ELM325
J1708 Interpreter

Description

With the advent of electronic engine controls, many vehicles also adopted some form of diagnostic tools to help monitor their operation. As more modules began to be used in vehicles, there was also a need for the devices to share information rather than each independently obtain this from separate sensors.

In the 1980's, the SAE J1708 standard was created to provide a specification for a common data bus to be used in heavy duty vehicles. It used RS485 wiring (already proven to be reliable in noisy environments), and a UART-based low speed data format. The SAE J1587 standard followed a few years later to describe the mechanism by which messages and data should be sent between vehicle modules.

The ELM325 allows a PC or similar device to be used to monitor and query devices on a J1708 data bus, using simple commands that can be sent from almost any terminal program. It is able to work with either the J1587 or the J1922 data formats.

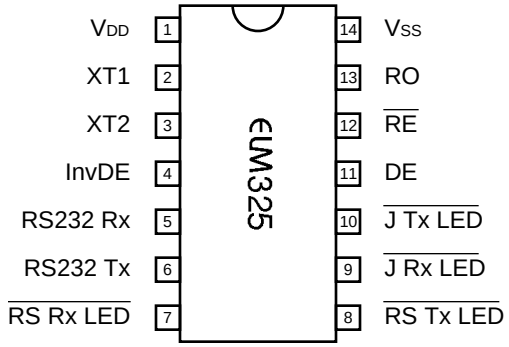
Applications

- Diagnostic trouble code readers
- Heavy duty vehicle scan tools
- Teaching aids
- ECU Simulators

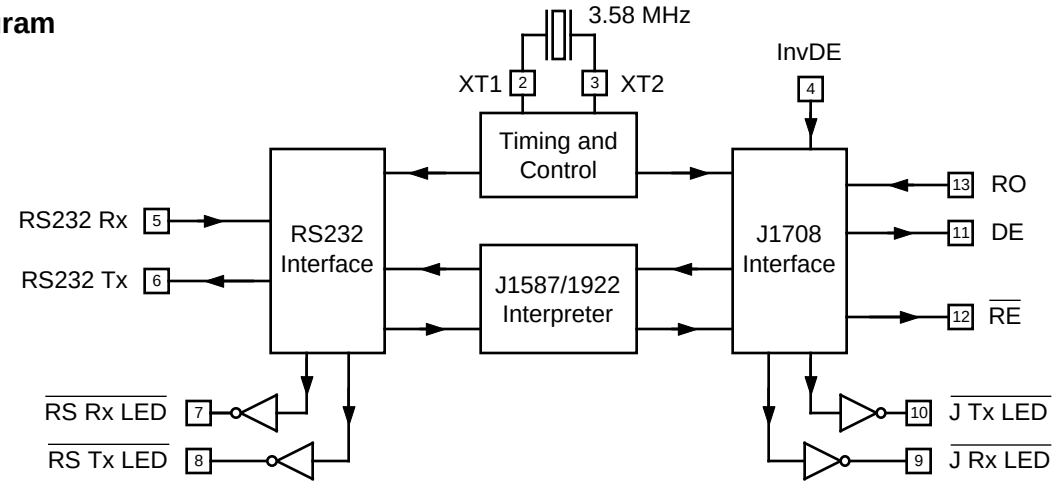
Features

- Supports both SAE J1587 and J1922
- High speed RS232 interface
- Works with standard RS485 transceivers
- Fully configurable with AT commands
- Wide operating voltage range (1.8 to 5.5V)
- Low power CMOS design

Connection Diagram
PDIP and SOIC
(top view)



Block Diagram



Contents

Pin Descriptions

V_{DD} (pin 1)

This pin is the positive supply pin, and should always be the most positive point in the circuit. Internal circuitry connected to this pin is used to provide power on reset of the microprocessor, so an external reset signal is not required. Refer to the Electrical Characteristics section for further information.

XT1 (pin 2) and **XT2** (pin 3)

A 3.579545 MHz oscillator crystal (often known simply as a '3.58 MHz' crystal) is to be connected between these two pins. Loading capacitors as required by the crystal (typically 27pF) also need to be connected from each of these pins to circuit common (Vss). A crystal is preferred, but you may also use a ceramic resonator.

Note that this device has not been configured for operation with an external oscillator, and it expects a crystal or resonator to be connected between these pins. Use of an external clock source is not recommended.

InvDE (pin 4)

This input is used to invert the output at the DE pin. This allows the ELM325 to be connected to a variety of RS485 interface circuits without the need for an external inverter.

Normally, the InvDE pin is connected to a high (V_{DD}) level. This causes the DE output (pin 11) to be at a low level during idle, and go to a high level for the start bit (ie. when active). This is required for most RS485 interface ICs such as the 75176, DS485, LTC485, ADM485, MAX485, and SN65HVD3080E. If you are using a device that requires an active low DE output (such as the DS36277), then connect the InvDE pin to a low level.

RS232 Rx (pin 5)

This is the RS232 (serial) data receive input. The signal level is compatible with most interface ICs (when at idle, the level should be high), but can be used with other interfaces as well, since the input has Schmitt trigger input wave shaping.

RS232 Tx (pin 6)

This is the RS232 (serial) data transmit output. The signal level is compatible with most interface ICs (the output is high when idle), and there is sufficient current drive to allow interfacing using only a PNP transistor, if desired.

$\overline{\text{RS Rx LED}}$ (pin 7), **$\overline{\text{RS Tx LED}}$** (pin 8),
$\overline{\text{J Rx LED}}$ (pin 9) and **$\overline{\text{J Tx LED}}$** (pin 10)

These four pins normally output a high level, and are driven low when the ELM325 is transmitting or receiving data. The internal circuitry is suitable for directly driving most LEDs through current limiting resistors, or interfacing to other logic circuits. When using lower V_{DD} levels, the LED should be chosen so that the forward voltage drop is not more than the supply level. If unused, these pins should be left open-circuited.

DE (pin 11)

This is the J1708 transmit data output, which should be connected to the DE pin on an RS485 transceiver integrated circuit. The polarity of this signal may be changed by changing the level on the InvDE pin.

$\overline{\text{RE}}$ (pin 12)

This is a receiver enable output, which may be used to enable or disable the RS485 receiver. For some circuits this might result in a substantial reduction in current, if the receiver is not being used.

This pin is set to a low level on powerup, or ELM325 reset. The level may be controlled with the AT RE0 and RE1 commands. If unused, this pin should be left open-circuited.

RO (pin 13)

This is the J1708 receive data input, and should be connected to the receive output (RO) pin of the RS485 transceiver IC. This is a standard CMOS input that accepts TTL level signals.

Vss (pin 14)

Circuit common must be connected to this pin.

Absolute Maximum Ratings

Storage Temperature..... -65°C to +150°C

Ambient Temperature with Power Applied.....-40°C to +85°C

Voltage on V_{DD} with respect to V_{SS}..... -0.3V to +6.5V

Voltage on any other pin with respect to V_{SS}..... -0.3V to (V_{DD} + 0.3V)

Note:

These values are given as a design guideline only. The ability to operate to these levels is neither inferred nor recommended, and stresses beyond those listed here will likely damage the device.

Electrical Characteristics

All values are for operation at 25°C and a 5V supply, unless otherwise noted. For further information, refer to note 1 below.

Characteristic		Minimum	Typical	Maximum	Units	Conditions
Supply voltage, V_{DD}		1.8	5.0	5.5	V	
V_{DD} rate of rise		0.05			V/ms	see note 2
Average current, I_{DD}			1.2	2.0	mA	see note 3
Output low current	V_{DD} = 5.0V		8.0		mA	V_{OUT} = 0.6V max
	V_{DD} = 3.3V		6.0		mA	V_{OUT} = 0.6V max
Output high current	V_{DD} = 5.0V		3.5		mA	V_{OUT} = 4.3V min
	V_{DD} = 3.3V		3.0		mA	V_{OUT} = 2.6V min
J1708 Baud Rate			9600		bps	
RS232 Baud Rate			57600		bps	
Reset time	AT Z		900		msec	measured from the end of the command to the start of the ID message (ELM325 v1.0)
	AT WS		0.75		msec	

Notes:

1. This integrated circuit is based on Microchip Technology Inc.'s PIC16F1823 device. For more detailed device specifications, and possibly clarification of those given, please refer to the Microchip documentation (available at http://www.microchip.com/).
2. This spec must be met in order to ensure that a correct power on reset occurs. It is quite easily achieved using most common types of supplies, but may be violated if one uses a slowly varying supply voltage, as may be obtained through direct connection to solar cells or some charge pump circuits.
3. ELM325 device only – does not include any load currents

Overview

The following describes how to use the ELM325 to obtain information from your vehicle.

We begin by discussing just how to 'talk' to the IC using a PC, then explain how to change options using the 'AT' commands, and finally we show how to communicate with a vehicle. For the more advanced experimenters, there are also sections on how to use some of the other features of this product as well.

Using the ELM325 is not as daunting as it first seems. Many users may never need to issue an 'AT' command, adjust timeouts, or change the MID. For those that do want to make changes, all that is required is a PC or smart device with a terminal program (such as HyperTerminal or ZTerm), and a little knowledge…

Communicating with the ELM325

The ELM325 expects to communicate with the controlling device through an RS232 serial connection. Although most modern devices do not usually provide a serial connection such as this, there are several ways in which a 'virtual serial port' can be created. The most common devices are USB to RS232 adapters, but there are several others such as Wi-Fi modules, ethernet devices, or Bluetooth to serial adapters.

No matter how you physically connect to the ELM325, you will need a way to send and receive data. The simplest method is to use one of the many 'terminal' programs that are available (HyperTerminal, ZTerm, etc.), to allow typing the characters directly from your keyboard.

To use a terminal program, you will need to adjust several settings. First, ensure that your software is set to use the proper 'COM' port, and that you have chosen the proper data rate – the ELM325 can only communicate at 57600 bps. If you select the wrong 'COM' port, you will not be able to send or receive any data. If you select the wrong data rate, but the right 'COM' port, the information that you send and receive will be unreadable by you or the ELM325. Don't forget to also set your connection for 8 data bits, no parity bits, and 1 stop bit, and to set it for the proper 'line end' mode. All of the responses from the ELM325 are terminated with a single carriage return character and, optionally, a linefeed character (depending on your settings).

