1 Introduction

This document gives an introduction to CAN bus and explains how to connect to CAN bus devices using the CANTool plugin in YourDyno.

This document is relevant for YourDyno Standard v4 only. CAN Bus handling is built into the YourDyno Ultimate and an option for YourDyno Standard v5, so the CANTool plugin is unnecessary for these units.

2 What is the CAN bus used for

Connection to the CAN bus to a Dyno DAQ is used for:

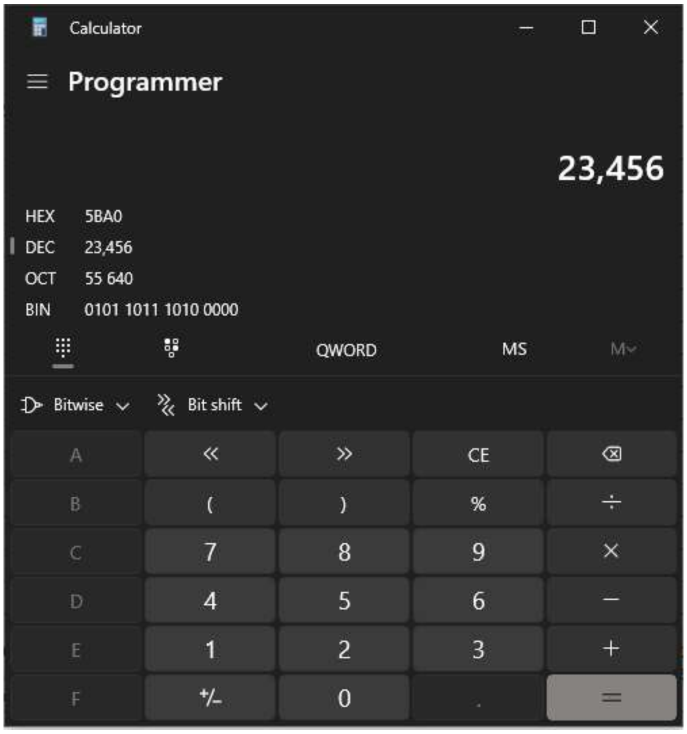
1) Read data channels directly from the engine's ECU. Most aftermarket ECUs has a CAN out option where you define which channels the ECU shall publish and the data format.
2) Read data from off the shelf CAN bus sensor modules
3) Send dyno data on the CAN bus for processing by other devices such as a data logger or an ECU

3 CAN bus format

To read a CAN bus data channel you need the data format of that channel. This must be available some fashion, otherwise you cannot make sense of the data. The data format is often but configurable. For example you can often choose the CAN ids, which need to be unique on the CAN bus.

Here is what you need:

1) CAN bus speed, often 500kBit/s or 1Mbit/s
2) CAN ID. This is the ID of the data packets that belong to the data you are interested in. The IDs are typically given in hexadecimal format, indicated by 0x1234 or 1234h. The numbers are written in hex format in YourDyno as well, so you don't need to convert the numbers into decimal format. If you do need to convert between hex and decimal then the Windows calculator can do that, see below:



23,456

HEX 5BA0
DEC 23,456
OCT 55 640
BIN 0101 1011 1010 0000

CAN ID can be either 11 bit or 29 bit, both are supported.

3) The CAN ID is the ID of up to 8 bytes of data. This may be 8 data channels of one byte each, but typically one data channel consists of more than one byte. 2 is common. It can represent a number from 0-65535 so it gives good resolution. If all channels are 2 bytes, each CAN ID can consist of up to 4 data channels. To individually address each of the data channels the term Byte offset is used. 0 is the first data channel, the next channel is at Byte offset 2 then 4 then 6 if all channels have 2-byte length.
4) Each data channel is converted into its value by this formula:
   Value = (CAN data + Offset) * Multiplier / Divisor
   Each of the three Offset, Multiplier and Divisor are entered into YourDyno so it can translate the CAN data into real sensor or data value

4 YourDyno CANTool usage

4.1 Canalyst-ii

To use the CANTool plugin you need a CAN to USB adapter. The unit called Canalyst ii is compatible.

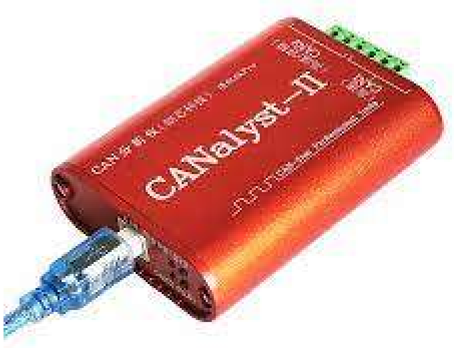
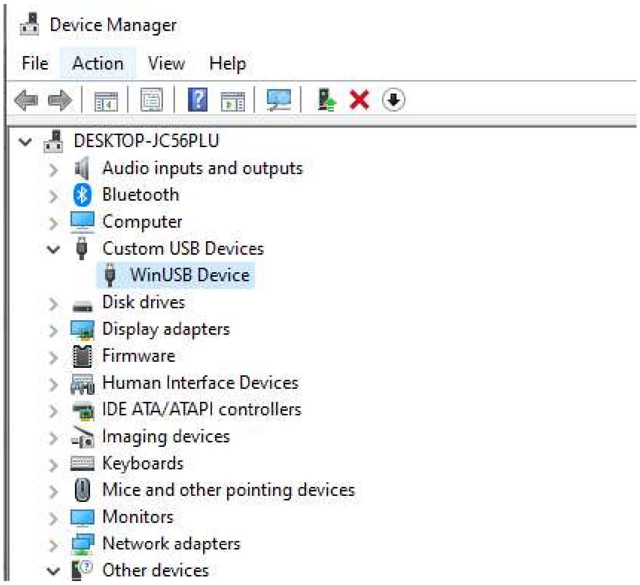


