Monitoring fuel consumption on your vehicle in "Real-Time"

PREFACE

Today we are all feeling the constraints of high gas prices. Unless your vehicle comes with an onboard monitor that displays fuel consumption then knowing what your actual fuel consumption is can only be done if you calculate miles driven by gallons burned. To do that you must keep track of your mileage and gallons burned from one fill up until the next. I decided I wanted more so I interface my 2000 Ford Expedition with a laptop to monitor my fuel consumption rate (MPG) in "Real-Time".

To create the real time Interface I needed:

1. OBDII Interface device
2. OBDII J1962 Connector Cable
3. A laptop running Win 98 or better
4. Microsoft Excel 2000 or better with the tool packs installed
5. Data Acquisition Software (DAS)

With the On Board Diagnostics Generation 2 (OBDII) that was made mandatory on all Vehicles sold in the US in 1996 it is possible to create an interface between the vehicles electronic control module (ECM) and a laptop to access this data.

WARNING

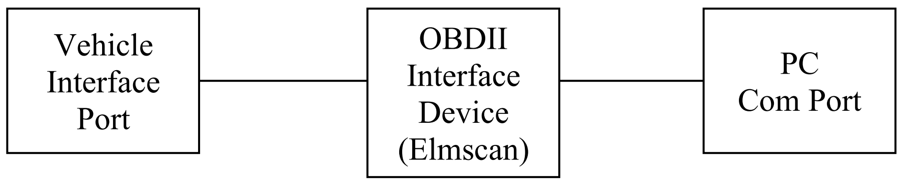
Driving while distracted can be extremely dangerous to yourself and others. It is <u>NOT RECOMMENDED</u> using, adjusting, changing or any other activity with a laptop while driving. Check your local and in-route traffic regulations regarding the use of a laptop/display device while driving.

TABLE OF CONTENTS

OBDII Interface

Since I have a FORD I selected the ELM Electronics Chipset ELM320 which uses a Pulse Width Modulation (PWM) protocol for Ford vehicles and is what my design is built around (There are other device available for other vehicles which I comment on later). With the J1962 connector plugged into the existing interface port located under the steering wheel, a laptop and some special software I can get real-time readings of the pressures, temperatures, vehicle speed, load and many other readings supported. The device communicates with the ECM by simple ASCII commands inputted from the software through the laptops COM port.

Block Diagram



Data Acquisition Software

The OBDII interface only works if you actually type in a specific command requesting data so I must use a terminal program like hypertext to get readings, and they are only updated when I send in a command so this method is not practical. I needed a software program that can automatically send the appropriate commands to the OBDII device; interpret the response and make it available through Dynamic Data Exchange (DDE). Searching around the internet I found Windmill, a simple DAS program that uses the COM ports of any windows based PC; it has logging capabilities and a DDE Server. This program is perfect – it can request specific data through the OBDII interface and serve the data readings onto Excel.

MS Excel

This spreadsheet program is perfect for manipulating the data coming into the PC. The functionality of Excel allows you to exchange data with other devices through DDE and then apply math functions to that data so it is understandable and provides meaningful data to you.

Computing the Fuel Consumption of your Vehicle

This is a fairly straightforward approach. Most vehicles do not have a fuel flow sensor, therefore; you need to use the Mass Air Flow (MAF) sensor and the Vehicle Speed Sensor (VSS) to calculate miles per gallon (MPG).

1. **Mass Air Flow** - The mass of Air in grams per second consumed.
2. **Vehicle Speed Sensor** - The actual speed of the vehicle.

For today's vehicles and by EPA regulations vehicles use the oxygen sensors to feedback data to the ECM and control the air to fuel ratio. This ratio is set at the chemically ideal value of 14.7 grams of air to every gram of gasoline. Since we now have a quantitative value we can use other known values to convert the MAF to Gallons of gas per hour (GPH) and then calculate miles per gallon (MPG).

Conversion Table

454 grams	1 LB
6.701 Lbs Gas	1 Gal
3600 sec	1 Hour
0.621317 Mile	1 Km

Here are the steps to make the conversion:

1. Divide the MAF by 14.7 to get grams of gas per second
2. Divide result by 454 to get Lbs gas per second
3. Divide result by 6.701 Gals gas per second
4. Multiply result by 3600 to get gallons per hour

The math expression for GPH is: MAF * 0.0805

The value for vehicle speed is delivered in Km/Hr, to convert to miles multiply by 0.621317. To calculate MPG divide the MPH by GPH. The final math expression for MPG will be:

$$\frac{VSS * 7.718}{MAF}$$

Configuring the DAS software

Now that we know how to calculate MPG we need to configure the Windmill software to request the data. The data from the ElmScan is returned in HEX format and will need to be converted. MS Excel can do this with the HEX2DEC command and we will address it in that section. First thing installing the software by following the install steps from the developer.

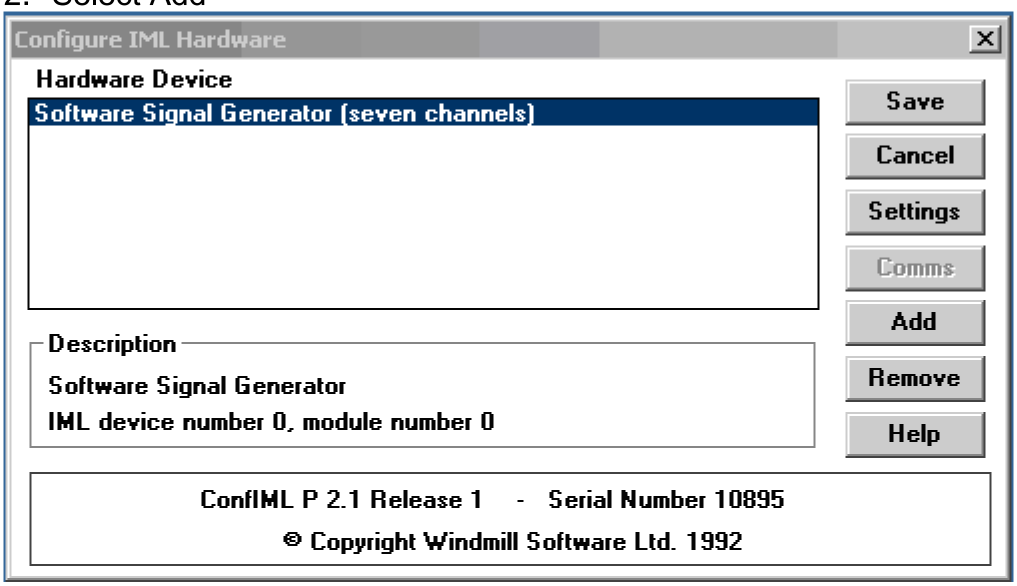
Once installed there are several programs associated with Windmill, the three programs we need to be concerned with are Confiml, Setupiml, and Wmdde that I will describe here.