Properly connected and powered, the ELM325 will energize the four LED outputs in sequence (as a 'lamp test') and will then send the message:

```
ELM325 v1.0

>
```

In addition to identifying the version of this IC, receiving this string is a good way to confirm that the computer connections and terminal software settings are correct (however, at this point no communications have taken place with the vehicle, so the state of that connection is still unknown).

The '>' character that is shown on the second line is the ELM325's prompt character. It indicates that the device is in the idle state, ready to receive characters on the RS232 port. If you did not see the identification string, try resetting the IC again with the AT Z (reset) command. Simply type the letters A T and Z (spaces are optional), then press the return key:

```
>AT Z
```

That should cause the LEDs to flash again, and the identification string to be printed. If you only see strange looking characters, then check your baud rate – you have likely set it incorrectly.

Characters sent from the computer can either be intended for the ELM325's internal use, or for reformatting and passing on to the vehicle. The ELM325 can quickly determine where the received characters are to be directed by monitoring the contents of the message. Commands that are intended for the ELM325's internal use will begin with the characters 'AT', while messages for the vehicle are only allowed to contain the ASCII codes for hexadecimal digits (0 to 9 and A to F).

Whether it is an 'AT' type internal command or a hex string for the J1708 bus, all messages to the ELM325 must be terminated with a carriage return character (hex '0D') before it will be acted upon. The one exception is when an incomplete string is sent and no carriage return appears. In this case, an internal timer will automatically abort the incomplete message after about 20 seconds, and the ELM325 will print a single question mark ('?') to show that the input was not understood (and was not acted upon).

Messages that are not understood by the ELM325 (syntax errors) will always be signalled by a single

Communicating with the ELM325 (continued)

question mark. These include incomplete messages, incorrect AT commands, or invalid hexadecimal digit strings, but are not an indication of whether or not the message was understood by the vehicle. One must keep in mind that the ELM325 is a protocol interpreter that makes no attempt to assess the content of messages for validity – it only ensures that hexadecimal digits were received, combined into bytes, then sent out the J1708 port, and it does not know if a message sent to the vehicle was appropriate or not.

While processing J1708 messages, the ELM325 will continually monitor for an RS232 character received. If it sees one, this will interrupt the IC, quickly returning control to the user. Since the time to respond varies with what the ELM325 was doing at the time, software should always wait for the prompt character ('>' or hex 3E) to be received, before beginning to send the next command.

Finally, it should be noted that the ELM325 is not case-sensitive, so the commands 'ATZ', 'atz', and 'AtZ' are all exactly the same to the ELM325. All commands may be entered as you prefer, as no one method is faster or better. The ELM325 also ignores space characters and all control characters (tab, etc.), so they can be inserted anywhere in the input to improve readability.

One other feature of the ELM325 is the ability to repeat the last command when only a single carriage return character is received. This may be convenient if you have sent a command and wish to repeat it in order to obtain updates. If you simply send a carriage return, then you do not have to resend the entire command. The memory buffer only remembers one previous command though – there is no provision in the current ELM325 to provide storage for more.

AT Commands

Several parameters within the ELM325 can be adjusted in order to modify its behaviour. These do not normally have to be changed before attempting to talk to the vehicle, but occasionally the user may wish to customize these settings – for example by turning the character echo off, or adjusting a timeout value. In order to do this, internal 'AT' commands must be used.

Those familiar with PC modems will immediately recognize AT commands as a standard way in which modems are internally configured. The ELM325 uses essentially the same method, always watching the data sent by the PC, looking for messages that begin with the character 'A' followed by the character 'T'. If found, the next characters will be interpreted as an internal configuration or 'AT' command, and will be executed upon receipt of a terminating carriage return character. If the command is just a setting change, the ELM325 will reply with the characters 'OK', to say that it was successfully completed.

Some of the commands require that numbers be provided as arguments, in order to set the internal values. These will always be hexadecimal digits. Also, one should be aware that for the on/off types of commands, the second character is the number 1 or the number 0, the universal terms for on and off.

The following is a summary of all of the current ELM325 commands. A more complete description of each command follows, starting on the next page.

AT Command Summary

General

<CR>	repeat the last command
D	set all to Defaults
E0, E1	Echo off, or on*
I	print the version ID
L0, L1	Linefeeds off, or on*
WS	Warm Start (quick software reset)
Z	reset all

Other

EM0, EM1	Error Messages off, or on*
R0, R1	Responses off, or on*
RE0, RE1	RE output pin low*, or high
S0, S1	Spaces off, or on*
ST hh	Set Timeout to hh

J1708 Specific

AF0, AF1	Auto Formatting off, or on*
C0, C1	Checksum display off*, or on
F1	reset Filter 1
F2	reset Filter 2
F1 hh hh hh hh hh	set Filter 1
F2 hh hh hh hh hh	set Filter 2
GM	Get Message
GO	Get One message
MA	Monitor All messages
MM hh	Monitor for MID hh messages
MP hh	Monitor for PID hh messages
MP 01hh	Monitor for page 2 PID hh
SM hh	Set MID to hh
SP h	Set Priority to h
TC	get Trouble Code messages

Note: Settings shown with an asterisk (*) are the default values

AT Command Descriptions

The following describes each AT Command that the current version of the ELM325 supports, in some detail. Many of these commands are also described further in other sections:

<CR> [repeat the last command]

Sending a single carriage return character causes the ELM325 to repeat the last command that it performed. This helps if you wish to repeat a request several times to obtain updates for a monitored value, for example.

AF0 and **AF1** [Auto Formatting off or on*]

All messages must start with the sender's address (the MID) and end with a checksum. If automatic formatting is on (it is by default), the ELM325 will add these to every message for you, so that you only need to provide the data bytes that you wish to send. If automatic formatting is off, you must provide all bytes, including the MID and the checksum.

Automatic (ie 'auto') formatting also provides help when making requests for a PID. If you provide one data byte, or two data bytes with the first being 01, the ELM325 assumes that you wish to make a PID request. It will add the MID, the request PID, checksum, etc. for you, set the filters if you haven't, and send the request. If the auto formatting is off, all bytes that you provide are sent without modification of any kind (but the ELM325 still attempts to set filters for you, if you have not set any).

One final issue to note is that the ELM325's internal J1708 buffer is 21 bytes long. This means that you may actually send a 23 byte message if formatting is on (MID + 21 bytes in buffer + checksum), but if the auto formatting is off, the total length of the message must be 21 bytes or less (as the entire message is taken from the buffer). It also means that if you try to provide 21 data bytes for a message, plus a message count nibble, you will get an error (as you have over-filled the internal buffer). This is generally of no concern if you are implementing the SAE J1587 protocol, but may be an issue if you are trying to send some special SAE J1922 messages.

C0 and **C1** [Checksum display off* or on]

These commands are used to make the received checksum byte visible or not. Usually, it is preferred that the byte not be shown, so the ELM325 hides it by default. If you wish to see the byte printed with every message, simply send AT C1.

D [set all to Defaults]

This command is used to set the options to their default (or factory) settings, as when power is first applied. Any settings that you had made for a custom MID, for filters, for message formatting, or timer settings will be restored to their default values.

E0 and **E1** [Echo off or on*]

These commands control whether or not the characters received on the RS232 port are echoed (retransmitted) back to the host computer. Character echo can be used to confirm that the characters sent to the ELM325 were received correctly. The default is E1 (or echo on).

EM0 and **EM1** [Error Messages off or on*]

If a message is received and there is a problem with it, the ELM325 will add an error description to the received line. An arrow ('<') that points to the message is printed first, then either PROT ERROR, RX ERROR, or DATA ERROR is printed to describe the problem. You may find that the description is of little use to you in your application, and wish to only see the 'arrow'. If you set AT EM0, the ELM325 will do that for you.

F1 [reset Filter 1]

If you have set message receive filter F1 and wish to disable that setting, simply send AT F1. The ELM325 will discard any setting that you had made.

F2 [reset Filter 2]

If you have set message receive filter F2 and wish to disable that setting, simply send AT F2. The ELM325 will discard any setting that you had made.

AT Command Descriptions (continued)

F1 hh hh hh hh hh [set Filter 1 to…]

Filter F1 can be used to selectively receive only the messages that contain matching bytes in the first five byte positions. Simply set the values for the five bytes (ten nibbles) with this command and the ELM325 will make the internal adjustments for you. If there is a nibble that can have any value, then use the letter 'X' to define it. See the Receive Filtering section for more details.

F2 hh hh hh hh hh [set Filter 2 to…]

Filter F2 can be used to selectively receive only the messages that contain matching bytes in the first five byte positions. Simply set the values for the five bytes (ten nibbles) with this command and the ELM325 will make the internal adjustments for you. If there is a nibble that can have any value, then use the letter 'X' to define it. See the Receive Filtering section for more details.

GM [Get Message]

Occasionally, you may wish to set the receive filter for a particular message and 'get' it from the data stream. The AT GM command may be used to do so.

This command is very similar to AT MA with the F1/F2 filters set, except that it respects the AT ST timeout value. That is, if no message is received in the ST time, the ELM325 will return with a NO DATA message, and if multiple messages are received within that time, they will all be displayed. Note that F1 and/or F2 must be set prior to using AT GM.

GO [Get One message]

This command performs exactly as the AT GM command, except that it only gets one response and then returns immediately to the prompt.

I [Identify yourself]

Issuing this command causes the chip to identify itself, by printing the startup product ID string (currently 'ELM325 v1.0'). Software can use this to determine exactly which integrated circuit it is talking to, without having to reset the IC.

L0 and **L1** [Linefeeds off or on*]

This option controls the sending of linefeed characters (0x0A) after each carriage return character (0x0D). Users will generally wish to have this option on if using a terminal program, but off if using a custom computer interface (as the extra characters transmitted will only serve to slow the communications down). The default setting is L1 (linefeeds on).

MA [Monitor All messages]

This command places the ELM325 into a bus monitoring mode, in which it continually monitors for (and displays) all messages that it sees. If the F1 and F2 filters have not been set, then all messages on the J1708 bus will be displayed.

Often, it is not desirable to display everything that is being transmitted on the system. If you wish to reduce the amount of information displayed, simply set the F1 and/or F2 filters to the values that you wish to see. Then, with AT MA, the ELM325 will display all messages that meet that criteria.