Figure 1 Canalyst-ii module

This is available off the shelf many places like Amazon.

Install the software https://yourdyno.com/USB_CAN%20TOOLSetup(V2.12).exe

Connect the Canalyst-ii module to the PC and go to Device Manager. You will find an unknown USB device. Right click and select Update Driver and select Browse my computer for drivers. Select C:\Program Files (x86)\. It will search and install the driver. It should look like this:



- DESKTOP-JC56PLU
  - Audio inputs and outputs
  - Bluetooth
  - Computer
  - Custom USB Devices
    - WinUSB Device
  - Disk drives
  - Display adapters
  - Firmware
  - Human Interface Devices
  - IDE ATA/ATAPI controllers
  - Imaging devices
  - Keyboards
  - Mice and other pointing devices
  - Monitors
  - Network adapters
  - Other devices

Figure 2 WinUSB Device indicates the successful installation of the driver

4.2 Physical connections

The Canalyst-ii is connected to the physical CAN bus (CAN-L and CAN-H wires) of CAN1. Turn on the R1 resistor with the dip switch.

4.3 CANTool software setup

Make sure the CANTool plugin was installed when you installed YourDyno. If not, reinstall and select CANTool. Do not attempt to install the plugin manually, use the YourDyno installer.

4.3.1 Connecting

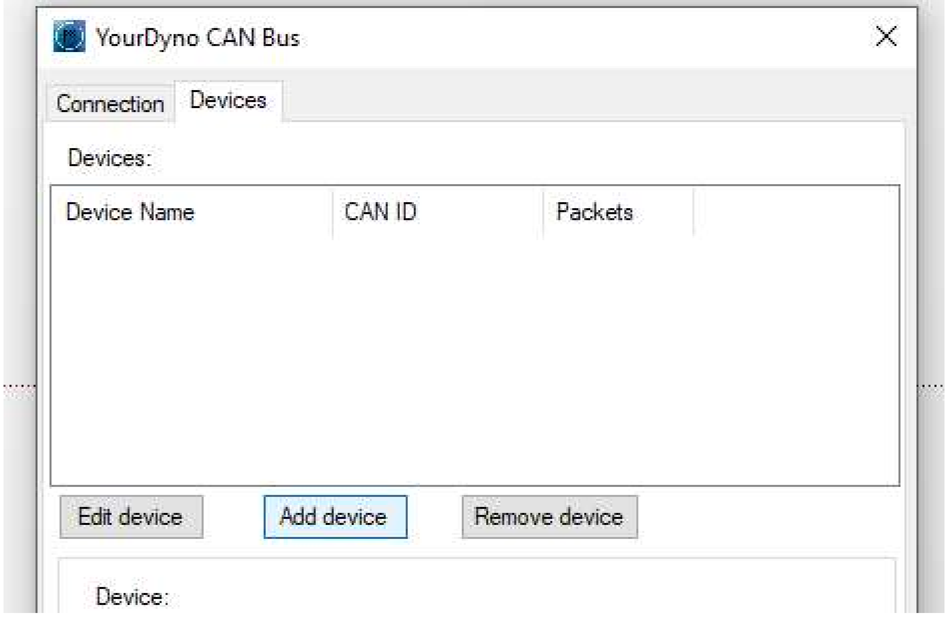
1) Select CANBUS 0 (corresponds to CAN1 connector), the correct Baudrate and press Connect.
2) Click the Show Debug option. You should see data coming in the window below. If not, the baud rate is wrong, CAN L/H is wrong, the CAN device is not powered on etc.
3) Turn off Show Debug as this consumes unnecessary CPU.

In the window you can see the raw CAN data. You can see which CAN IDs you are receiving as well as the data, but you will not be able to make sense of the data.