Confiml: This software performs the work of sending the requests to the ElmScan through the identified COM port, interprets the results and forwards the data to the other Windmill programs.

To configure (see fig below):

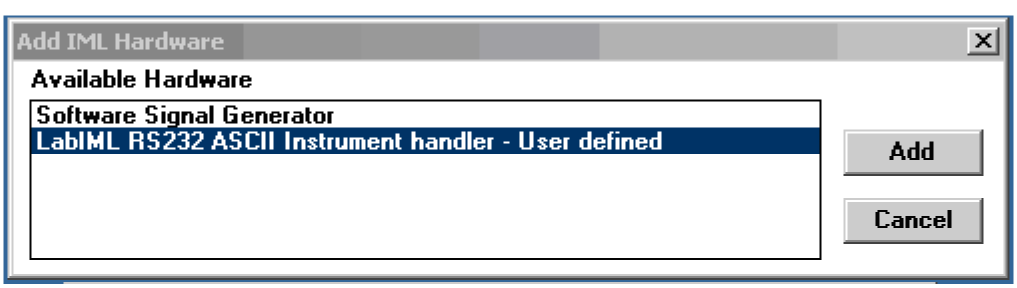
Configure IML Hardware:

1. Open Confiml
2. Select Add



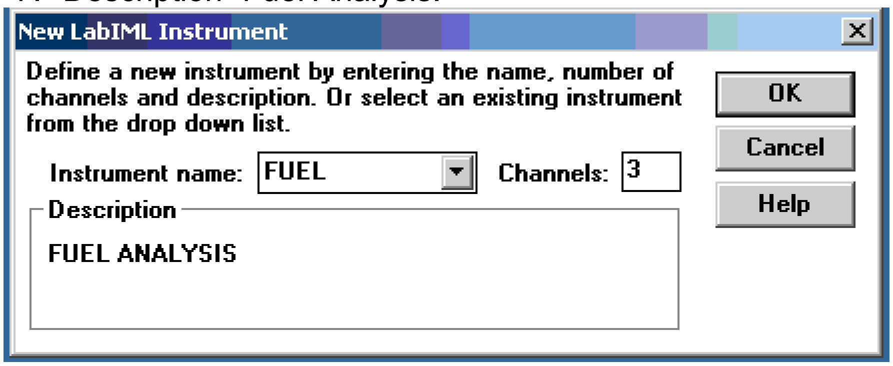
Add IML Hardware

3. Select LabIML RS 232 ASCII Instrument Handler – User Defined
4. Select Add



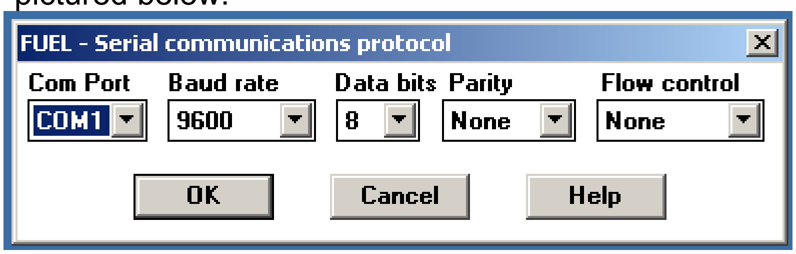
New LabIML Instrument

5. Name – Fuel
6. # of channels - 3 channels
7. Description -Fuel Analysis.

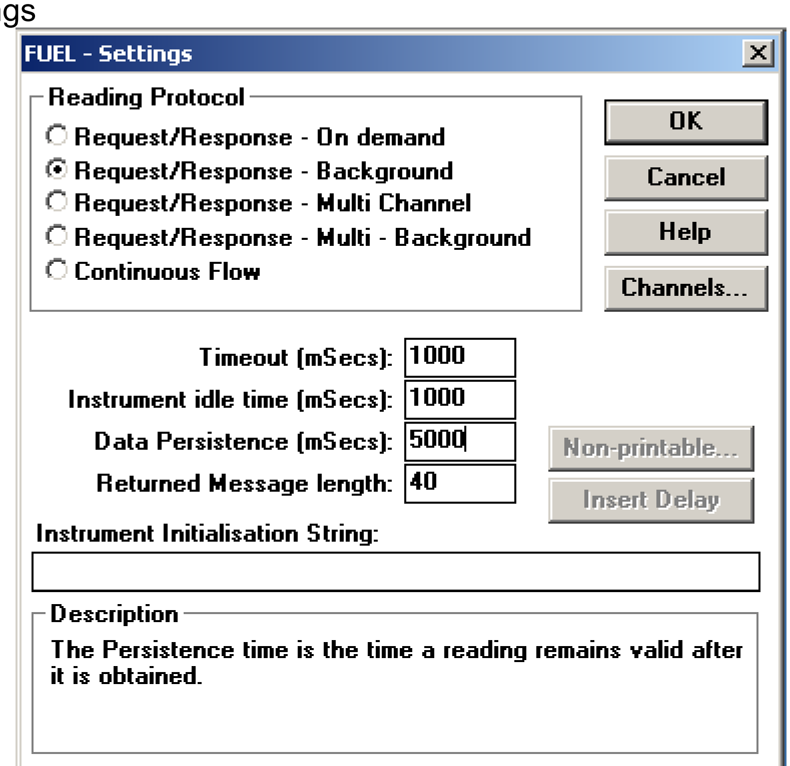


Serial communications protocol

8. Select the proper COM Port (Default 1) and leave the defaults as pictured below:

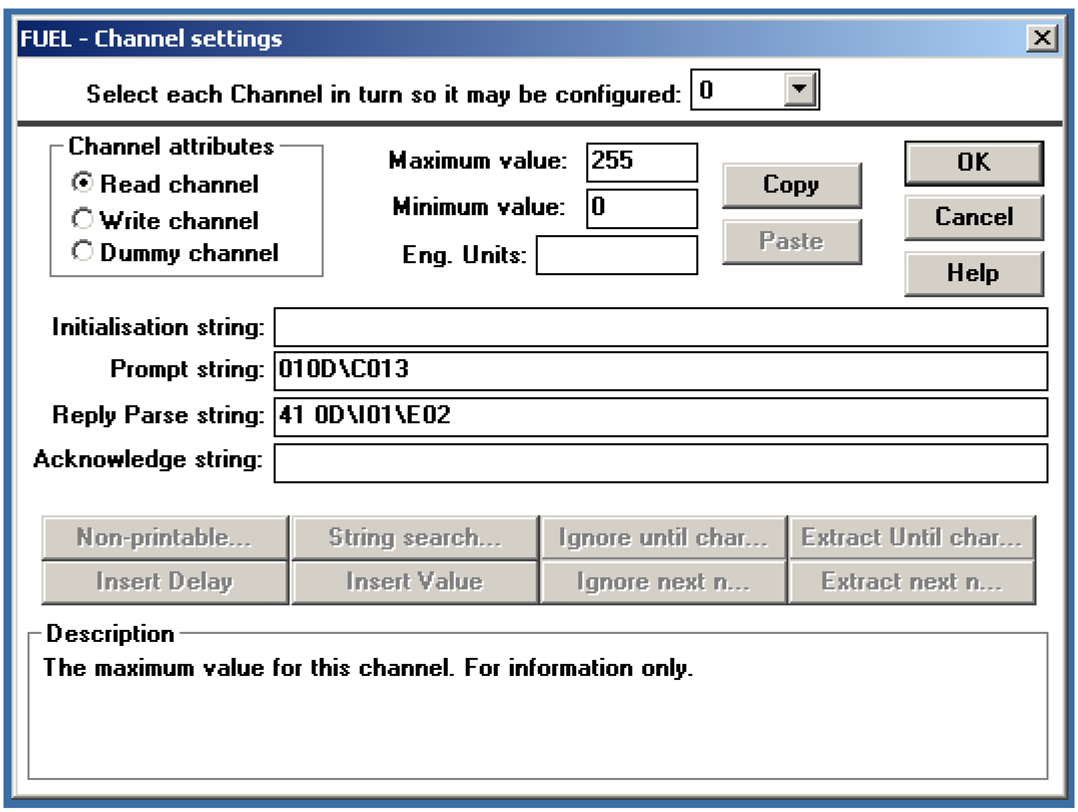


Settings



Channel Settings - Configure the channels using the following commands.

9. Chan 0 (See fig below)
    i. Attributes = Read Channel
    ii. Max Value = 255, Min Value = 0
    iii. Prompt String = 010D\C013
    iv. Reply Parse String 41 0D\I01\E02
10. Chan 1
    i. Attributes = Read Channel
    ii. Max Value = 255, Min Value = 0
    iii. Prompt String = 0110\C01
    iv. Reply Parse String 41 10\I01\E02
11. Chan 2
    i. Attributes = Read Channel
    ii. Max Value = 255, Min Value = 0
    iii. Prompt String = 0110\C013
    iv. Reply Parse String 41 10\I04\E02



12. Select OK
13. Select SAVE on the main page.

Confiml will now close.

I will provide some explanation on the settings above.

1. Setting - perform a request/response – background, this sets the device to request data and respond while working in the background. It is set to timeout if no response is received in 1 second, the idle time is the time it waits to send out another request, the data persistence is how long the data will remain active and returned message length is the maximum number of characters *(you may have to experiment with these settings to get the best data return rate).*