MM hh [Monitor for MID hh]

The AT MM command is similar to the AT MA command except that it also filters for only messages that match the MID provided. That is, the AT MM command is effectively an extra filter that can be applied to incoming messages.

MP hh [Monitor for PID hh]

The AT MP command is very similar to the AT MM command except that it only allows messages with the provided PID value in the second byte position. Again, the F1 and/or F2 filters may be used to further refine the data that is shown.

MP 01hh [Monitor for page 2 PID hh]

This command is identical to the previous command except that it filters for page 2 PIDs (that is, for PIDs of value 0100 to 01FF). Again, F1 and F2 filters can also be used with this command.

As an aside, this command actually monitors for 0xFF in the second byte position, and for the hh value in the third byte position (which is the way that page 2 PIDs are defined in the SAE J1587 standard).

R0 and **R1** [Responses off or on*]

These commands control whether the ELM325 will look for a response from the vehicle, after a message has been sent. If responses have been turned off (with

AT Command Descriptions (continued)

AT R0), the ELM325 will not wait for a reply from the vehicle after sending a request – it will simply say 'OK', and will then return immediately to wait for the next command. An R0 setting will always override any 'number of responses digit' that is provided with a message.

R0 may be useful for sending commands blindly when simulating an ECU for demonstration or test purposes. The default setting is R1, or responses on.

RE0 and **RE1** [RE output to 0* or 1]

This command is used to control the state of the RE output pin. After a powerup or reset, the ELM325 always sets this pin to a low level. If you send AT RE1, you will change the output to a high level (ie to V_{DD}), and AT RE0 will restore it to a low level (V_{SS}).

While this pin was intended for use with RS485 transceiver 'receive enable' inputs, it does not need to be used in that way. This control pin functions independently of any internal circuitry so can be used for many other functions, such as disabling peripherals or blinking an LED.

S0 and **S1** [printing of Spaces off or on*]

This command controls whether or not space characters are inserted in the message response.

The ELM325 normally formats message responses as a series of two nibbles followed by a space (0x20) character. If you wish to reduce the amount of received data and your PC's processing time, you may wish to turn spaces off. By default, spaces are on (S1), and space characters are inserted in every response.

SM hh [Set the MID to hh]

The ELM325 normally assumes that you wish to send using the MID 'AC' (which is decimal 172). This value has been assigned by SAE J1587 to the #1 Off-board Diagnostics unit. If you wish to change the MID that the ELM325 uses when sending messages, simply define it with this command. See the 'Setting the MID' section for more information.

SP h [Set the Priority to h]

Each message that is sent by the ELM325 has a priority assigned to it. These priorities can have values from 1 (the highest) to 8 (the lowest).

The AT SP command may be used to change the message priority, but great care must be used with this. It is possible to preempt other more important messages if you change the priority, possibly adversely affecting the operation of the vehicle. Do not adjust this parameter if you are unsure of why you are doing so, and what affect it may have. By default, the ELM325 uses a priority value of 8.

ST hh [Set Timeout to hh]

After sending a request, the ELM325 waits a preset time for a response before it can declare that there was 'NO DATA' received from the vehicle. The same timer setting is also used after a response has been received, while waiting to see if any more are coming. The AT ST command allows this timer to be adjusted, in increments of 100 msec.

The ST timer is set to 0F (15 decimal) by default, which gives a time of 1.5 seconds. Note that sending AT ST 00 does not result in no time – it causes the default time to be set.

TC [monitor for Trouble Code messages]

This command is used to quickly monitor for trouble codes that are being broadcast – that is, all PIDs that are equal to C2 (or decimal 194), without having to set the filters.

The AT TC command uses the AT ST timer to limit the time that it waits for a message to arrive.

WS [Warm Start]

This command causes the ELM325 to perform a complete reset which is very similar to the AT Z command, but does not include the power on LED test. Users may find this a convenient way to quickly 'start over' without having the extra delay of the AT Z command.

Z [reset all]

This command causes the chip to perform a complete reset as if power were cycled off and then on again. All settings are returned to their default values, and the chip will be put into the idle state, waiting for characters on the RS232 bus.

Sending AT Commands

Before learning some J1587 Commands, we will show a few examples of how to use an AT Command. We will assume that you have built (or purchased) a circuit that is similar to that of Figure 7 in the Example Applications section.

Do not connect your circuit to a vehicle at this time. Power it from a test source only (a 9V 'transistor radio' battery works well), and connect it to your PC as discussed in the Communicating with the ELM325 section.

For your first command, simply reset the IC by sending AT Z. Try this a few ways (don't forget to press enter or return after each):

```
>AT Z
```

or

```
>atz
```

or

```
>a    T  z
```

You should see the RS232 LEDs blink as you type each letter, and after you press return (or enter) you should see all four Tx/Rx LEDs blink, in order, followed by a final blink of the RS232 Tx LED as the ELM325 sends 'ELM325 v1.0'.

Most J1708 messages are continually sent on the data bus, at a predetermined rate. Some messages may be sent 10x per second, while others are only 1x per 10 seconds. This means that you may need to adjust the internal timeout setting depending on what message you are attempting to receive. We will adjust this timeout setting next.

Try this request for trouble codes:

```
>AT TC
```

After about 2 seconds, you should see a response that looks like:

```
NO DATA

>
```

since there is no vehicle attached, there was no data received.

Now, adjust the timeout to 10 seconds. If you look at the AT command list, you will see that there is a Set Timeout command that is used for this. Timing is in increments of 100 msec (0.1 sec), so to obtain a 10 second delay, the AT ST setting should be 100. We need to convert the 100 to hexadecimal, however, as all numbers handled by the ELM325 must be in hexadecimal. Converting then, 100 (= 6 x 16 + 4) is 64 in hexadecimal, so we send:

```
>AT ST 64
```

Again, don't forget to press return (or enter). Now, the timeout should be set to 10 seconds. To verify this, repeat the Trouble Codes command:

```
>AT TC
```

It should be 10 seconds before you see the NO DATA response this time. To try it again, you do not need to enter the AT TC command again, you only need to press enter, and the ELM325 will repeat your last command (AT TC) for you.

As a final test, enter AT TC again, but before the 10 seconds is up, press any key on the keyboard. You should see the ELM325 respond with:

```
STOP?
```

which means that it was interrupted and it thinks that you wish to stop. If you ever see the 'STOP?' response, it means that the ELM325 thinks it has been interrupted by you.

Now, restore the AT ST time to the default value, with the Defaults command:

```
>AT D
```

You should see a response of 'OK' and then a prompt character on a new line, to show that the ELM325 is waiting for you. In this case, you might also have sent AT ST 00, since that also restores the timeout setting to its default value.

Experiment with these commands – you are not connected to a vehicle, so can do no harm. Sending AT Commands is not difficult, they just require a little practice.

J1587 Commands

The SAE J1708 standard defines how messages should be sent on what is essentially an RS485 data bus. The actual content of the messages are defined by the SAE J1587 standard (and the J1922 standard, too, but it is very similar). Our discussion from this point on will be in regard to the J1587 messages only (the ELM325 handles the rest of the conversion process for us).

J1587 messages always begin with a single byte called the Message ID, or MID. It is basically the sender's address (they are always predefined, and no two devices on the bus can have the same MID).

The next byte contains a Parameter IDentification number, or PID. It tells you what type of information is contained in the message. One or more data bytes follow the PID and are the actual content of the message being sent. Several PID/data groups can be contained in a single message as long as there are no more than 19 bytes of data used (that is, the message must be 21 bytes or less in length).

If multiple groups of data are transmitted in one message, there needs to be a way to separate them when received, and the SAE J1587 standard sets out how this is done. PIDs 00 to 7F (ie 0 to 127), always have one data byte only, while PIDs 80 to BF (128 to 191) have two data bytes. PIDs C0 to FD (192 to 253) can have a variable number of data bytes – the number is always given in the first byte following the PID. The pattern repeats again for the page 2 PIDs (0100 to 017F have one byte, 0180 to 01BF have two, etc.)

PID FE (ie 254) is a special one called the data link escape PID. The data contained within that message is manufacturer specific.

Finally, PID FF (ie 255) is used to signal that all the PIDs in that message are in page 2. That is, PIDs are always defined with one byte, modulo 256. To say that all of the PIDs referred to in the message are actually in the range 0100 to 01FF (256 to 511), the first PID (ie. the second byte in the message) will be FF (255).

The final byte of a message, after all the PID and Data groups, contains a checksum. This helps the receiver to detect if errors have occurred in the data that is received. The ELM325 does not normally show this byte when receiving a message, but you can have it displayed if you send the AT C1 command.

Figure 1 below shows pictorially how a J1587 message is formed. If you wish to learn more about the SAE J1587 standard, you may purchase a copy from www.sae.org.

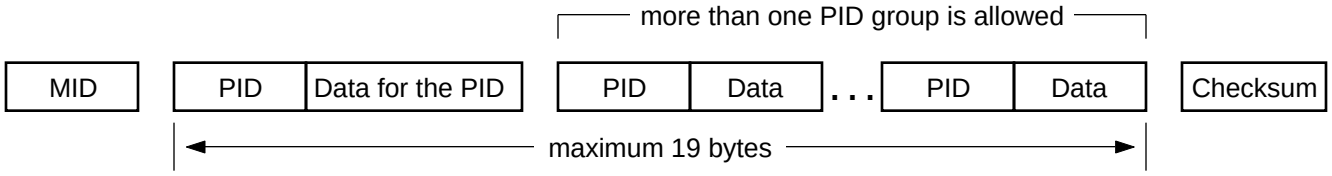


Figure 1. The SAE J1587 Message Structure

Listening to a Vehicle

Before we begin, it should be noted that the ELM325 always uses hexadecimal numbers when communicating with the PC. That is, while the J1587 standard may talk of engine #1 being assigned a MID of 128, the ELM325 always uses 80 for that same address. If you are not familiar with the hexadecimal numbering system, you should review it before proceeding.

Listening to a vehicle with the ELM325 is exactly the same as with the other ELM OBD ICs – simply tell the IC that you want to monitor all data:

```
>AT MA
```

and the chip will begin displaying all the data that it sees. (If you are connected to a 'live' vehicle, but are not seeing data, you may have reversed the polarity of your J1708 connections.) To stop the flow of data, simply press any key on the keyboard.

