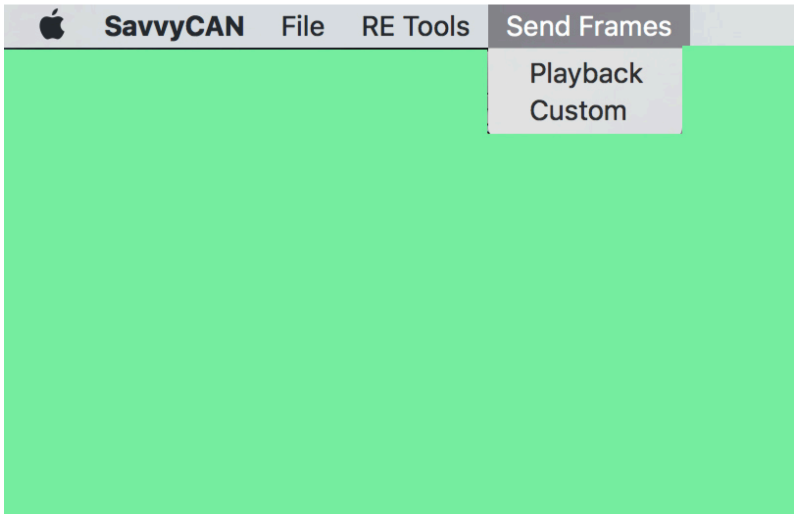
IDs found only in file of interest.
IDS found only in reference frames.

IDS found in both.

We can see that we have three new message IDs that appear in our charging.txt log that do not appear in our notcharging.txt reference file. They are 0x508, 0x620, and 0x63A. Those message IDs might bear further examination if we were seeking clues to the charging process in CAN data.

Finally SAVE DETAILS TO FILE will allow us to save this analysis to an external file for use elsewhere.

SENDING FRAMES – SIMULATING THE SYSTEM



In our work reverse engineering the CAN control of various OEM electric vehicle components, the central approach is to FIRST record an actual CAN bus message traffic of an operating CAR while it is in the act of doing what we want the component to do.

We can then play this recording back to a standalone component, and it should respond exactly the way it would if it were installed in the vehicle.

We basically then discard individual messages from the recording, and see if it will still do it.

Gradually we discard ALL the messages until we get down to JUST those messages required to get the component to do the task.

This allows us to focus our reverse engineering and analysis on JUST those messages necessary to operate the device – ignoring all others. For most devices, this is two or three messages but in some cases up to a dozen we send, and perhaps a similar number it responds with.