4.3.2 Defining devices data channels

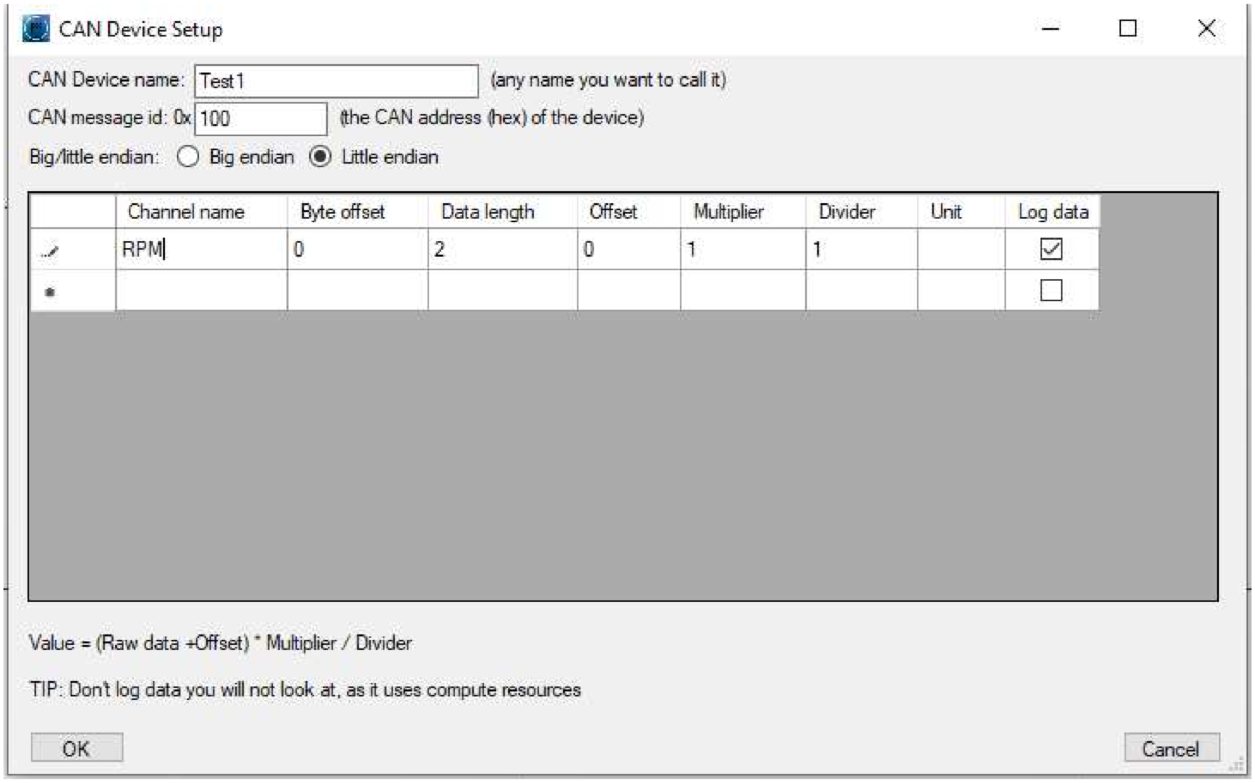
Each CAN ID is a Device and each data channel in that ID is a channel.

1) Click Devices and click Add device



2) Name the device something
3) Enter the CAN ID in hexadecimal

4) Enter a channel name, byte offset, data length, Offset, Multiplier, Divider and Unit. Click Log data



CAN Device name: Test1 (any name you want to call it)

CAN message id: 0x 100 (the CAN address (hex) of the device)

Big/little endian: ○ Big endian ● Little endian

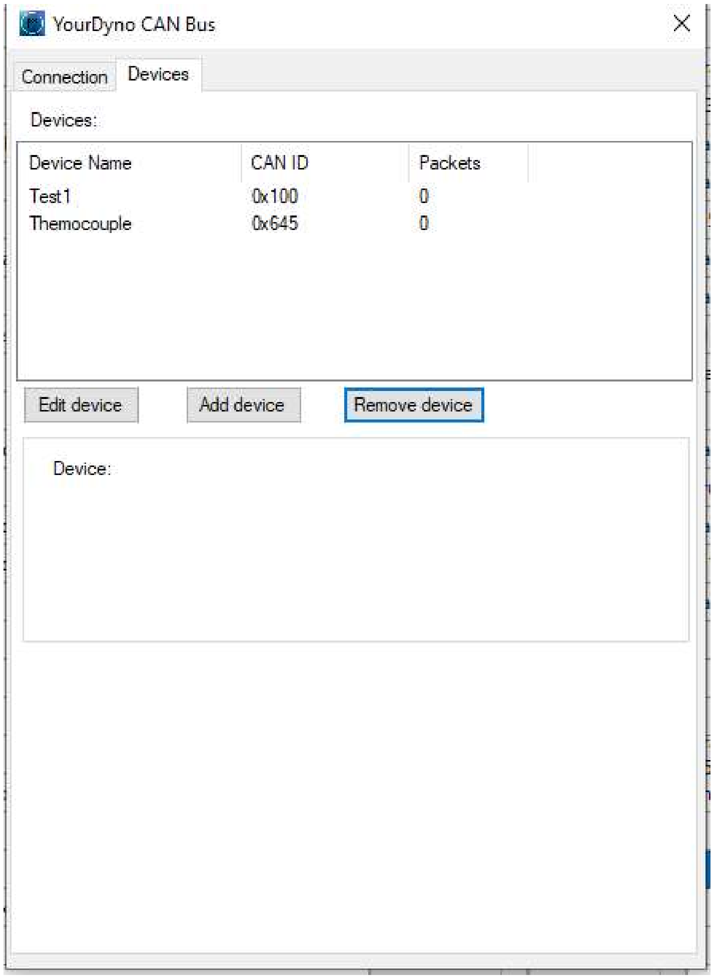
	Channel name	Byte offset	Data length	Offset	Multiplier	Divider	Unit	Log data
	RPM	0	2	0	1	1		☑
*								□

Value = (Raw data +Offset) * Multiplier / Divider

TIP: Don't log data you will not look at, as it uses compute resources

5) Click ok

You should see the packet count increase (not 0).