2. Channels – Each channel identified will have a specific task to request a specified piece of data from the ElmScan, in this case we need three pieces of data. The VSS is returned in 1 BYTE in HEX and needs 1 channel. The MAF has 2 BYTES in HEX so we need 2 channels for it. The first box on the very top is the channel number, attributes is set to read channel, the prompt string is what Windmill sends to the device 010D is the request for VSS, 01 is the request for data and 0D is the PID (Identifier in OBDII) for VSS. \C013 is Windmill's command to send character return so this whole string is to simply send a request for the VSS. The reply parse string is what Windmill will do with the response. 41 0D is the response from Elmscan that data was received for PID 0D, \I01 is the Windmill command to ignore the first character after is sees 41 0D and \E02 is the command to extract the next 2 characters.

Setupiml: This is the software that allows you to configure the data channels and build a customized file.

Creating a new data file

1. Open Setupiml
2. Select Create new setup
3. Name it FUEL, Description Fuel Analysis, hit OK

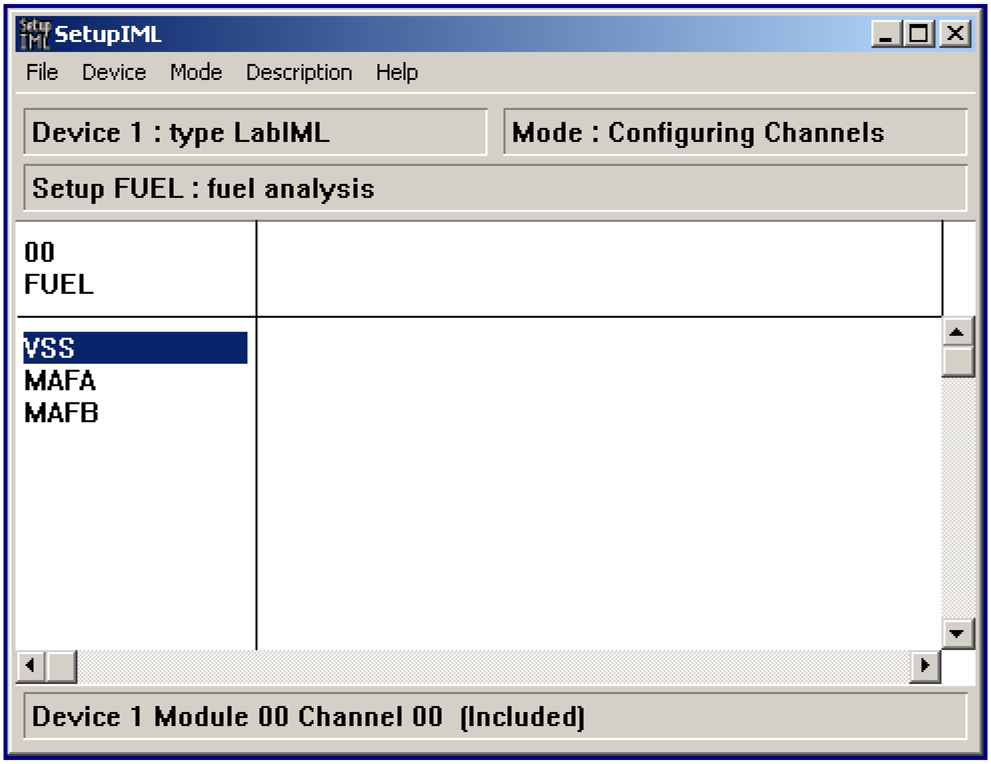
Configuring the file

4. Under Device select "Device 0"
5. Under Mode Select "Select Channels"
6. Double Click all channels on left side to grey out.
7. Under Device Select "Device 1"
8. Under Mode select "Configure Channels"

Name each Channel

9. Double Click the first value on the left column (00000) and name it VSS – select OK
10. Double Click the first value on the left column (00001) and name it MAFA – select OK
11. Double Click the first value on the left column (00002) and name it MAFB – select OK

You will now have something like:

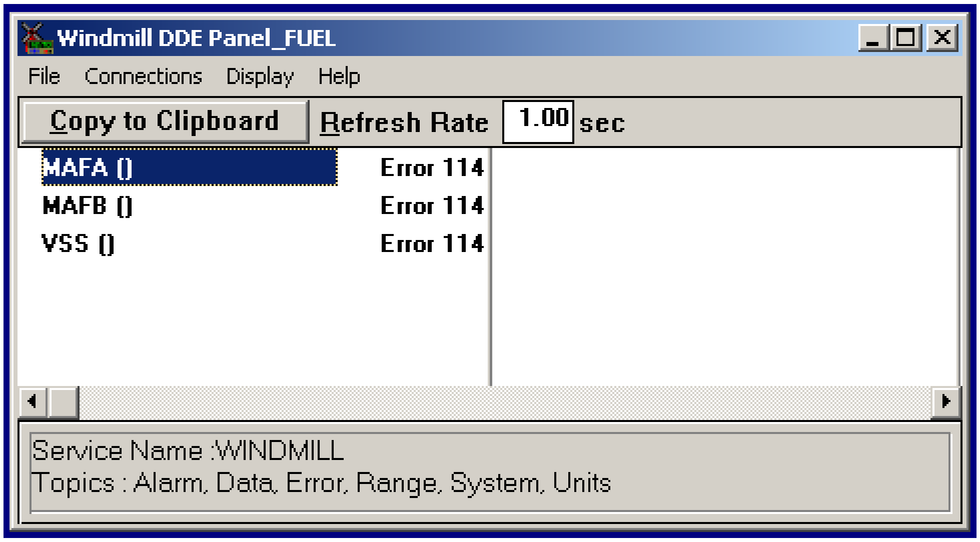


Under file select save and choose the file name you entered (Fuel). You have now completed the setup of the Windmill software.

WmDDE:
The final program in windmill is Wmdde, this program uses the configuration file you have just created to access the data on your vehicle and serve it out with DDE.

1. Run Wmdde
2. File – Load Hardware Setup – select Fuel.IMS (or whatever you named the file in SetupIML).
3. Select Connect All - Ok

If your Elmscan device is connected to your vehicle and PC, your vehicle key is in the on position or running you should now see the data streaming in and may look like 05, A3 etc. Since these are in HEX they must be converted to decimal and then apply a specific math function that is identified under the OBDII PID Table. At this point all we must do next is setup MS Excel to see the data from Wmdde and convert it to MPG.



What you see above is the WmDDE software running with the hardware profile we just setup. The ERROR 114 simply means there is no response, this is because the device is not connected. When working you will see data changing based on the Refresh rate set above (1 Sec is the default but you can set it as low as 0.3 sec).

MS EXCEL Setup:

Most of the work is done now, all we need to do is put together the math functions into Excel Cells. The first thing you must do is ensure you have the Analysis Toolpak activated. You can do this by selecting tools/Add-Ins and the Analysis Toolpak is checked. You may need to install this at this point so make sure you have the install disk available.

Now we must get the data stream from WmDDE into excel, to do this follow these steps.

1. Cell1 enter"=windmill|data!VSS" (without quotes)
2. Cell2 enter"=windmill|data!MAFA" (without quotes)
3. Cell3 enter"=windmill|data!MAFB" (without quotes)

Convert the data from HEX to Decimal

4. Cell4 enter "HEX2DEC(Cell1)" (without quotes)
5. Cell5 enter "HEX2DEC(Cell2)" (without quotes)
6. Cell6 enter "HEX2DEC(Cell3)" (without quotes)

Calculate MAF

7. Cell7 enter "=((256*cell5)+cell6)/100" (without quotes)

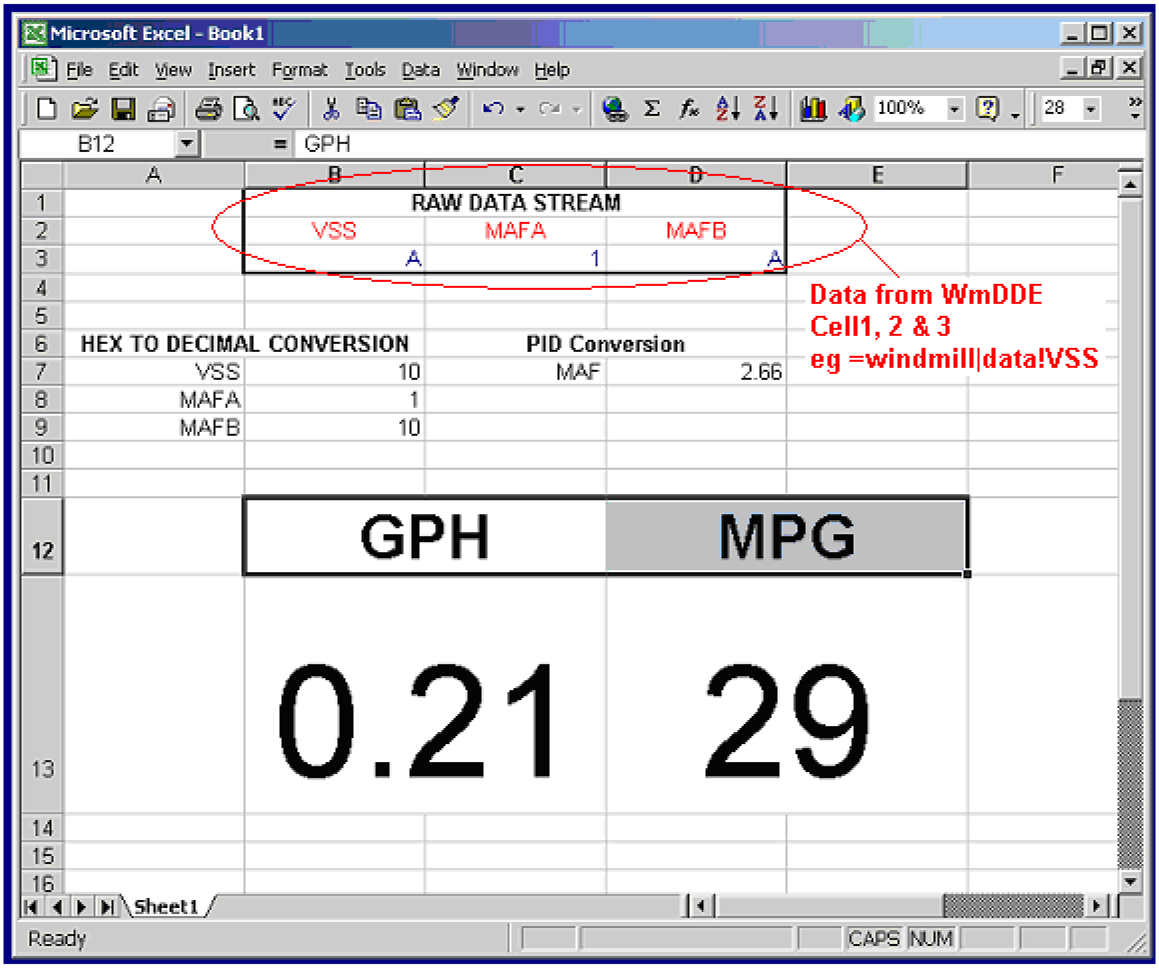
Calculate GPH

8. Cell9 enter "=cell7 * 0.0805" (without quotes)

Calculate MPG

9. Cell10 enter "=(cell1*7.718)/cell4" (without quotes)

This completes all calculations functions, if your system is connected, WmDDE is running and collecting data then you should start seeing data in Excel. Below is how I configured my Excel Sheet.