Perhaps this is too much data and you wish to only see data being sent by the engine. You may easily monitor for only that MID, by sending:

```
>AT MM 80
```

and you should then only see data that begins with MID 80. Note that the ELM325 does not show the checksum by default – you need to tell it to show that byte. For example, monitoring for MID 80 might show:

```
>AT MM 80
80 5C 66 BE E0 2E
80 B7 40 06 5C 64 BE 30 2F
80 5C 64 BE 30 2F
etc.
```

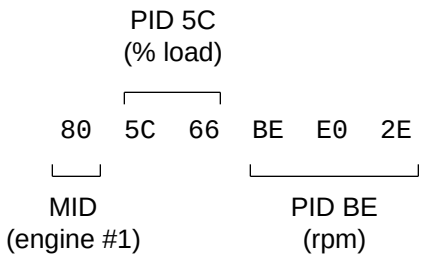
if you wish to see the checksum byte, send AT C1 then monitor for MID 80:

```
>AT C1
OK

>AT MM 80
80 5C 66 BE E0 2E F2
80 B7 40 06 5C 64 BE 30 2F A6
80 5C 64 BE 30 2F A3
etc.
```

While other J1587 monitors tend to always show the checksum, the ELM325 hides it by default (as it doesn't really help in assessing the data).

Just what is the information contained in these messages? If you look at the first line that was received (with the checksum turned off), you can break it into components as follows:



The first byte (80) represents the engine #1 MID, while the second byte is always a PID (5C in this case). Since 5C is in the range from 00 to 1F, there is one data byte following the PID (the 66) before the next PID begins. The next PID is then BE, which is in the range 80 to BF so it should have two data bytes associated with it (which it does).

What does this data mean? Well, 5C is equivalent to 92 in decimal. Looking up this PID in J1587, one finds that it is for the % engine load, and each digit represents 1/2 %. Converting 66 to decimal gives 102, so the % engine load is 102 x 1/2% = 51%.

The second PID is BE (190 decimal), which is for engine speed, with each digit representing 1/4 rpm. Simple enough, but how do you interpret the two data bytes? ie Should they be E02E or 2EE0? It turns out that the J1587 standard uses what is known as 'little endian' data representation, so the first byte that you see is the least significant (little) one. The engine rpm data should be interpreted as 2EE0, or 12000 decimal. With 1/4 rpm per digit, the speed is then 3000 rpm.

The other lines received (or other messages) can be analyzed in a similar fashion. Start with the first PID, and work your way through. If the first PID happens to be FF, you will need to start with the next byte (the third one), and you must then treat all PIDs in the message as if they are page 2 PIDs (ie in the range from 0100 to 01FF, or 256 to 511).

That's about all there is to interpreting the data received. It is almost essential to get a copy of the SAE J1587 standard if you are going to do much interpreting of the data, as there are hundreds of MIDs and PIDs that are defined in it.

Receive Filtering

The ELM325 is also capable of monitoring for a particular PID, instead of for a MID as we showed in the previous section. For example, if you would only like to see messages with 5C in the first PID position, you might send:

```
>AT MP 5C
```

and the ELM325 would display only messages that it receives that have 5C in the second byte position, such as:

```
80 5C 66 BE E0 2E
80 5C 64 BE 30 2F
etc.
```

Note that the current version of the ELM325 only displays PIDs that are in the second byte position and is not able to follow the PID chain. That is, it can not detect the PID 5C in a message such as this:

```
80 B7 40 06 5C 64 BE 30 2F
```

You will need to manually assess messages that contain multiple PIDs, or write software to do so.

When monitoring for page 2 PIDs, the procedure is very similar. For example, to monitor for engine oil level (PID 366 or hex 016E), simply send:

```
>AT MP 016E
```

(don't worry about the extension PID as the ELM325 takes care of that for you).

Being able to monitor for a PID or a MID may be helpful at times, but it would often be more helpful to monitor for both a MID and a PID. You can not do this with the AT MM and AT MP commands, but you can by setting a receive filter.

The ELM325 has two special purpose receive filters built in. These filters act on every received message, and only allow messages with matching bytes to pass. Both F1 and F2 allow a five byte pattern to be defined.

What if you would like to see a range of values, so do not wish to define all five bytes of the pattern? Is it possible to define only the MID and the PID and let messages that match those bytes pass? Certainly. Simply tell the ELM325 that you do not care about those extra bytes (by providing an 'X' for each nibble that you do not care about). For example, to filter for all messages that are from MID 80, and contain a 5C in the first PID position, simply say:

```
>AT F1 80 5C XX XX XX
```

From that point on, all received messages must start with 80 5C, or they will be discarded. Note that this will pass messages of two or more bytes in length (not including the checksum byte), since you told the ELM325 that you did not care what values the other bytes had, if any.

Since this filter will act on all received messages,

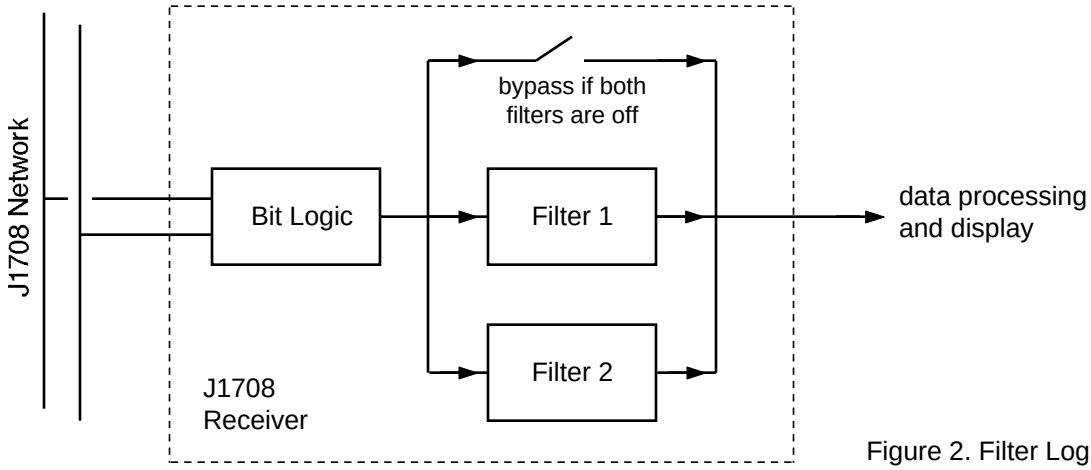


Figure 2. Filter Logic

Receive Filtering (continued)

you may use it with any of the AT Commands that are available. For example, to see all messages that the J1708 Receiver logic is now passing, simply send:

```
>AT MA
```

Of course, you may still use AT MM or MP when a filter has been defined, but then the received message must also meet the MM or MP condition.

As shown in Figure 2, the ELM325 also contains a second filter that you may define in exactly the same way. If a second filter is defined, then messages that meet either the F1 or the F2 criteria are allowed to pass on (and if neither filter has been defined, all messages will be allowed to pass).

There are a few more examples of how to define these filters in the next section (Getting Trouble Codes) and also in the Automatic Receive Filters section. Creating filters is not very difficult if you know what you expect the received message to 'look' like.

Getting Trouble Codes

Trouble (diagnostic) codes are usually broadcast over the J1708 network so that all devices can know of problems. They will be transmitted when a problem is first discovered, while it is active, and when it clears. For this reason, all that you should normally have to do is monitor for the Trouble Code PID (it is C2 or decimal 194). Since this is likely to be a very common use of the ELM325, we have built in a special function just for this. To monitor for currently active trouble codes, simply send the command:

```
>AT TC
```

and the IC will configure the filters that you need for receiving them. Note that if you have defined the filters previously for another function, the ELM325 will not change your settings. It will instead treat this command as if you had entered AT GM, so be careful.

If there are no trouble codes found in the time set by the AT ST timer, the AT TC command will return with a response of 'NO DATA.' If there are trouble codes found, they will be printed out for you.

The AT TC command doesn't do anything that you can not do, it just saves a few steps for you. If you wish to do exactly as the TC command does, try this:

```
>AT F1 XX C2 XX XX XX

>AT F2 XX C0 XX C2 XX

>AT GM
```

The filtering for C0 in the F2 setting is done so that any long responses (multi section parameter IDs) will also be detected and displayed.

The AT GM command is actually quite a handy one to use when you wish to look for specific messages on the J1708 bus. Simply set the filters as required, ensure that the AT ST timer is set properly (some PIDs are only broadcast every 10 seconds), and then send AT GM. Of course, you could also set the filters and then use the AT MA command, but that command does not time out – you must manually stop it by pressing a key on the keyboard (or sending one from your program). The AT GM command will stop waiting for a message after the AT ST time, which is often more useful. A variation of this command, the Get One (AT GO) is also very useful as it acts exactly like AT GM, except that it only gets one response before returning immediately to the prompt.

We have discussed the monitoring for trouble codes that are being broadcast over the network, but what if you should wish to actually ask the modules for trouble codes? You can do this easily as well – all you need to do is send a request like this:

```
>C2
```

and the ELM325 will return with either a response from the MID(s), or it will will say 'NO DATA'. The sending of requests such as this is discussed further in the next section (Making Requests).

Making Requests

Using the ELM325 to send messages on the J1708/J1587 bus is very similar to sending OBD messages with our other products (the ELM327, etc.). Simply provide the data bytes that you want sent and the ELM325 takes care of the details for you.

For example, assume that you wish to send the bytes 11 22 33 44 for some reason. Simply provide them to the ELM325 from the prompt:

```
>11 22 33 44
```

and the IC will add the MID and checksum for you before sending the entire message. You can provide as many as 21 data bytes (42 digits) to be sent in this manner.

As a another example, assume that you wish to make a request for PID BE data (this is the engine rpm PID that was discussed previously). PID 00 is used to request a parameter, so to request parameter BE, simply send:

```
>00 BE
```

The ELM325 adds your MID for you, as well as the checksum at the end, then transmits the entire message. Note that the MID and checksum are only added if you have left the auto formatting on – if you have turned it off, only the bytes that you provide will be transmitted.

Similarly, if you wish to obtain % engine load (also discussed previously), then send:

```
>00 5C
```

What if the information being requested is currently being broadcast regularly, and you only wish to see one message in response, not many? The ELM325 allows you to specify a single digit for the number of responses that you wish to see. Just add it after the bytes. For example, to see only one response for the above request, send:

```
>00 5C 1
```

and the IC will return with a prompt after receiving only one message. The count that you provide is a hex digit, and can have values from 0 to F.