Device Name	CAN ID	Packets
Test1	0x100	0
Themocouple	0x645	0

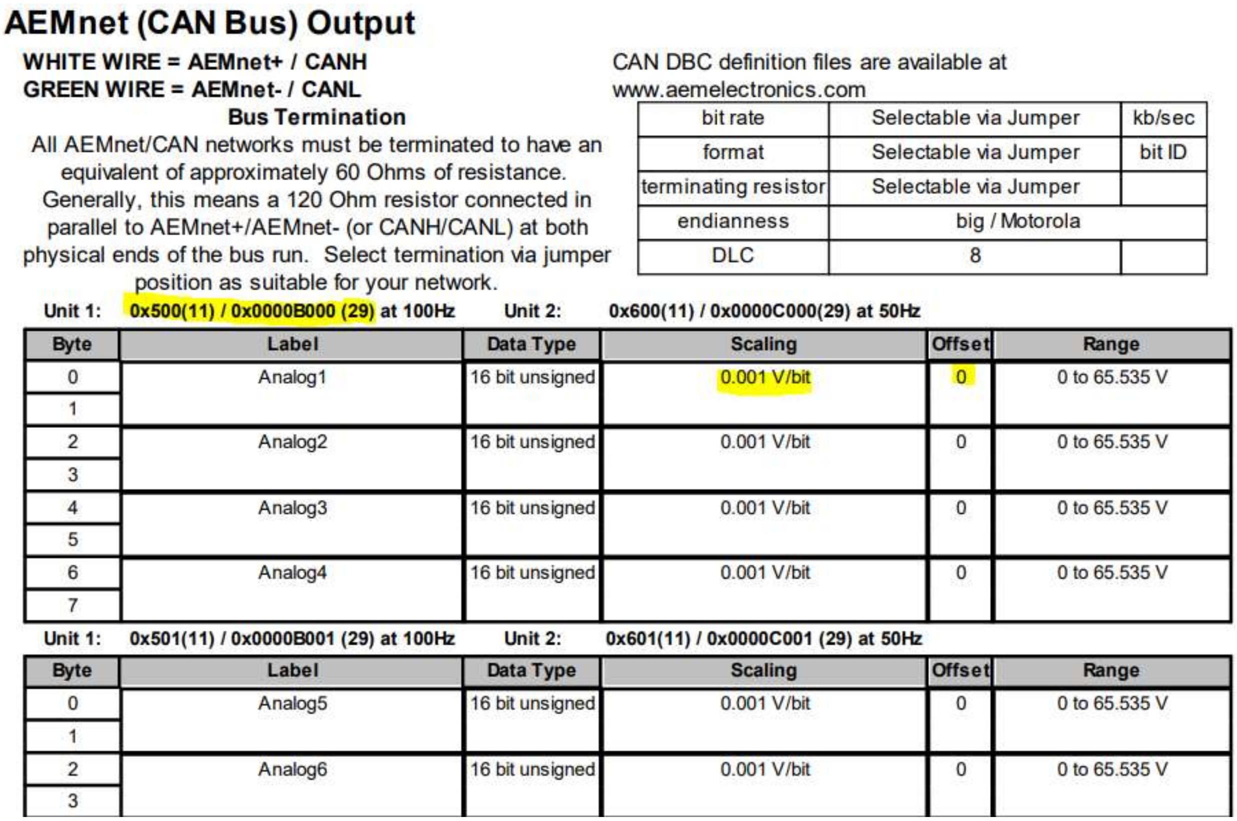
Click on the device to see data in the different data channels in the device. If it is wildly wrong, the typical problem is that it is big-endian instead of little-endian or the other way around. Just try the two options if you are unsure.

The data channels are now available.

5 Examples

5.1 AEM 22-channel CAN sensor module.

The specification of this module gives the following (beginning only):



AEMnet (CAN Bus) Output

WHITE WIRE = AEMnet+ / CANH
GREEN WIRE = AEMnet- / CANL

Bus Termination

All AEMnet/CAN networks must be terminated to have an equivalent of approximately 60 Ohms of resistance. Generally, this means a 120 Ohm resistor connected in parallel to AEMnet+/AEMnet- (or CANH/CANL) at both physical ends of the bus run. Select termination via jumper position as suitable for your network.