Appendix A

DAS SOFTWARE

The DAS software Windmill is freely available for download from their website http://www.windmill.co.uk

OBDII to RS232 interpreter

This device is used to read the data from your OBDII (1996 and newer) vehicle through a PC COM Port and ASCII commands. There are different protocols for FORD (PWM), GM (VPW), and other vehicles (ISO, CAN). You will need to determine which your vehicle uses. You can choose to build your own device or purchase one already built.

Description	Protocol	Scantool.net
ElmScan 5 USB Scan Tool	All	421200
ElmScan PWM Scan Tool	PWM	420200
ElmScan VPW Scan Tool	VPW	420300
ElmScan ISO USB	ISO	420500

The following list the parts and schematics for the three common type devices.

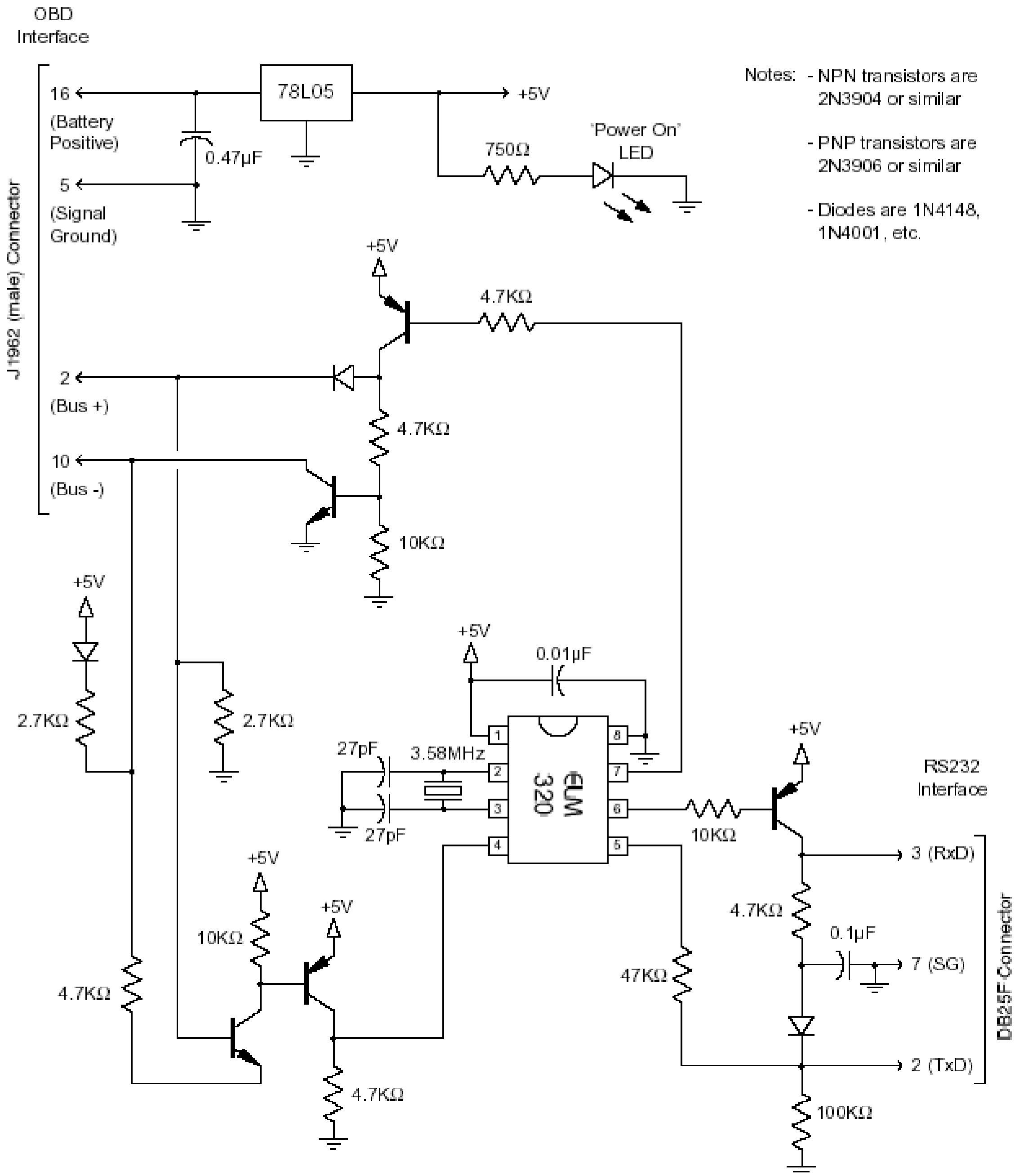
OBDII - ELM PARTS LISTS

ELM320 Data Sheet http://www.elmelectronics.com/DSheets/ELM320DS.pdf

ELM-320 PARTS LIST

Description	QTY	Allied Part #	Scantool.net
ELM 320 Chip	1		360101
DB25M to DB9F Cable	1	0088-1 (1ft)	141001
J1962M to DB9F Cable	1		143301
750 Ohm	1	895-0881	
2.7 kOhm	2	296-6320	
4.7 kOhm	5	832-1393	
10 kOhm	3	832-1118	
47 kOhm	1	296-2182	
100 kOhm	1	832-1110	
0.01 uF	1	507-0326	
0.1 uF	1	881-0478	
0.47 uF	1	613-0497	
27 pF	2	881-5122	
1N4148	3	935-0242	
2N3904 (NPN)	2	431-0406	
2N3906 (PNP)	3	568-0293	
+5v Voltage Regulator	1	568-0965	
ICSOCKET/DIP8	1	900-0004	
3.579545 MHz	1	895-0675	
DB9RA/M	1	720-6170	
DB25RA/F	1	810-0091	
LED5MM/RED	1	679-9981	