Since these requests for parameters may be very common, we have simplified matters a little for you. The ELM325 always monitors what you are sending, and if it finds that you are sending a single byte only (while auto formatting is on), it will assume that you are making a request for a parameter, and it will add the 00 for you. That is, to request parameter BE, simply send:

```
>BE
```

and to obtain only 1 response for parameter 5C, send:

```
>5C 1
```

The page 2 parameters normally add a little complexity to your sending, but with the ELM325's automatic formatting, they are again just a matter of entering the PID number. To request the engine oil level (PID 016E), simply send:

```
>016E
```

and if you wish to limit the number of responses to say 3, send:

```
>016E 3
```

Note that the automatic formatting will only change your data bytes if you send a single byte or two bytes that begin with 01. All other messages will remain unaffected, and will be transmitted exactly as provided.

This of course leads to the question 'but what if I want to send a single byte message, or a two byte message that begins with 01?'. Then you will need to turn the automatic formatting off and send the entire message yourself (you will need to provide the MID and the checksum too). Fortunately, it would be a very special case if you wanted to send only a PID number or if you wanted to send a single byte with PID 01 (which has been discouraged since PIDs 194 to 196 were introduced in 1988).

The above said to simply send a few bytes and all would be well, if the auto formatting was on. But how does the IC know what messages to look for in the response? That is, what about setting the filters so that the ELM325 can selectively receive only the desired response? That is taken care of for you too, even if the auto formatting is off. The next section discusses this.

Automatic Receive Filters

The ELM325 receiver always accepts bytes from the J1708 data stream, and then decides if they should be passed on to the other logic, or discarded. This decision to pass the messages on or not is done with selective logic called filters. You can set these selective filters yourself (see the Receive Filtering section), or allow the ELM325 to do it for you.

If you do not set either receive filter, and send a message, the ELM325 will choose filter settings for you based on the content of your message (the automatic formatting does not have to be on for this to happen). The settings may not be perfect for every situation, but they will be acceptable for most. The chart in Figure 3 shows what the settings will be for different messages that you might send.

If you do not agree with the filter settings that the ELM325 chooses for you, then simply define your own, using the AT F1 and AT F2 commands. Also if you have already chosen filters, and would now prefer that the ELM325 choose them for you, then turn off the filter(s) that you have enabled. To turn off filter 1, for example, send the following:

```
>AT F1
```

Note that there are no parameters provided with this command – just the AT and the F1. Similarly, disabling F2 is done with AT F2.

Seeing how the ELM325 sets its filters should at least provide a starting point, if you are looking for specific data. With a little experimentation, you will find that it is not that difficult to set the filters, and so limit what is printed by the ELM325.

Description	the send data looks like...	Filter 1 Filter 2
PID Request	`MID 00 PID`	`XX PID XX XX XX` `XX C0 XX PID XX`
Extended PID Request (page 2)	`MID FF 00 PID`	`XX FF PID XX XX` `XX FF C0 XX PID`
Component Specific (page 1)	`MID 80 PID RxMID`	`RxMID PID XX XX XX` `RxMID C0 XX PID XX`
Component Specific (page 2)	`MID FF 80 PID RxMID`	`RxMID FF PID XX XX` `RxMID FF C0 XX PID`
Diagnostic Data Request	`MID C3 03 RxMID PID`	`RxMID C4 XX XX XX` `RxMID C0 XX C4 XX`
All other requests	(data that does not match any of the above)	`XX XX XX XX XX` (not used)

Figure 3. Filter Settings When Sending Messages to the ECU

Setting the MID

The Message ID (MID) is a single byte value that is assigned by the SAE Truck and Bus Low Speed Communications Network Subcommittee. No two transmitters on the network may use the same ID, so for this reason, care should be used if you are experimenting with different values of it.

The ELM325 uses the MID for the 'Off-board Diagnostics #1' device by default. That is hex value AC, or decimal value 172. If you wish to send messages using another MID, it is easily done with the Set MID command. For example, to use B4 (decimal 180) which is for the Off-board Diagnostics #2 unit, then simply send the Set MID command:

```
>AT SM B4
```

Every message that the ELM325 sends from that point (as long as auto formatting is on) will use the MID B4, instead of AC. Of course, you could also pretend to be an engine or transmission, but you would almost certainly cause problems, unless you were in a teaching situation, with controlled conditions.

The MID assigned in this way stays in effect until the ELM325 restores its default values. This could be through the use of the AT D command, through AT WS or AT Z, or from a power off and on. This version of the ELM325 does not provide Programmable Parameters (like the ELM327 or the ELM329), so a preferred value (other than AC) can not be stored in EEPROM.

Multiline Responses

There are occasions when a vehicle must respond with more information than is able to fit in a single 'message'. In these cases, it responds with several data frames which the receiver must assemble into one complete response. The following shows how this is done with the SAE J1587 protocol.

Figure 3 of the Automatic Receive Filters section showed that the ELM325 looks for two types of responses to a PID request. In the first, the requested PID number appears in the very first data byte position. This is the standard reply where a response of 1 to 18 data bytes may be expected. The second response that it looks for is one where the data begins with the hex digits C0. The C0 PID is a special one that is used to signify that the response is a special multisection one, and that the total number of data bytes will be something greater than 18.

Figure 4 on the next page describes how the C0 PID encodes the data contained. The very first message (ie. section 0) has one extra byte to show how many bytes in total are in the response. The next lines received are identical to the first, except that they do not have this one byte (and so have room for one more data byte).

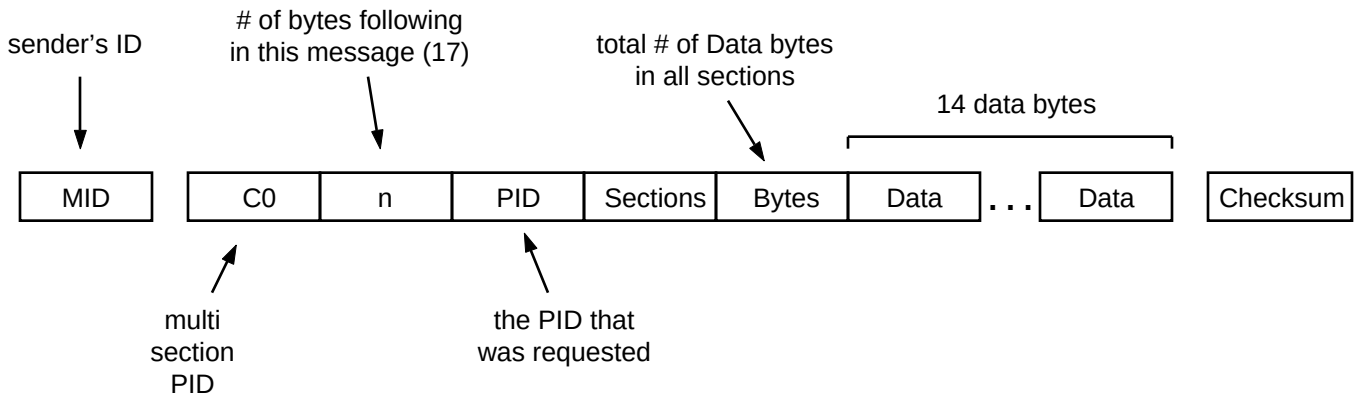
In total, there can be as many as 15 sections in one message, so the maximum number of data bytes that can be sent in this way is 14 + 15 x 15 = 239 . If a vehicle needs to send more than that, the J1587 protocol has made provision for it with the Transport Protocol. While the ELM325 does not explicitly support the Transport Protocol in firmware, you can support it with software (you will need one filter to watch for C5's and the other to get C6's).

The diagram in Figure 4 makes it fairly easy to see what each byte is doing, but again, you may wish to refer to the SAE standard for more complete details. We'll just add a few notes for some of the more subtle points:

- The ELM325 does not assemble these messages for you – it must be done manually or with software.
- There may be more than one MID responding to a request for a PID.
- The segments are all sent in sequence, and you can not request random resends.

Multiline Responses (continued)

The first section:



All the other sections:

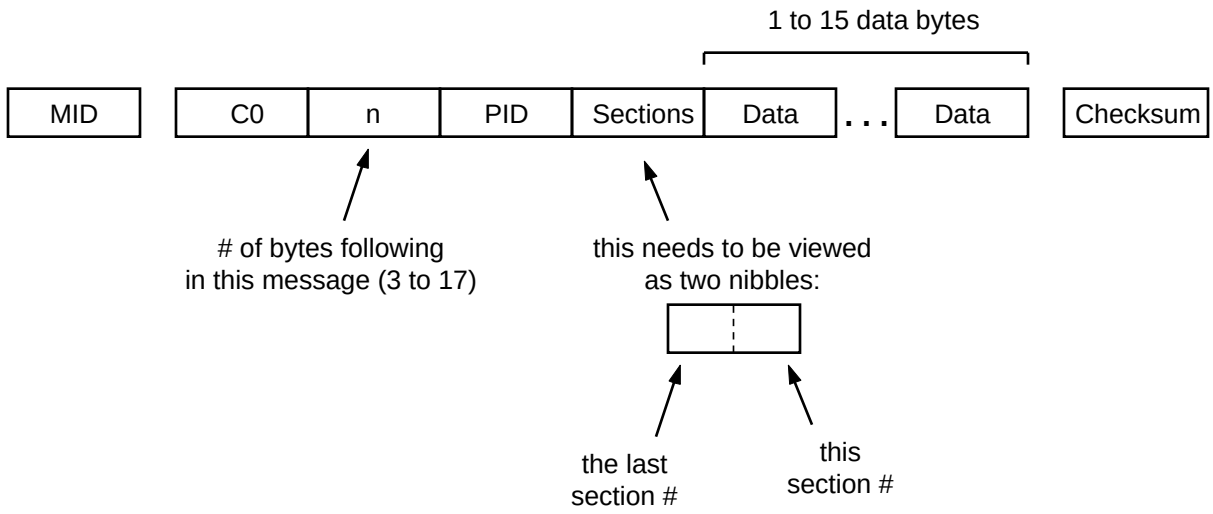
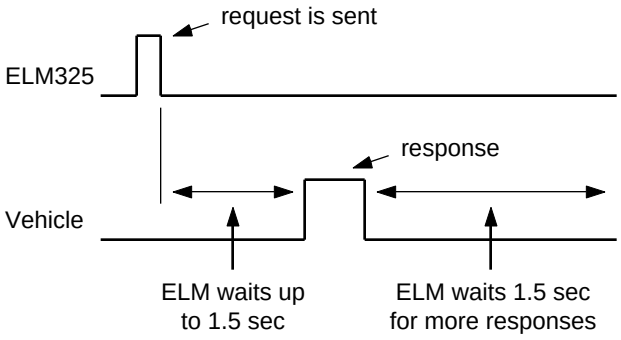


Figure 4. The Multisection Response

Setting the Timeout

Users often ask about how to obtain faster scanning rates when obtaining information from a vehicle. There is no definite answer for all vehicles, but you can improve the response rate if you understand the ELM325 internal timing process.