CAN DBC definition files are available at www.aemelectronics.com

bit rate	Selectable via Jumper	kb/sec
format	Selectable via Jumper	bit ID
terminating resistor	Selectable via Jumper	
endianness	big / Motorola	
DLC	8	

Unit 1: 0x500(11) / 0x0000B000 (29) at 100Hz Unit 2: 0x600(11) / 0x0000C000(29) at 50Hz

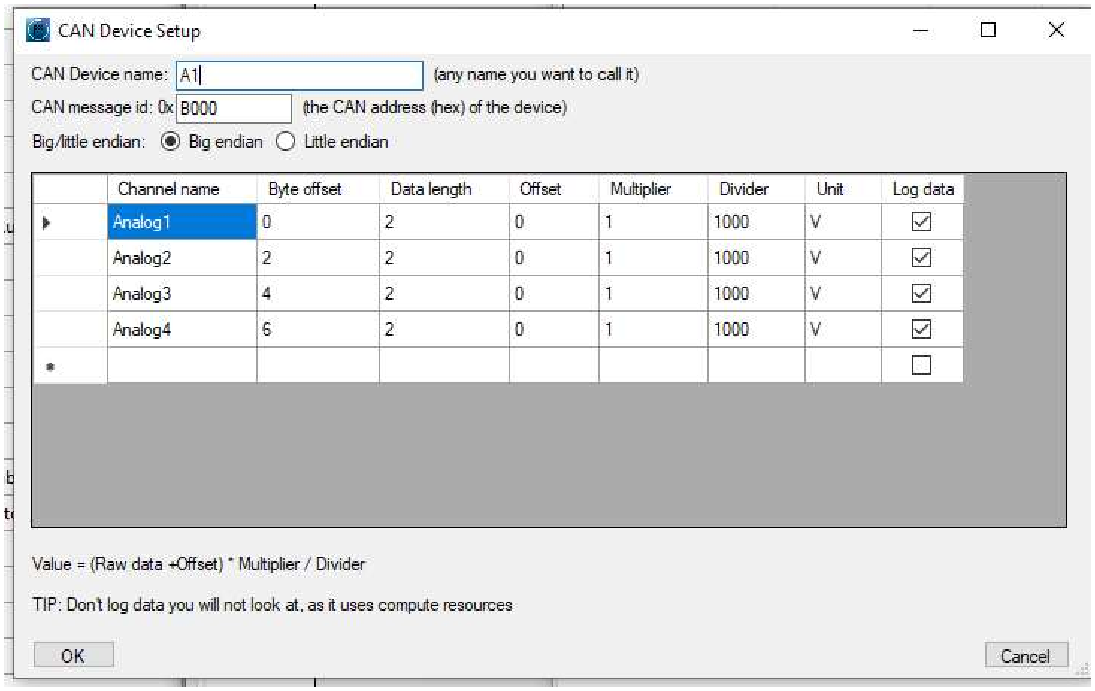
Byte	Label	Data Type	Scaling	Offset	Range
0	Analog1	16 bit unsigned	0.001 V/bit	0	0 to 65.535 V
1					
2	Analog2	16 bit unsigned	0.001 V/bit	0	0 to 65.535 V
3					
4	Analog3	16 bit unsigned	0.001 V/bit	0	0 to 65.535 V
5					
6	Analog4	16 bit unsigned	0.001 V/bit	0	0 to 65.535 V
7					

Unit 1: 0x501(11) / 0x0000B001 (29) at 100Hz Unit 2: 0x601(11) / 0x0000C001 (29) at 50Hz

Byte	Label	Data Type	Scaling	Offset	Range
0	Analog5	16 bit unsigned	0.001 V/bit	0	0 to 65.535 V
1					
2	Analog6	16 bit unsigned	0.001 V/bit	0	0 to 65.535 V
3					

Figure 3 AEM 22 channel sensor module CAN specification

Set up with 29 bit ID, 500kBit/s, this is how to define the first 4 analog inputs. Scaling 0.001V/bit means we need to divide by 1000 to get volt.



CAN Device Setup

CAN Device name: A1 (any name you want to call it)

CAN message id: 0x B000 (the CAN address (hex) of the device)

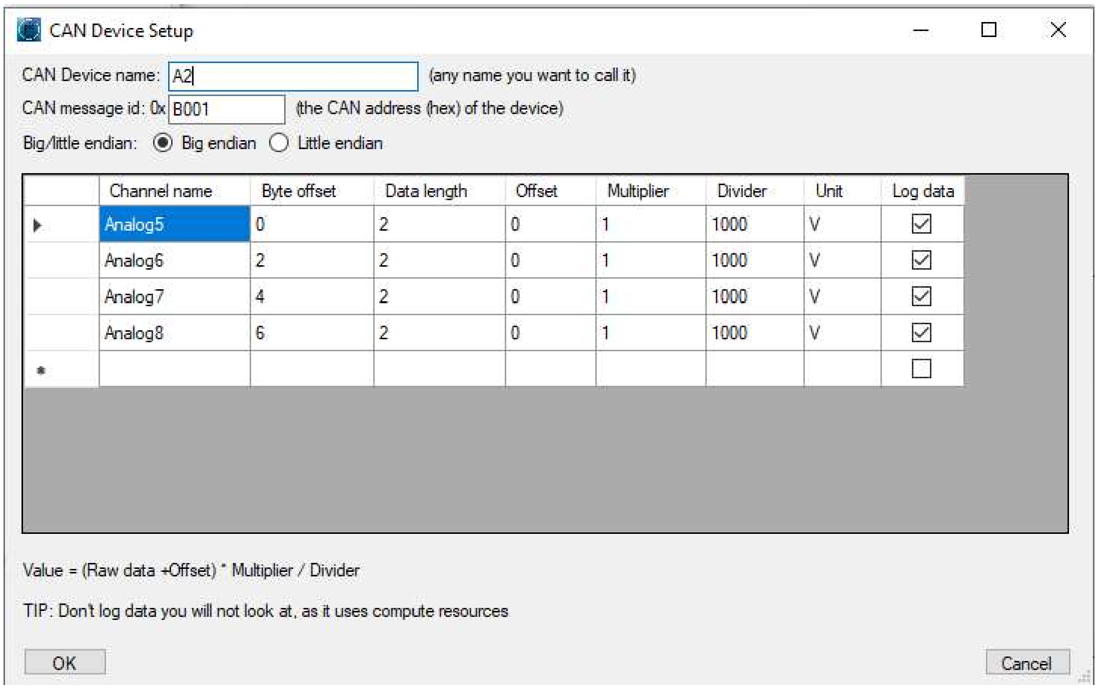
Big/little endian: ● Big endian ○ Little endian