ELM320 Schematic

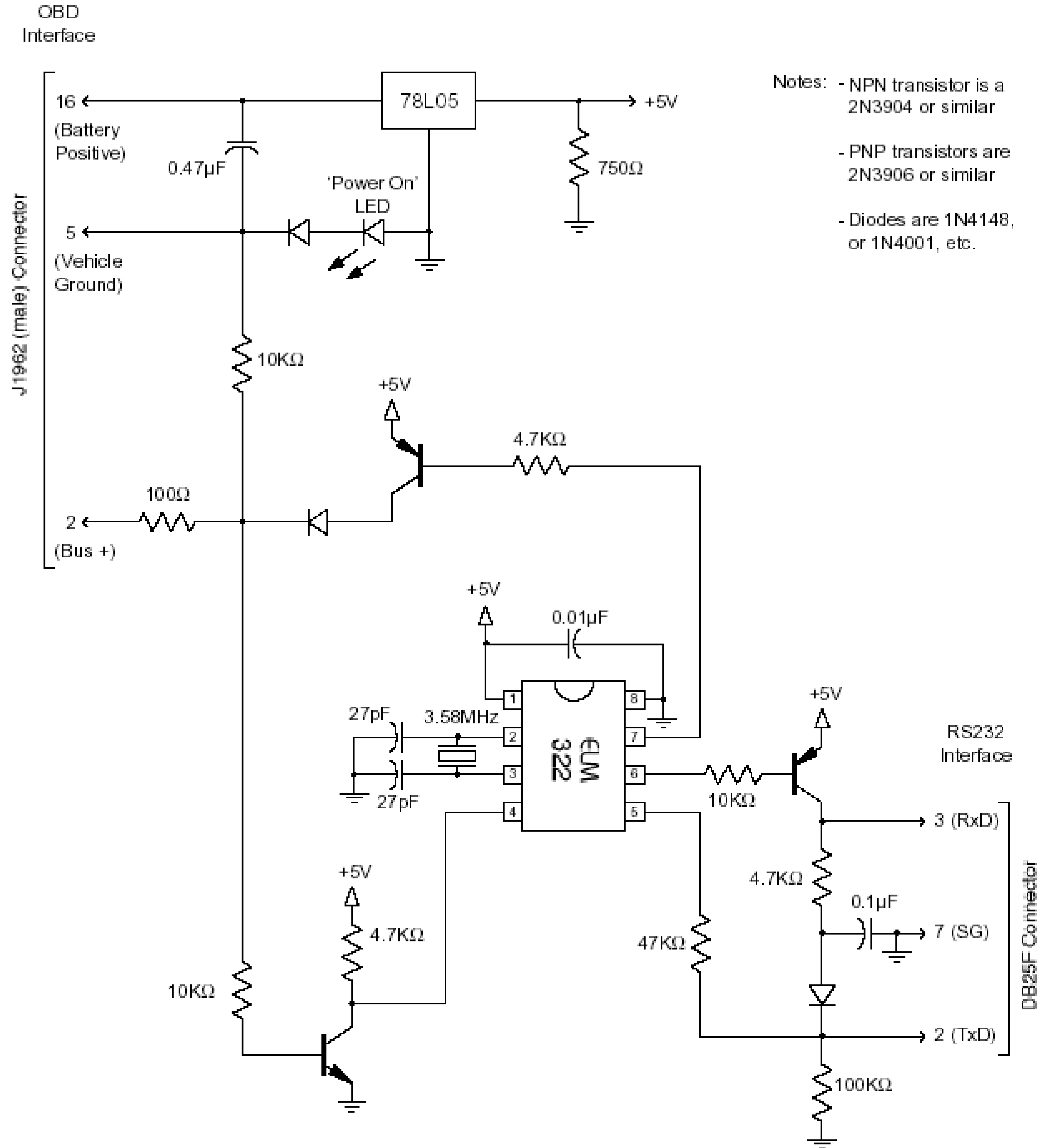


ELM322 Data Sheet http://www.elmelectronics.com/DSheets/ELM322DS.pdf

ELM-322 PARTS LIST

Description	QTY	Allied Part #	Scantool.net
ELM 322 Chip			360201
DB25M to DB9F Cable		0088-1 (1ft)	141001
J1962M to DB9F Cable			143301
750 Ohm	1	895-0881	
100 Ohm, ½ Watt	1	895-0069	
4.7 kOhm	3	832-1393	
10 kOhm	3	832-1118	
47 kOhm	1	296-2182	
100 kOhm	1	832-1110	
0.01 uF	1	507-0326	
0.1 uF	1	881-0478	
0.47 uF	1	613-0497	
27 pF	2	881-5122	
1N4148	3	935-0242	
2N3904 (NPN)	1	431-0406	
2N3906 (PNP)	2	568-0293	
+5v Voltage Regulator	1	568-0965	
ICSOCKET/DIP8	1	900-0004	
3.579545 MHz	1	895-0675	
DB9RA/M	1	720-6170	
DB25RA/F	1	810-0091	
LED5MM/RED	1	679-9981	

ELM322 Schematic



ELM323 Data Sheet http://www.elmelectronics.com/DSheets/ELM323DS.pdf

ELM-323 PARTS LIST

Description	QTY	Allied Part #	Scantool.net
ELM 323 Chip			360301
DB25M to DB9F Cable		0088-1 (1ft)	141001
J1962M to DB9F Cable			143301
750 Ohm	1	895-0881	
2.2 kOhm	2	296-6318	
4.7 kOhm	2	832-1393	
10 kOhm	3	832-1118	
47 kOhm	1	296-2182	
100 kOhm	1	832-1110	
510 Ohm, ½ Watt	2	823-0045	
330 Ohm	2	823-0018	
0.01 uF	2	507-0326	
0.1 uF	2	881-0478	
0.47 uF	1	613-0497	
27 pF	2	881-5122	
1N4148	2	935-0242	
2N3904 (NPN)	2	431-0406	
2N3906 (PNP)	2	568-0293	
+5v Voltage Regulator	1	568-0965	
ICSOCKET/DIP14	1	900-0006	
3.579545 MHz	1	895-0675	
DB9RA/M	1	720-6170	
DB25RA/F	1	810-0091	
LED5MM/RED	1	679-9981	
LED5MM/GREEN	2	679-3001	
LED5MM/YELLOW	2	679-7753	