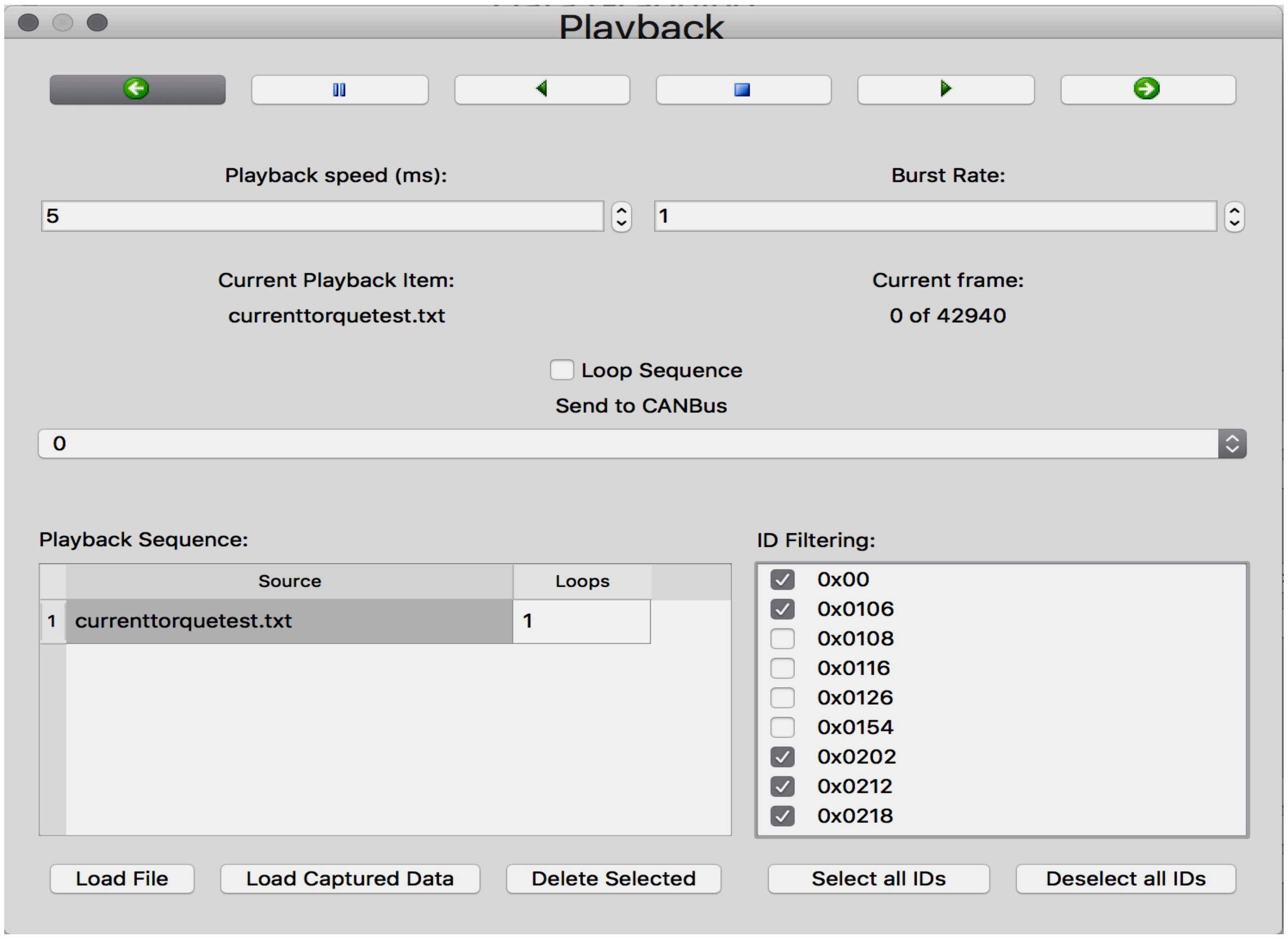
SavvyCAN provides two functions just for this. PLAYBACK and CUSTOM. PLAYBACK is oriented toward taking an entire log file, filtering some messages perhaps, and playing it back like a recording.

CUSTOM is more a function to design and send individual frames. You could for example, hookup SavvyCAN to an operating vehicle, and have it periodically transmit a single frame containing some data that changes the operation of the vehicle. It doesn't send an entire log file, just periodically inserts a message.

PLAYBACK

As described, PLAYBACK is provided to allow you to "play back" a previous data capture much as you would a tape recording. It simply puts the same messages back on a bus in the same order they were originally received.

The devil being in the details, it also allows you to modify this playback to some degree.



Playback speed (ms): 5

Burst Rate: 1

Current Playback Item: currenttorquetest.txt

Current frame: 0 of 42940

Loop Sequence

Send to CANBus: 0

Playback Sequence:

	Source	Loops
1	currenttorquetest.txt	1

ID Filtering:

- [✓] 0x00
- [✓] 0x0106
- [ ] 0x0108
- [ ] 0x0116
- [ ] 0x0126
- [ ] 0x0154
- [✓] 0x0202
- [✓] 0x0212
- [✓] 0x0218

At the top of the playback screen are our playback controls – much like a video camera or tape player. From left to right,

– BACK steps backward one frame.