Channel name	Byte offset	Data length	Offset	Multiplier	Divider	Unit	Log data
Analog1	0	2	0	1	1000	V	☑
Analog2	2	2	0	1	1000	V	☑
Analog3	4	2	0	1	1000	V	☑
Analog4	6	2	0	1	1000	V	☑
							□

Value = (Raw data +Offset) * Multiplier / Divider

TIP: Dont log data you will not look at, as it uses compute resources

The next 4 analog channels are setup like this:



CAN Device name: A2 (any name you want to call it)

CAN message id: 0x B001 (the CAN address (hex) of the device)

Big/little endian: ● Big endian ○ Little endian

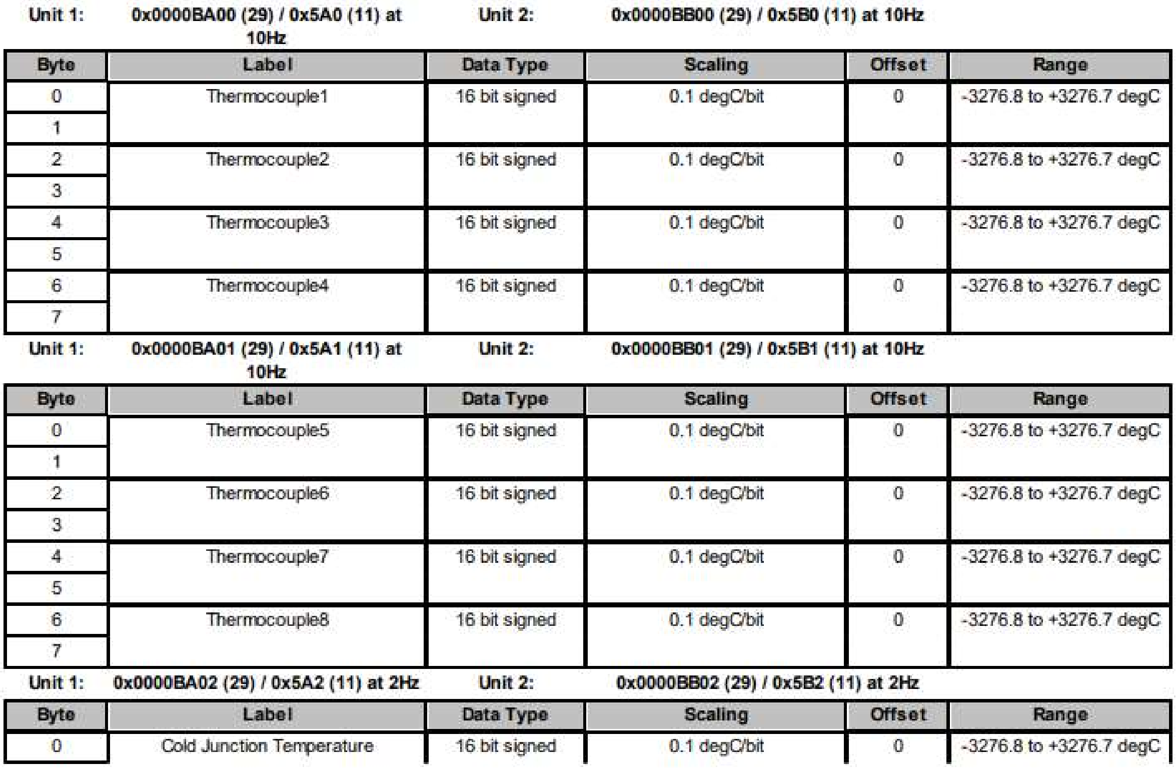
	Channel name	Byte offset	Data length	Offset	Multiplier	Divider	Unit	Log data
▸	Analog5	0	2	0	1	1000	V	☑
	Analog6	2	2	0	1	1000	V	☑
	Analog7	4	2	0	1	1000	V	☑
	Analog8	6	2	0	1	1000	V	☑
*								□

Value = (Raw data +Offset) * Multiplier / Divider

TIP: Don't log data you will not look at, as it uses compute resources

5.2 AEM 8 channel thermocouple module

Spec says:



Unit 1: 0x0000BA00 (29) / 0x5A0 (11) at 10Hz **Unit 2:** 0x0000BB00 (29) / 0x5B0 (11) at 10Hz

Byte	Label	Data Type	Scaling	Offset	Range
0	Thermocouple1	16 bit signed	0.1 degC/bit	0	-3276.8 to +3276.7 degC
1					
2	Thermocouple2	16 bit signed	0.1 degC/bit	0	-3276.8 to +3276.7 degC
3					
4	Thermocouple3	16 bit signed	0.1 degC/bit	0	-3276.8 to +3276.7 degC
5					
6	Thermocouple4	16 bit signed	0.1 degC/bit	0	-3276.8 to +3276.7 degC
7					

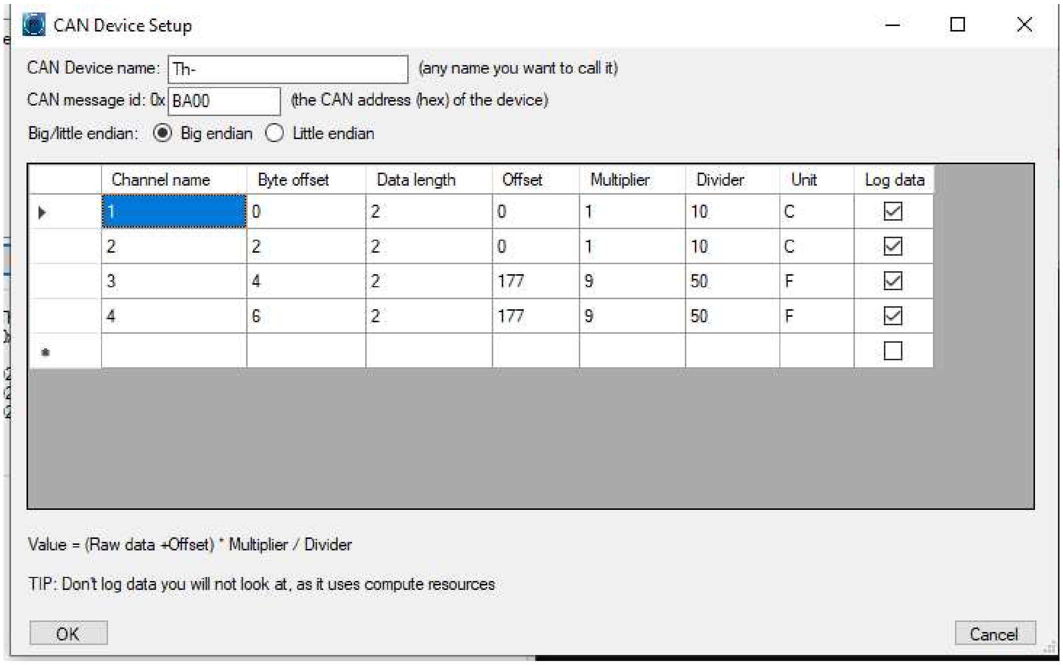
Unit 1: 0x0000BA01 (29) / 0x5A1 (11) at 10Hz **Unit 2:** 0x0000BB01 (29) / 0x5B1 (11) at 10Hz

Byte	Label	Data Type	Scaling	Offset	Range
0	Thermocouple5	16 bit signed	0.1 degC/bit	0	-3276.8 to +3276.7 degC
1					
2	Thermocouple6	16 bit signed	0.1 degC/bit	0	-3276.8 to +3276.7 degC
3					
4	Thermocouple7	16 bit signed	0.1 degC/bit	0	-3276.8 to +3276.7 degC
5					
6	Thermocouple8	16 bit signed	0.1 degC/bit	0	-3276.8 to +3276.7 degC
7					

Unit 1: 0x0000BA02 (29) / 0x5A2 (11) at 2Hz **Unit 2:** 0x0000BB02 (29) / 0x5B2 (11) at 2Hz

Byte	Label	Data Type	Scaling	Offset	Range
0	Cold Junction Temperature	16 bit signed	0.1 degC/bit	0	-3276.8 to +3276.7 degC

Here is how to setup in YourDyno



CAN Device name: Th- (any name you want to call it)

CAN message id: 0x BA00 (the CAN address (hex) of the device)

Big/little endian: ● Big endian ○ Little endian

	Channel name	Byte offset	Data length	Offset	Multiplier	Divider	Unit	Log data
▸	1	0	2	0	1	10	C	☑
	2	2	2	0	1	10	C	☑
	3	4	2	177	9	50	F	☑
	4	6	2	177	9	50	F	☑
*								□