ELM323 Schematic

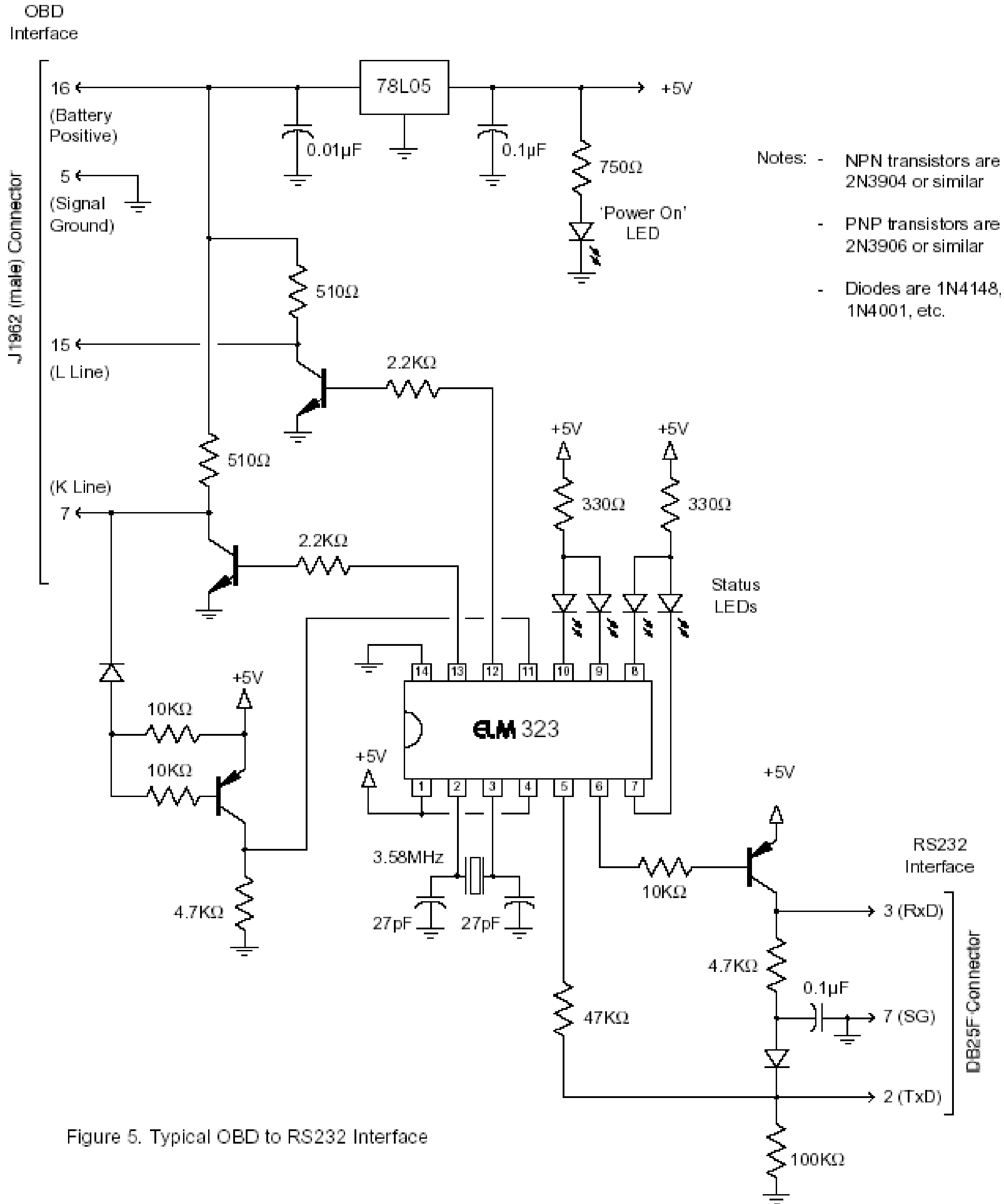


Figure 5. Typical OBD to RS232 Interface

APPENDIX B – OBD- II Testing

TESTING THE ELMSCAN UNIT

In order for you to access the data from your vehicle on your PC Laptop you will need to two vital pieces of information. A terminal program that can send ASCII commands to the ELMScan and the (PID) table that lists all the signal commands needed to request the appropriate data from the vehicle. Windows comes with a terminal program called Hyper-Terminal that you can use to send simple commands to the ELMScan in ASCII format, however; the data returned is in HEX and has several bits of information, you will need to understand this data stream and convert to Decimal.

The Data sheet for the ELMScan Chipset explains in detail what modes you can put the chipset in but for this project we will focus on the “show current data” mode. This is signified by sending “01” out. Now this is meaningless unless you add the appropriate PID code for the specific data you are after, in our case we are looking for the Mass Air Flow (MAF) and Vehicle Speed Sensor (VSS). These PID’s are “10” and “0D” respectively. Therefore, to request the vehicle speed you would send the command 0110 through the terminal program and would be returned:

HEX	DEC
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Send 010D return:

Receive “41 0D 37” This shows a mode 1 response (41) from PID 0D and the value is 37. The value is in HEX, to convert to Decimal multiply the first digit by 16 and add the second digit to reveal a value of 55 kilometers per hour.

Hyper-Terminal is good for testing but in order to continuously receive updated “real-time” values a more powerful program is needed. There are many off the shelf programs out there that will do this for you, many are freeware and are provided when you purchase the ELMScan device or for download form their website. There is one problem with these programs; they do not offer real-time Miles per Gallon monitoring (I have found one program out there but this is a commercial version and costly). Beside, what fun is there if you buy everything already done?

The Data Acquisition Software (DAS) I used to request the signals from the Vehicle Electronic Control Module (ECM) through the ELMScan hardware is named Windmill and is freely available from their website (http://www.windmill.co.uk).

APPENDIX-C OBD-II PIDs

Mode (hex)	PID (hex)	Data bytes returned	Description	Min value	Max value	Units	Formula
01	00	4	PIDs supported				Bit encoded [A7..D0] == [PID 0x01..PID 0x20]
01	01	4	Number of trouble codes and I/M info				Bit encoded. See below.
01	03	2	Fuel system status				Bit encoded. See below.
01	04	1	Calculated engine load value	0	100	%	A*100/255
01	05	1	Engine coolant temperature	-40	215	°C	A-40
01	06	1	Short term fuel % trim —Bank 1	-100 (lean)	99.22 (rich)	%	0.7812 * (A-128)
01	07	1	Long term fuel % trim —Bank 1	-100 (lean)	99.22 (rich)	%	0.7812 * (A-128)
01	08	1	Short term fuel % trim —Bank 2	-100 (lean)	99.22 (rich)	%	0.7812 * (A-128)
01	09	1	Long term fuel % trim —Bank 2	-100 (lean)	99.22 (rich)	%	0.7812 * (A-128)
01	0A	1	Fuel pressure	0	765	kPa (gauge)	A*3
01	0B	1	Intake manifold	0	255	kPa	A

			pressure			(absolute)	
01	0C	2	Engine RPM	0	16,383.75	rpm	((A*256)+B)/4
01	**0D**	**1**	**Vehicle speed**	**0**	**255**	**km/h**	**A**
01	0E	1	Timing advance	-64	63.5	° relative to #1 cylinder	A/2 - 64
01	0F	1	Intake air temperature	-40	215	°C	A-40
01	**10**	**2**	**MAF air flow rate**	**0**	**655.35**	**g/s**	**((256*A)+B) / 100**
01	11	1	Throttle position	0	100	%	A*100/255
01	12	1	Sec.(?) air status				Bit encoded. See below.
01	13	1	Oxygen sensors present				[A0..A3] == Bank 1, Sensors 1-4. [A4..A7] == Bank 2...
01	14	2	Bank 1, Sensor 1: Oxygen sensor voltage, Short term fuel trim	0 0	1.275 99.2	Volts %	A * 0.005 (B-128) * 0.7812 (if B==0xFF, sensor is not used in trim calc)
01	15	2	Bank 1, Sensor 2: Oxygen sensor voltage, Short term fuel trim	0 0	1.275 99.2	Volts %	A * 0.005 (B-128) * 0.7812 (if B==0xFF, sensor is not used in trim calc)