A typical vehicle request and response is shown here:



The ELM325 sends a request then waits up to 1.5 seconds for a reply. If no reply arrives in that time, an internal timer stops the waiting, and the ELM325 prints 'NO DATA'. Even after a reply has been received, the ELM325 must wait to see if any more replies are coming (and it uses the same internal timer to stop the waiting if no more replies arrive). While most replies should be received within the default 1.5 seconds, the setting is adjustable so that you can cater to almost every situation. (It can be adjusted in increments of 100 msec from 0.1 sec to 25.5 sec.)

As an example, consider a vehicle that responds to a query in 100 msec. With the ST timeout set to 1.5 sec, the fastest scan rate possible would only be about 1 query every 1.6 seconds, since the IC always waits for that extra 1.5 seconds after receiving a response. Changing the ST timer setting to 200 msec would make the IC and the vehicle seem much more responsive, as the total time taken would only be about 300 msec (but it would miss any messages that take longer than 200 msec to appear).

It is not easy to determine how long it takes for a vehicle to respond to requests, so it may require a bit of experimentation to find a setting that works well for you. Setting the timeout too low may also mean that some important information (ie. trouble codes) may be missed, since they are often broadcast every second, so be careful (we have chosen the setting of 1.5 seconds purposely to ensure that messages such as trouble codes are easily detected).

If you want to set the timer to a low value, but do not want to set it too low (and miss some responses), then what is to be done? One option is to decide how many replies would be adequate (usually 1), and include that in your query. That is, instead of sending say:

```
>BE
```

for the engine rpm, send:

```
>BE 1
```

instead. That way, the ELM325 will send the request, set the timeout timer to 1.5 seconds (or whatever the AT ST time has been set to), and will then look for one response. If a response never arrives, you will see a 'NO DATA' message, but if a response is obtained, the ELM325 will return immediately to the prompt state, ready for another command.

By adding the number of responses to a message, you always eliminate the final wait while the ELM325 sees if any more responses are coming. This means that every response is effectively shortened by that ST time. The single digit can have a value of 0 to F, and may follow any hex bytes that you send (it doesn't work with AT commands). Note that the AT GO command actually uses a setting of 1, so it obtains one response and returns immediately. This might be useful when polling for some data.

Since you may encounter messages that are only broadcast every 10 seconds, you may have to adjust the AT ST timer in order to see them. You should not be reluctant to do so, however, as this does not mean that the ELM325 always takes that much time, if you provide the number of responses to get.

About J1922

SAE J1922 is another standard that may be used by heavy duty trucks and busses. The data format is similar enough to J1587 that you may use your ELM325 circuit with it.

J1922 uses a number of predefined messages for communicating status and performing various control functions. These are either broadcast at a regular rate, or provided on demand, just like J1587. Unlike J1587, they do not use PIDs for defining functions though – they do that with the MID byte. The J1708 standard defines MIDs 45 to 56 (ie. 69 to 86 decimal) for use with J1922).

The messages may be from 2 to 23 bytes in length with J1922 (21 data bytes are actually allowed by J1708 if the vehicle's engine is not running, and it's not moving). This is not an issue with the ELM325, as it is capable of receiving an unlimited number of bytes, and is able to send as many as 21 data bytes (23 total) if the automatic formatting is on.

When monitoring for J1922 messages, use the same commands as you would with J1587 – MA, MM, GM, etc. will all work. Note that AT MP will still work too, but it will filter for the first data byte (which is likely of minimal use). The only concern might be that if you are sending messages and looking for responses, you may have to define your own F1 and F2 filters (see the Receive Filtering section for more information on this).

Restoring Order

There may be times when the ELM325 settings have been adjusted, and it's not responding properly. Perhaps you are not sure of the present settings (but you do know that you were getting responses before, and are now not seeing any). Perhaps you have told the ELM325 to monitor all data, and there are screens and screens of data flying by.

The ELM325 can always be interrupted from a task by a single keystroke from the keyboard. As part of its normal operation, the ELM325 constantly checks for any received characters on the serial port, and if found, it will stop what it is doing at the next opportunity. This may mean that it will continue to send the information for the current line, then stop, print a prompt character, and wait for your input. The stopping may not always seem immediate if it has just begun printing a line, for example, so be patient.

There may be times when the problems seem more serious and you don't remember just what you did to make them so bad. Perhaps you have 'adjusted' the timer, and experimented with the filters, or perhaps tried to see what happens if the MID byte is changed. To reset only the filters to their initial state, simply send:

```
>AT F1
```

then

```
>AT F2
```

and the ELM325 will remove any settings that you have made to either filter.

If the problem is a little more involved than this, then all of the settings can be reset by sending the 'set to Defaults' command:

```
>AT D
```

This is usually enough to restore order, but of course it removes all of the settings that you have made (echo, linefeeds, etc), so should only be used when you truly want all the settings to be restored to their default values.

If the AT D command still does not bring the expected results, it may be necessary to do something more drastic – like resetting the entire IC. There are a few ways that this can be performed with the ELM325. One way is to simply remove the power and then reapply it. Another way that acts exactly the same way as a power off and then on is to send the full reset command:

```
>AT Z
```

It takes approximately one second for the IC to perform this reset, initialize everything and then test the four status LEDs in sequence. A much quicker option is available, however, if the led test is not required – the 'Warm Start' command:

```
>AT WS
```

The AT WS command performs a software reset, restoring the same items as AT Z, but it does not perform the LED test.

Microprocessor Interfaces

Many applications will require that the ELM325 interface directly to a microprocessor, and not connect to a PC. This is not a problem, especially if you use the same power supply for both circuits.

The ELM325 is actually a microprocessor that contains a standard UART type interface, connected to the RS232 Tx and Rx pins. The logic type is CMOS, and this is compatible with virtually all TTL and CMOS circuits, so you should be able to connect directly to these pins, provided that the two devices share the same power supply (5V, 3.3V, etc - it does not matter).

The normal (idle) levels of the ELM325 transmit and receive pins are at the V_{DD} level. Almost all microprocessors and RS232 interface ICs expect that to be the idle level, but you should verify it for your microprocessor before connecting to the ELM325. The connections are straightforward – transmit connects to receive, and receive connects to transmit, as shown in Figure 5 below. Don't forget to set both devices to the same baud rate (57.6K).

Your microprocessor and the ELM325 should not be physically more than about 10 to 20 inches apart, as CMOS circuits are subject to latchup from induced currents, and this may be a problem if you have too much separation. If you must have a large distance between your circuits, consider placing 1KΩ resistors at the Rx and the Tx pins of both the ELM325 and the microprocessor. That is, put 2KΩ of resistance in series with each lead, with half of each resistance (1KΩ) physically located as close to each IC as possible. By limiting the current available, you should be able to reduce the chance of a latchup.

That's about all there is to connecting the ELM325 to a microprocessor – simply share the supplies and watch your wiring lengths.

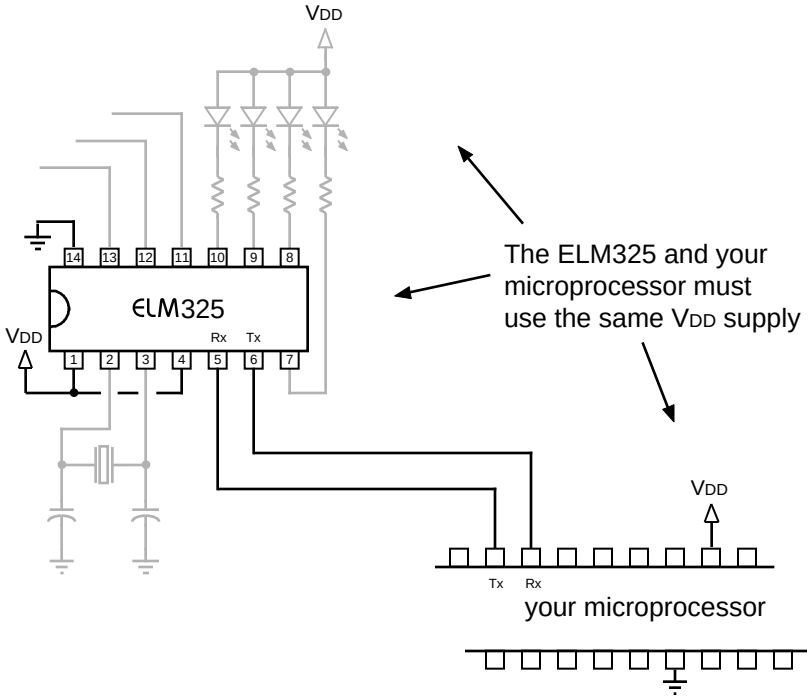


Figure 5. Microprocessor Direct Connection

Example Applications

To use the ELM325 in a circuit requires only a few interface components and a power supply, as shown in Figure 7.

The circuit obtains its power from the J1708 (vehicle) interface. The pins labelled A, B, C, etc. refer to the connections on standard 'Deutsch' connectors (refer to the section 'Tester Connectors' for more on them). As shown, battery positive is connected through diode D1 to a standard 7805 regulator, which in turn provides the 5 volt supply for the entire circuit. The ELM325 can operate over a wide range of voltages, but 5 volts is a standard, and readily available value. Strictly speaking, diode D1 is not required for circuit operation, but it is a good idea to add it in order to protect against reversed connections. While the J1708 pin connections are now standardised, early ones were not and you never know if there might be one with a reversed connection to the battery (terminals C and E on the 6 pin.