Value = (Raw data +Offset) * Multiplier / Divider

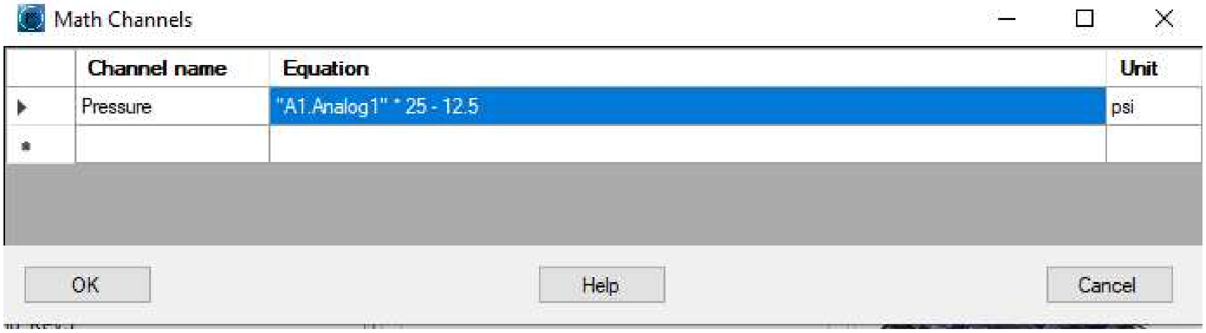
TIP: Dont log data you will not look at, as it uses compute resources

Here two of the channels are setup in degC and two in degF.

6 Converting a CAN channel voltage to something else

With the above procedure you just get the voltage of an analog channel. Say you have connected a pressure sensor with 0.5V = 0psi and 4.5V = 100psi to a 0-5V CAN channel. How is this voltage converted to the pressure?

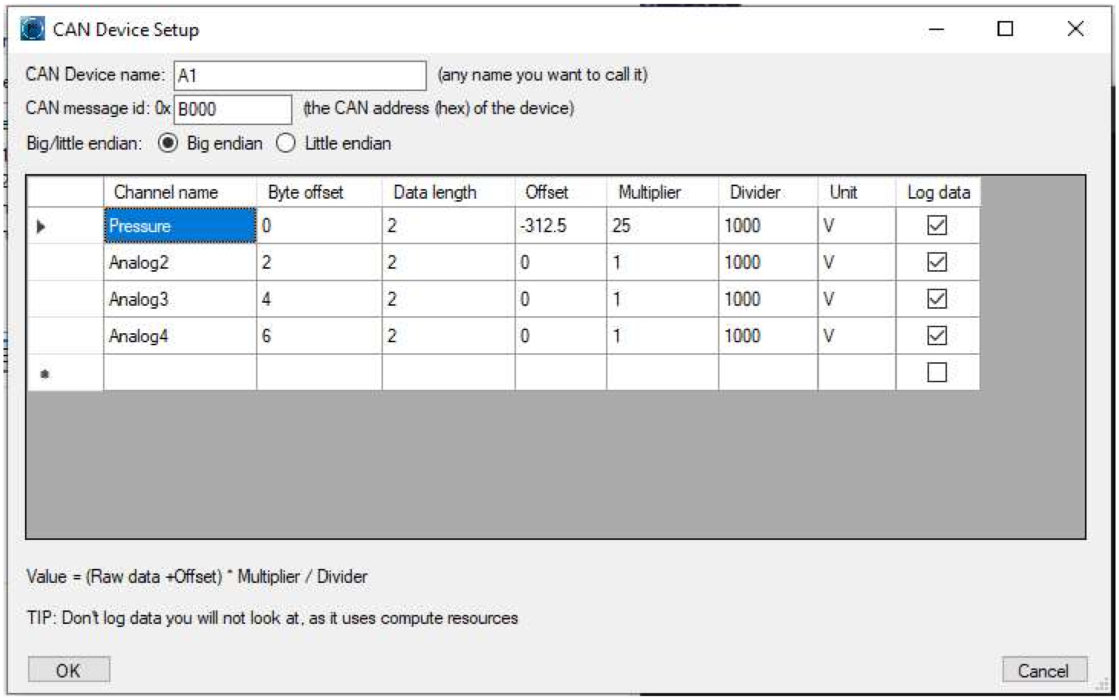
1) Work out the y = a*x + b relationship for the sensor. In this case the solution is Pressure = 25 * voltage – 12.5.
2) Now there are two choices
   a. Use a math channel to convert voltage to pressure:



	Channel name	Equation	Unit
▸	Pressure	"A1.Analog1" * 25 - 12.5	psi
*			

   b. Use the Offset, Multiplier and Divisor in the CAN directly:
      The voltage is the raw CAN data divided by 1000 (check the spec again) so the equation becomes Pressure = 25 * CAN data / 1000 – 12.5

To convert this into CAN Offset, Multiplier and Divisor you use the formula Value = (CAN data + Offset) * Multiplier / Divisor. This gives
Offset = -12.5 * 25 = -312.5
Multiplier = 25
Divisor = 1000



CAN Device name: A1 (any name you want to call it)
CAN message id: 0x B000 (the CAN address (hex) of the device)
Big/little endian: ● Big endian ○ Little endian

	Channel name	Byte offset	Data length	Offset	Multiplier	Divider	Unit	Log data
▶	Pressure	0	2	-312.5	25	1000	V	☑
	Analog2	2	2	0	1	1000	V	☑
	Analog3	4	2	0	1	1000	V	☑
	Analog4	6	2	0	1	1000	V	☑
*								☐

Value = (Raw data +Offset) * Multiplier / Divider

TIP: Don't log data you will not look at, as it uses compute resources

For non-linear channels, use the YourDyno Aux inputs, as these are much easier to configure.