PAUSE – pauses the playback

REVERSE – plays backwards from current position

STOP/RESET – ends playback and returns us to frame one

PLAY – plays the current file forward

FORWARD ONE FRAME – sends the next frame in single step fashion

PLAYBACK SPEED AND BURST RATE

Playback speed and burst rate allow you to modify the speed at which frames are sent. Speed is normally a value in milliseconds between transmissions. Burst rate is more how many frames are sent during that transmission. Using these two variables, you can modulate the rate the CAN file is sent out on the bus. This can be quite important as timing is, as always, everthing. It is quite common for equipment to receive frames every so many milliseconds. If they do not receive them before the timout, they simply shut down and quit operating.

CURRENT PLAYBACK ITEM

This simply lists the file we are playing back.

CURRENT FRAME

This displays which frame we are on. For example 1215 of 42940 and gives us some idea of where we are at in the file. If for example, we know we did not put on the brake and put it in DRIVE until frame 2145 of the capture, we can detect that point in the process by observing this field.

LOOP SEQUENCE

This is a checkbox that determines what happens when we get to the end of the capture file. If it is unchecked, we simply stop. If it is checked, we immediately go to the first frame and resume transmission there. In this way, our file plays as a continuous loop.

SEND TO CAN BUS

This drop down menu allows us to select which CAN bus port the data goes out on,. One option is none.

PLAYBACK SEQUENCE

This allows actually quite complex playbacks using mulitiple files. And indeed it can do variable passes in those files. You could for example go through two iterations of file 1, a single iteration of file 2, and then three iterations of file three if desired.

ID FILTERING

The star of the playback screen is of course, again the ability to filter messages. This filter operates exactly as our display filter on the main capture screen, but in this case, applying to the messages transmitted onto the bus.

This is the most powerful function of SavvyCAN actually. To take a recording of CAN message traffic, transmit it onto a bus, and gradually decrease the messages sent until you only have the IMPORTANT ones for your task.

SELECT ALL IDS and DESELECT ALL IDS makes this easier. But again, the basics are that if you check a box, the message gets sent, if no checkbox, it does NOT get sent.

And note that you can do this WHILE a playback is occurring. So deselecting a message while transmitting a file simply discontinues sending that message going forward.

CUSTOM

Selecting CUSTOM from the top bar menu calls up the FRAME SENDER screen.

Frame sender allows you to quickly design and send custom CAN message frames onto the bus.