The 7805 IC is typically able to withstand 35V at it's input without damage, but not all regulators are this capable. When choosing your regulator, be sure to make sure that the input is able to withstand higher battery voltages (not just rated voltage, but also the spikes that occur due to loads being 'dumped', etc.). As to current carrying capacity, the 7805 regulator is more than adequate for supplying this circuit (a 78M05 would do too), it is fairly inexpensive, and it is readily available.

Figure 7 shows a 'power on' LED (L5) connected between the output of the 7805 regulator and circuit common. This provides an indication that voltage is present, which is generally a good thing to provide in a circuit such as this. Another alternative might be to connect the LED and resistor between 5V and pin 12 of the ELM325 (which is the RE or Receiver Enable output). In this way, you could show that power is on, and that the RS485 receiver is also enabled (so the ELM325 is ready). If you do not use the pin 12 output to enable the RS485 receiver, then the RE output could be dedicated to the control of the LED from your software. This presents several opportunities for providing feedback to the user.

There are really very few connections to the ELM325 itself. A crystal is needed to maintain the timekeeping functions (a ceramic resonator might also be used, with only a slight reduction in frequency accuracy). Either way, the frequency required is that of an NTSC 'colourburst' signal. That is, it needs to be 3.579545 MHz. This is often shortened to simply 3.58 MHz. Loading capacitors C1 and C2 are shown connected to the crystal. Your values may vary slightly from this, but 27 pF provides about 15 to 20 pF of crystal loading, which is what is typically required.

Four LEDs are shown connected to the ELM325 through current limiting resistors. 470 Ω is likely a good starting point for the resistor as that allows 4 to 5 mA of LED current, but you may want to change the value depending on the chosen V_{DD}, and whether you want a different brightness. This might be an idea if you wish to reduce total current consumption for the circuit, for example. Note that the perceived brightness will not change that much if you increase the current from that shown (but you are free to try it).

U2 (a standard RS485 transceiver IC) is shown connected to pins 11, 12 and 13 of the ELM325. If you plan to operate with the transceiver enabled at all times, then you really don't need the connection from pin 12 of the ELM325 to pin 2 of the DS485. Simply leave pin 12 open-circuited, and connect U2 pin 2 to circuit common to have the transceiver constantly enabled.

While we show a DS485 integrated circuit for the interface IC, there are many RS485 transceiver ICs on the market. The DS485 shown is produced by National Semiconductor (www.national.com), as is the DS75176B, and the DS36277. Maxim Semiconductor (www.maxim-ic.com) produces the MAX483/487 family of parts, while Linear Technology makes parts like the LTC485. Analog Devices (www.analog.com) makes parts like the ADM485 family, the ADM2582E and ADM2587E, as well as the ADM2484E. Be sure to see Texas Instruments (www.ti.com) for devices like the SN65HVD3080E. These are all very capable devices.

No matter which transceiver IC that you decide on, it needs to be connected to the J1708 bus through an R-C network as shown (C6, C7 and R6 to R9). This network is specified in the SAE J1708 standard so should be adhered to. It provides some filtering for slew rate limiting and protection, as well as pullup and pulldown for the J1708 data bus. Note that while RS485 systems normally use terminating resistors between the two signal wires (ie. data lines), the J1708 standard specifically states that terminating resistors such as that are not to be used.

The final part of Figure 7 is the serial to USB converter, U4 (a DB9-USB-D5-F), which is a product of Future Technology Devices International which is also known as FTDI (see www.ftdichip.com). This is a handy little circuit that is built into a DB9 shell (that is,

Example Applications (continued)

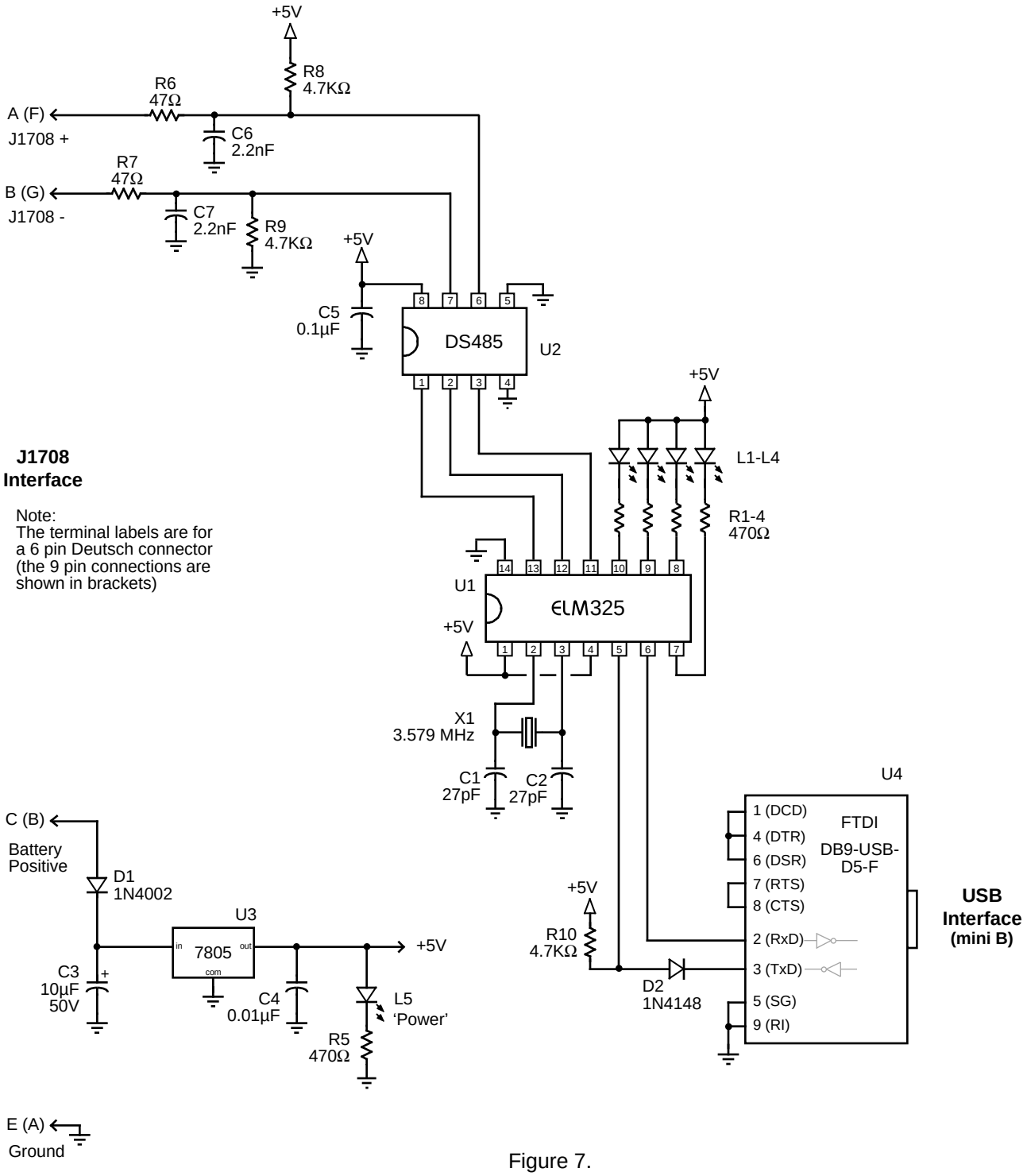


Figure 7.
A J1708 to USB Interpreter

Example Applications (continued)

it looks like a 9 pin serial connector, with a mini USB socket where the cable normally connects). It obtains it's power from the PC through the USB cable, so is not a burden on the ELM325 circuit. In fact, the FTDI module will even try to power the ELM325 when connected, which is why we added D2 and R10 – to block any reverse current from flowing back into the ELM325.

There isn't much more that that we can say about the FTDI module, except that it works as advertised, and works well. It requires software to be installed before you can use it, but that is available at their web site (www.ftdichip.com) for Macintosh, Windows, and Linux. Go to Drivers, then choose VCP Drivers for the needed virtual com port support. Once the software is installed, you may 'talk' to the ELM325 circuit with standard serial interface software (such as HyperTerminal or ZTerm), or with custom interface software.

There are other RS232 to USB solutions available such as the CP2102 from Silicon Laboratories (www.silabs.com), or you might try other devices such as a Bluetooth or Wi-Fi module. We only show the FTDI product here as one possible option.

The final circuit that we offer is that of Figure 9. It is identical to that in Figure 7, except that we show a discrete RS232 interface connected to the ELM325's pins 5 and 6. This circuit is more than adequate for this application, and is relatively low in cost. (It is also the same as the one that we suggest with our ELM327 and ELM329 integrated circuits.) Of course, if you want to reduce the amount of wiring, you may wish to consider products like the MAX232 family from Maxim Integrated Products (www.maxim-ic.com), and similar devices.

That should get you started with the ELM325. It actually needs very few components to make a fully functioning circuit, so should not be that difficult or expensive to build. If you should need help, check the help pages on our web site, or write an email to our technical support (techsupport@elmelectronics.com) and we'll do what we can to get you going.

