ELM325
J1708 Interpreter

Description

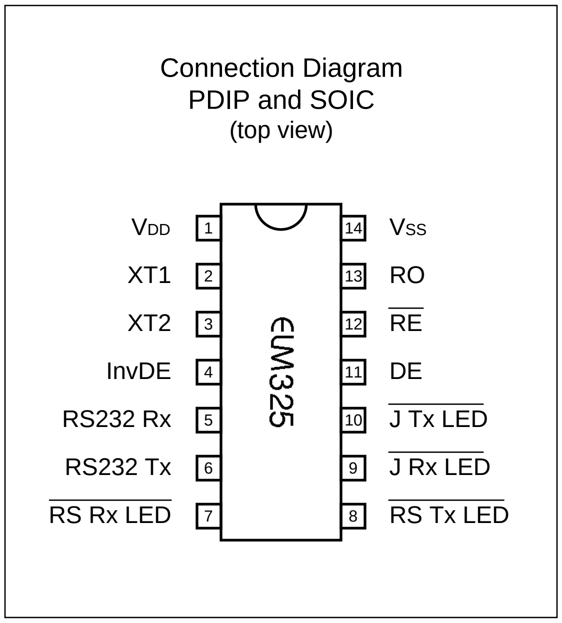
With the advent of electronic engine controls, many vehicles also adopted some form of diagnostic tools to help monitor their operation. As more modules began to be used in vehicles, there was also a need for the devices to share information rather than each independently obtaining it from separate sensors.

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 modules 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.

Features

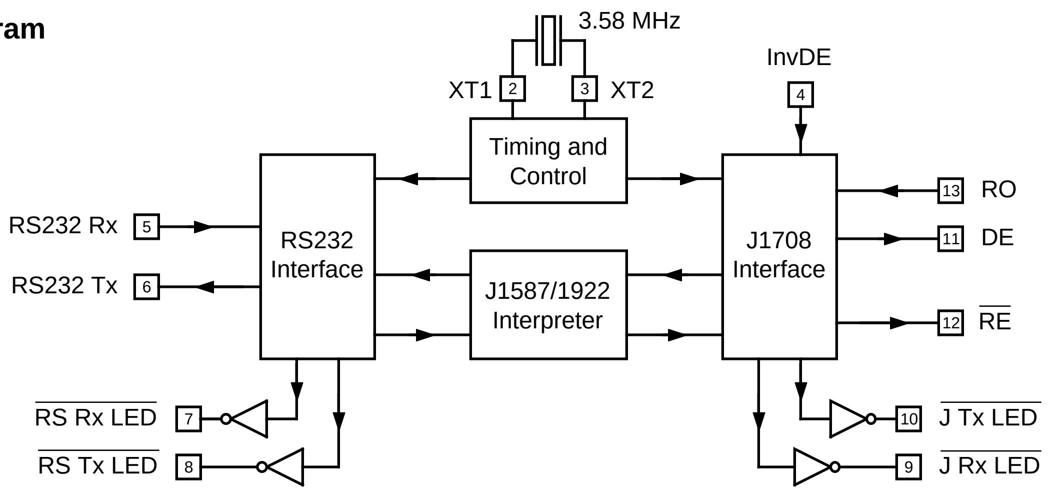
- Supports both SAE J1587 and J1922
- High speed RS232 interface
- Works with standard RS485 transceivers
- Fully configurable with AT commands
- Wide operating voltage range (1.8 to 5.5V)
- Low power CMOS design



Applications

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

Block Diagram



Contents

continued on the next page…

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 22 or 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.

The ELM325 expects a crystal or resonator to be connected between these two pins, and has not been configured for operation with an external oscillator. We do not recommend attempting to connect it to an external clock source.

InvDE (pin 4)

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

The SAE J1708 standard requires that all RS485 transceivers be turned off when transmitting a logic 1 (high level), and be on for a logic 0 (low level). In order to meet this requirement, the ELM325's data output (pin 11) is actually connected to the transceiver driver enable (DE) input, causing the transceiver to turn on and off with the output data. By connecting the transceiver's Data input to circuit common (logic low), the transceiver can be made to turn on and transmit a low data level, and turn off when a high needs to be sent.

Most RS485 transceivers use an active high level for their Driver Enable (DE) input. If the ELM325 InvDE pin is connected to a high level, it will work correctly with those transceivers. That is, the DE output (pin 11) will normally be at a low level during idle, and will go to a high level for the start (low) bit, enabling the transceiver. This connection is compatible with devices such as the SN75176, 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 ELM325's 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 many other types of interfaces as well, since the input has Schmitt trigger input wave shaping. The baud rate expected is 57600 bps, and is not adjustable.

RS232 Tx (pin 6)

This is the ELM325's serial data transmit output. The signal polarity 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. The baud rate transmitted is 57600 bps, and is not adjustable.