EN

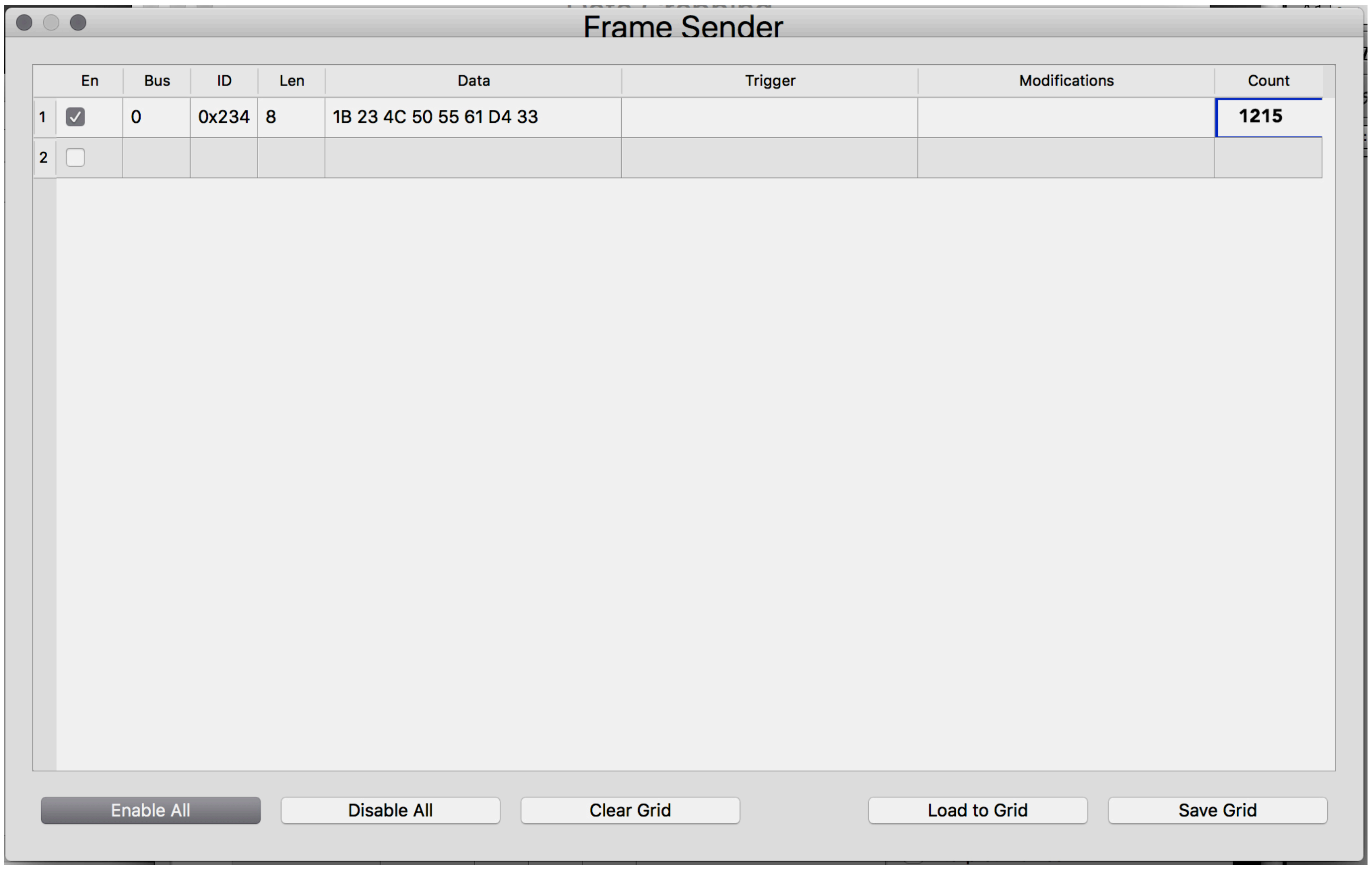
Enable. If this is checked, the program starts sending this message periodically. If you remove the checkbox, it discontinues.

BUS

Again, you can output the frame on any bus, typically 0 or 1.

ID

The message identification number of the message you want to simulate/send. Either 11-bit or 29-bit.



	En	Bus	ID	Len	Data	Trigger	Modifications	Count
1	✓	0	0x234	8	1B 23 4C 50 55 61 D4 33			1215
2	☐							

LEN

Length in bytes of data payload. Up to 8.

DATA

The data bytes you want to send.

TRIGGER

Triggers are a powerful and very flexible means to determine WHEN the described frame is sent out on the bus. And there are several conditions you can use to trigger this transmission.

- **`ms`**
- **`x`**
- **`id`**
- **`bus`**

`ms`

`ms` is the most basic form of trigger. It simply specifies that the defined CAN frame go out every x milliseconds. So **`40ms`** in this field would indicate to send the CAN frame every 40 milliseconds.

`x`

This is really a modifier to **`ms`**. It specifies the maximum number of times the message should be sent. So **`40ms 100x`** would indicate that the CAN frame should be sent every 40ms until 100 frames have been sent then terminate.

`id`

`id` is a trigger based on an INCOMING message frame. You can specify any message id and the described frame will be sent whenever that message is received.

This function also modifies the use of **`ms`** and **`x.`** If you specify an **`id`**, then **`ms`** is the time delay between when the message is received and when the described message is transmitted. **`x`** remains the maximum number of times this happens.

`Id0x222 40ms 100x` would cause the frame to be sent 40 milliseconds after 0x222 was received, but this would occur on the first 100 such messages received.

bus

`bus` is a modifier for **`id`**. In this way, you can specify that not only the message ID has to be correct, but it has to come in on the correct bus.