Semiconductors
D1 = 1N4001
D2 = 1N4148
L1, L2, L3, L4 = Yellow LED
L5 = Green LED
U1 = ELM325
U2 = DS485 (RS485 transceiver)
U3 = 7805 (5V, 1A regulator)
U4 = FTDI DBP-USB-D5-F (USB I/F)

Resistors (1/8W or greater)
R1, R2, R3, R4, R5 = 470 Ω
R6, R7 = 47 Ω
R8, R9, R10 = 4.7 KΩ

Capacitors (16V or greater, except as noted)
C1, C2 = 27pF
C3 = 10µF 50V
C4 = 0.01uF
C5 = 0.1µF
C6, C7 = 2.2nF (2200pF)

Misc
X1 = 3.579545 MHz crystal
6 or 9 pin Deutsch connector
8 pin DIP socket
14 pin DIP socket

Figure 8. Parts List for Figure 7

Example Applications (continued)

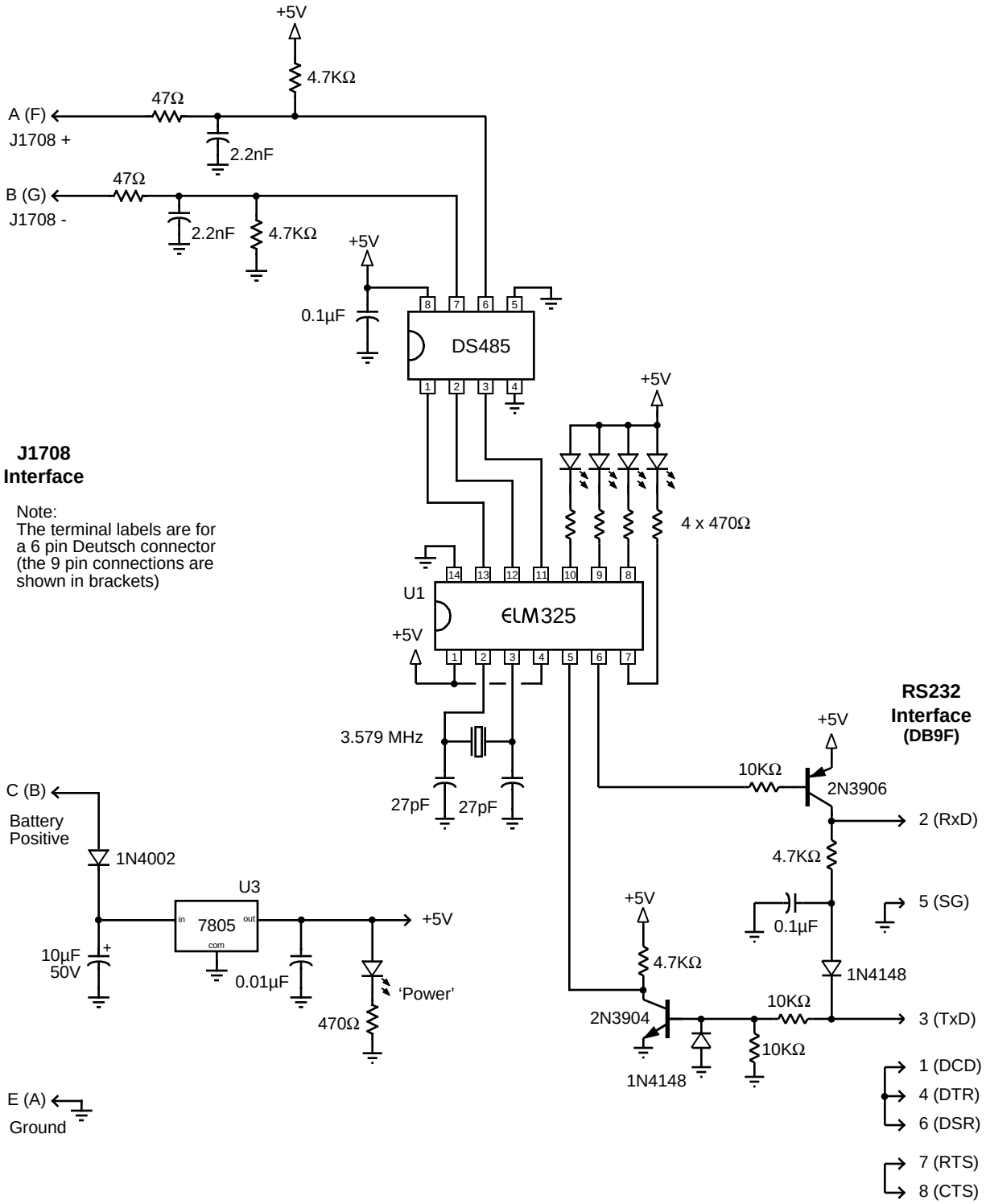


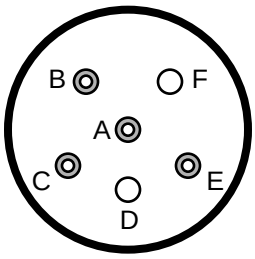
Figure 9. An RS232 Serial Interface

Tester Connectors

The tester that you make will need to connect to the vehicle's diagnostics port using a cable and special connector. You may need a few connectors for this, as the types used by the manufacturers have varied over the years. The two most common ones that you will find are the 6 pin, and the 9 pin 'Deutsch' connectors. Deutsch Industrial is a multinational company that makes products for use in harsh environmental conditions.

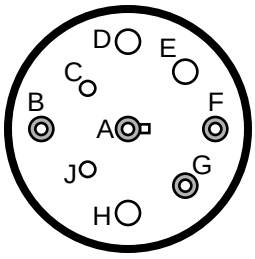
The following shows the pin configurations for the 'Deutsch' connectors, and lists part numbers for each. Should you wish to make your own connectors, you will need to buy a shell (or casing) as well as the metal pins to insert into it.

While the Deutsch Industrial products are what is normally specified for these connectors, you may find the Amphenol parts to be an acceptable alternative and possibly easier to obtain in your area. We provide a Digi-Key (www.digikey.com) part number in brackets under each Amphenol part to help you if ordering from them.



6 pin 'Deutsch' test connector
(looking into the open end)

	Deutsch Industrial	Amphenol Sine Systems
Shell	HD16-6-12S or HD16-6-96S	AHD16-6-12S (889-1064-ND) or HD16-6-12S-B010 (889-1063-ND)
# 12 Socket Pins	1062-12-0122	AT62-210-1231 (889-1053-ND) or 65-54748 (889-1057-ND)



9 pin 'Deutsch' test connector
(looking into the open end)

	Deutsch Industrial	Amphenol Sine Systems
Shell	HD16-9-1939S	AHD16-9-1939S (889-1059-ND)
# 16 Socket Pins	1062-12-0122	AT62-16-0622 (AT62-16-0622-ND) or AT62-16-0122 (889-1049-1-ND) or 65-54757 (889-1055-ND)

Error Messages and Alerts

The following describes the messages that this version of the ELM325 sends to tell you of a problem.

BUFFER FULL

The ELM325 provides an internal memory space (or 'buffer') for temporarily storing data bytes before they are transmitted as RS232 data. The state of this buffer is continually monitored and a warning is given if it should become full (as data will be lost if that continues).

If you are receiving BUFFER FULL messages, then you may be trying to connect to a system that is operating at a non-standard baud rate, or possibly has excessive noise. You may reduce the amount of data transmitted (and so reduce the amount of data that needs to be put into the buffer) by setting the filters to only show specific data. You might also try setting checksums off (AT C0), spaces off (AT S0) and possibly error messages off (AT EM0) to reduce the amount of data being sent.

BUS BUSY

Messages provided to the ELM325 are sent after a certain time passes (based on the message priority), and while no other device is sending. If another device should begin sending at the same time as the ELM325, then both devices will stop, wait some time, and attempt a resend. Usually, this is sufficient to allow a message to be successfully sent, but should the ELM325 not be successful after a few seconds, it will stop trying and report that the bus is too busy. If this occurs, you will need to determine if there is a problem (eg. possibly in the wiring), and decide if you wish to try again.

<DATA ERROR

This message appears if the checksum calculation did not agree with the one that was sent from the vehicle. It means that there is a problem with one or more of the bytes received. There could have been a noise burst which interfered, or possibly there is a circuit problem. Try the command again – if it was a noise burst, it may be received correctly the second time.

NO DATA

The IC waited for the period of time (as set by the AT ST timer), and detected no response from the vehicle. It may be that the vehicle did not respond, or possibly that it did, but the filters were set so that the response was not seen. If you are certain that there should have been a response, try increasing the ST time (to be sure that you have allowed enough time for the ECU to respond).

<PROT ERROR

A protocol error occurs if there was a violation of the timing requirements before the message. This might be due to some noise or another problem, but the message is intended to only make you aware that something isn't quite right. If the checksums agree (no DATA ERROR), then the content of the message should be OK. If the problem persists, there may be an issue with the sender of the message which would require further investigation.

<RX ERROR

A receive error occurs if there was a problem detected with the low level J1708 data. That is, there was a problem with the actual length of the byte received and its component bits. This usually occurs if the baud rate is incorrect (which should not be the case if you are connected to a J1708 system, as they all use 9600 bps).

STOP?

If an operation is interrupted by a received RS232 character, the ELM325 will print STOP? then return to the prompt state. If you should see this response, then something that you have done has interrupted the ELM325.

?

This is the standard response for a misunderstood command received on the RS232 input. Usually it is due to a typing mistake, but it can also occur if you try to do something that is not appropriate for the command (for example, the AT GM command will give an error if a filter has not been set).

Outline Diagrams

The diagrams at the right show the two package styles that the ELM325 is available in.

The first shows our ELM325P product in what is commonly known as a 300 mil Plastic DIP (or PDIP) package. It is used for through hole applications.

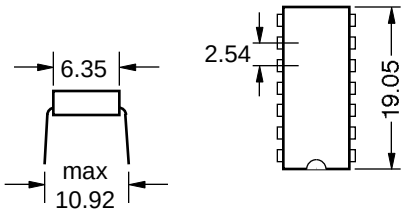
The ELM325SM package shown at right is our surface mount option. It is 3.90 mm wide (or 150 mils) and is known as a narrow Small Outline IC (or SOIC) package. We have chosen to simply refer to it as an SM (surface mount) package.

The drawings shown here provide the basic dimensions for these ICs only. Please refer to the following Microchip Technology Inc. documentation for more detailed information:

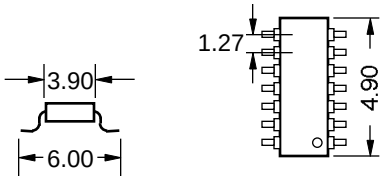
• The *Microchip Packaging Specification*, document name en012702.pdf (7.5MB). At the home page (www.microchip.com), select Design Support then Documentation then Packaging Specifications, or go to www.microchip.com/packaging

• The *PIC12(L)F1822/PIC16(L)F1823 Data Sheet*, file name 41413C.pdf (7 MB). At the www.microchip.com home page, use the Search Data Sheets box to look for 16F1823.

ELM325P



ELM325SM



Note: all dimensions shown are in mm.

Ordering Information

These integrated circuits are 14 pin devices, available in either a 300 mil wide plastic DIP format, or in a 150 mil (3.90 mm body) SOIC surface mount type of package. We do not currently offer any other package options for this device.

The ELM325 part numbers are as follows:

300 mil 14 pin Plastic DIP.....ELM325P 150 mil 14 pin SOIC.....ELM325SM

Index

Index (continued)