01	16	2	Bank 1, Sensor 3: Oxygen sensor voltage, Short term fuel trim	0 0	1.275 99.2	Volts %	A * 0.005 (B-128) * 0.7812 (if B==0xFF, sensor is not used in trim calc)
01	17	2	Bank 1, Sensor 4: Oxygen sensor voltage, Short term fuel trim	0 0	1.275 99.2	Volts %	A * 0.005 (B-128) * 0.7812 (if B==0xFF, sensor is not used in trim calc)
01	18	2	Bank 2, Sensor 1: Oxygen sensor voltage, Short term fuel trim	0 0	1.275 99.2	Volts %	A * 0.005 (B-128) * 0.7812 (if B==0xFF, sensor is not used in trim calc)
01	19	2	Bank 2, Sensor 2: Oxygen sensor voltage, Short term fuel trim	0 0	1.275 99.2	Volts %	A * 0.005 (B-128) * 0.7812 (if B==0xFF, sensor is not used in trim calc)
01	1A	2	Bank 2, Sensor 3: Oxygen sensor voltage, Short term fuel trim	0 0	1.275 99.2	Volts %	A * 0.005 (B-128) * 0.7812 (if B==0xFF, sensor is not used in trim calc)
01	1B	2	Bank 2, Sensor 4: Oxygen sensor voltage, Short term fuel trim	0 0	1.275 99.2	Volts %	A * 0.005 (B-128) * 0.7812 (if B==0xFF, sensor is not used in trim calc)
01	1C	1	OBD standards this vehicle conforms to				Bit encoded. See below.
01	1D	1	Oxygen sensors present				Similar to PID 13, but [A0..A7] == [B1S1, B1S2, B2S1, B2S2, B3S1, B3S2, B4S1, B4S2]
01	1E	1	Auxiliary input status				A0 == Power Take Off (PTO) status (1 ==

							active) [A1..A7] not used
01	1F	2	Run time since engine start	0	65,535	seconds	(A*256)+B
01	20	4	PIDs supported 21-40				Bit encoded [A7..D0] == [PID 0x21..PID 0x40]
01	21	2	Distance traveled with malfunction indicator lamp (MIL) on	0	65,535	km	(A*256)+B
01	22	2	Fuel Rail Pressure (relative to manifold vacuum)	0	5177.265	kPa	((A*256)+B) * 0.079
01	23	2	Fuel Rail Pressure (diesel)	0	655350	kPa (gauge)	((A*256)+B) * 10
01	24	4	O2S1_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*0.0000305 ((C*256)+D)*0.000122
01	25	4	O2S2_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*0.0000305 ((C*256)+D)*0.000122
01	26	4	O2S3_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*0.0000305 ((C*256)+D)*0.000122
01	27	4	O2S4_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*0.0000305 ((C*256)+D)*0.000122
01	28	4	O2S5_WR_lambda(1): Equivalence Ratio	0 0	2 8	N/A V	((A*256)+B)*0.0000305 ((C*256)+D)*0.000122

			Voltage				
01	29	4	O2S6_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*0.0000305 ((C*256)+D)*0.000122
01	2A	4	O2S7_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*0.0000305 ((C*256)+D)*0.000122
01	2B	4	O2S8_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*0.0000305 ((C*256)+D)*0.000122
01	2C	1	Commanded EGR	0	100	%	100*A/255
01	2D	1	EGR Error	-100	99.22	%	A*0.78125 - 100
01	2E	1	Commanded evaporative purge	0	100	%	100*A/255
01	2F	1	Fuel Level Input	0	100	%	100*A/255
01	30	1	# of warm-ups since codes cleared	0	255	N/A	A
01	31	2	Distance traveled since codes cleared	0	65,535	km	(A*256)+B
01	32	2	Evap. System Vapor Pressure	-8,192	8,192	Pa	((A*256)+B)/4 - 8,192
01	33	1	Barometric pressure	0	255	kPa (Absolute)	A

01	34	4	O2S1_WR_lambda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)*0.0000305 ((C*256)+D)*0.00391 - 128
01	35	4	O2S2_WR_lambda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)*0.0000305 ((C*256)+D)*0.00391 - 128
01	36	4	O2S3_WR_lambda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)*0.0000305 ((C*256)+D)*0.00391 - 128
01	37	4	O2S4_WR_lambda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)*0.0000305 ((C*256)+D)*0.00391 - 128
01	38	4	O2S5_WR_lambda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)*0.0000305 ((C*256)+D)*0.00391 - 128
01	39	4	O2S6_WR_lambda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)*0.0000305 ((C*256)+D)*0.00391 - 128
01	3A	4	O2S7_WR_lambda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)*0.0000305 ((C*256)+D)*0.00391 - 128
01	3B	4	O2S8_WR_lambda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)*0.0000305 ((C*256)+D)*0.00391 - 128
01	3C	2	Catalyst Temperature Bank 1, Sensor 1	-40	6,513.5	°C	((A*256)+B)/10 -40
01	3D	2	Catalyst Temperature Bank 1, Sensor 2	-40	6,513.5	°C	((A*256)+B)/10 -40

01	3E	2	Catalyst Temperature Bank 2, Sensor 1	-40	6,513.5	°C	((A*256)+B)/10 -40
01	3F	2	Catalyst Temperature Bank 2, Sensor 2	-40	6,513.5	°C	((A*256)+B)/10 -40
01	40	4	PIDs supported 41-60 (?)				Bit encoded [A7..D0] == [PID 0x41..PID 0x60] (?)
01	41	?	Monitor status this drive cycle	?	?	?	?
01	42	2	Control module voltage	0	65.535	V	((A*256)+B)/1000
01	43	2	Absolute load value	0	25696	%	((A*256)+B)*100/255
01	44	2	Command equivalence ratio	0	2	N/A	((A*256)+B)*0.0000305
01	45	1	Relative throttle position	0	100	%	A*100/255
01	46	1	Ambient air temperature	-40	215	°C	A-40
01	47	1	Absolute throttle position B	0	100	%	A*100/255
01	48	1	Absolute throttle position C	0	100	%	A*100/255
01	49	1	Accelerator pedal position D	0	100	%	A*100/255
01	4A	1	Accelerator pedal	0	100	%	A*100/255

			position E				
01	4B	1	Accelerator pedal position F	0	100	%	A*100/255
01	4C	1	Commanded throttle actuator	0	100	%	A*100/255
01	4D	2	Time run with MIL on	0	65,535	minutes	(A*256)+B
01	4E	2	Time since trouble codes cleared	0	65,535	minutes	(A*256)+B
01	C3	?	?	?	?	?	Returns numerous data, including Drive Condition ID and Engine Speed*
01	C4	?	?	?	?	?	B5 is Engine Idle Request B6 is Engine Stop Request*
02	02	2	Freeze frame trouble code				BCD encoded, see below.
03	N/A	n*6	Request trouble codes				3 codes per message frame, BCD encoded. See below.
04	N/A	0	Clear trouble codes / Malfunction indicator lamp (MIL) / Check engine light				Clears all stored trouble codes and turns the MIL off.
09	02	5x5	Vehicle identification number (VIN) number				Returns 5 lines, A is line ordering flag, B-E ASCII

							coded VIN digits.