`Id0x222 bus0 40ms 10x` would then transmit 40 milliseconds after message ID 222 comes in on bus 0, but only for the first 10 instances. A 222 message incoming on bus1 would be ignored entirely.

These values can also be compounded in multiple triggers separated by commas:

`Id0x222 40ms, id0x111 10ms, 1000ms`

This would transmit 40 milliseconds after a 222 message was received, but also 10 milliseconds after a 111 message was received. And in any case, it is going to transmit once every 1000 milliseconds (once per second) whether or not anything is received.

MODIFICATIONS

Modifications is again a powerful and flexible way of customizing our transmissions mathematically. Let's assume that the eight data bytes in our DATA section are numbered from 0 to 7 with the last data byte on the right being 7. In modifications, we will refer to these eight bytes as D0-D7.

Modifications allow us to perform mathematical operations on these eight data bytes. The operations include:

```
+  Addition
- Subtraction
* Multiplication
/ Division
&  bitfield operation AND
| bitfield operation OR
^ bitfield operation XOR
```

So for example: **`D0=D0*4`** would replace the value in data byte 7 with that value times 4.

Or it could be a simple replacement: **`D5=0xF3`**. Or **`D5=D3+0xF3`**

These examples look more or less like nonsense. After all, we can put anything we like in the data fields anyway. But we can also use the value of incoming CAN message frames in our calculations and that makes it slightly more interesting**.** We do this with the **`ID:`** and **`BUS:`** commands.

```
D3=BUS:0:ID:0x222:D3
```

In this example, we are copying data byte 3 from any message 0x222 arriving on bus 0 into our own D3 data byte.

These can become quite exotic:

```
D3=BUS:0:ID:0x222:D3 + BUS:1:ID:0x123:D7 / 4
```

In this case, we take data byte 3 from message 222 arriving on bus 0, sum it with data byte 7 of message 123 arrriving on bus1, and divide the result by 4. This calculated value is placed in our data byte 3 of the frame we are transmitting.

PUTTING TRIGGERS AND MODIFICATIONS TOGETHER

A simple example illustrates how powerful this can be. Message 222 on bus 0 carries motor coolant temperature in data byte four. This is on the bus going from the Vehicle Control Unit to an instrument cluster that displays it on a gage. We are going to SEND a message 0x222 of our own.

TRIGGER: **`id0x222 bus0`**
This indicates that we are going to send our frame immediately on receipt of a message ID 0x222 on bus 0.

```
MODIFICATIONS: D0=BUS:0:ID:0x222:D0, D1=BUS:0:ID:0x222:D1,
D2=BUS:0:ID:0x222:D2, D3=BUS:0:ID:0x222:D3, D4=BUS:0:ID:0x222:D4 *2,
D5=BUS:0:ID:0x222:D5, D6=BUS:0:ID:0x222:D6, D7=BUS:0:ID:0x222:D7
```

Although this is somewhat long, we are really copying all the data out of the received message 222 into our own replacement message 222 and sending it back out immediately. But in the process, coolant temperature in data byte 4 gets multiplied by 2 – essentially doubled.

The instrument cluster receives the message from the VCU, but every time it does, it immediately receives one from SavvyCAN doubling the value. And so the instrument cluster shows a coolant temperature twice as high as it actually is measured by the vehicle control unit.

THE GRID

You can enter actually a number of different messages on the screen in what we refer to as a GRID. Individual messages can be enabled or disabled using the EN field at different times and for different purposes.

At the bottom of the screen are several buttons for managing this grid.

ENABLE ALL

Enables all messages in the grid.

DISABLE ALL

Unchecks EN for all messages in the grid.

CLEAR GRID

Eliminates all current messages from grid.

SAVE GRID

Saves the current grid to an **.fsd** file for later use.

LOAD GRID

Loads grid from a previously saved **.fsd** file.