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

These four pins normally output a high level, and are driven low for a short period while 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 when on is not more than the supply level. If unused, these pins may be left open-circuited.

Note that the J Rx LED output only goes low for messages that meet the filter criteria, not for every J1708 message that the ELM325 sees.

DE (pin 11)

This is the J1708 transmit data output, which should be connected to the DE (Driver Enable) pin on an RS485 transceiver integrated circuit. The polarity of this signal may be changed by changing the level on the InvDE pin. If pin 4 is at a high level, the DE pin normally sits idle at a low level, and goes high when an active (low level) output is required on the RS485 bus.

Pin Descriptions (continued)

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

This is a general purpose output, which is typically used to enable or disable an RS485 receiver. For some circuits this might result in a substantial reduction in current, when the receiver is not being used.

This pin is manually controlled by software, with the AT RE0 and RE1 commands. It is always set to a low level on powerup, or an ELM325 reset, and does not change unless instructed to do so. If unused, this pin should be left open-circuited.

RO (pin 13)

This is the J1708 receive data input, which should be connected to the receive output (R or RO) pin of the RS485 transceiver IC. The ELM325 expects this signal to normally be at a high (Vdd) level when idle, and shift low (Vss) for start and '0' bits. This is a standard CMOS input that will accept TTL level signals.

Vss (pin 14)

Circuit common must be connected to this pin.

Unused Pins

The ELM325 is a CMOS integrated circuit that must have any unused inputs connected to either Vdd or VSS, or unpredictable behaviour can result (and possibly damage to the integrated circuit).

There are only 14 pins on an ELM325 package, and many of those pins must be connected in order for it to function. Pins 1 and 14 must be connected to provide power, and pins 2 and 3 must be connected to a crystal for circuit timing. Pins 5 and 6 are used for the serial control interface and both will typically be connected to an interface circuit (USB, RS232, Bluetooth, etc.).

If you do not wish to transmit data on the J1708 bus (ie only monitor data), then you do not have to connect pin 11 to anything, and may leave it open-circuited. The transceiver's Driver Enable pin will need to be set appropriately to turn off the driver, however, but you should check the data sheet to be sure that this does not affect the receiver. No matter whether pin 11 is used or not, pin 4 must be connected to either Vdd or Vss – which you connect to is your choice.

Pin 12 is an output that may be used if you wish, or left open-circuited if you do not require it. The RE pin on the transceiver will need to be connected such that the receiver is enabled though (typically low).

The final 4 pins are the LED outputs. These are provided so that you may see that the circuit is functioning, but are not required for operation of the circuit. If you decide not to connect a particular LED, then that pin may be left open-circuited.

Note that you may not short-circuit one of these LED output pins to another. Occasionally, people ask about connecting the RS or J LED pins together to provide an 'or' function. This may result in circuit damage. If you wish to drive one LED from two pins, consider adding some logic (two diodes provide an adequate 'or' for this purpose).

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. 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}	V_{DD} = 5.0V		1.2	2.0	mA	see note 3
	V_{DD} = 3.3V		1.0	1.7	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 v2.0)
	AT WS		0.8		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 a 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 correct ‘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 v2.0

>
```

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.

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 value. If you are not familiar with the hexadecimal numbering system, you may find the chart at right to be helpful.

Hexadecimal Number	Decimal Equivalent
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

Figure 1. Hex to Decimal Conversion

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 were 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 commands that are supported by the current version of the ELM325. Complete descriptions of each command are provided in the following section.

AT Command Summary

General

<CR>	repeat the last command
AMC	display the Activity Monitor Count
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

J1708 Control

C0, C1	Checksum display off*, or on
EM0, EM1	Error Messages off, or on*
M0, M1	MID display off, or on*
RE0, RE1	RE output pin low*, or high
RT0, RT1	Relaxed Timing off*, or on
S0, S1	Spaces off, or on*
ST hh	Set Timeout to hh
ST 00	Set Timeout to the default value

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

J1708 Data Specific

AF0, AF1	Auto Formatting 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 Messages
GO	Get One message
GO hh	Get One message with PID hh
GO 01hh	Get One message with PID 01hh
GP hh	Get messages with PID hh
GP 01hh	Get messages with PID 01hh
MA	Monitor All messages
MM hh	Monitor for MID hh messages
MP hh	Monitor for PID hh messages
MP 01hh	Monitor for PID 01hh messages
R0, R1	Responses off, or on*
SM hh	Set MID to hh
SP h	Set Priority to h
SP 0	Set Priority to the default value
TC	get Trouble Code messages

AT Command Descriptions

The following describes each AT Command that the current version of the ELM325 supports. Many of the 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.

AMC [Activity Monitor Count]

The ELM325 resets a counter every time it sees a J1708 byte on the data bus (whether you asked to see it or not). If it does not see a byte, the counter is allowed to increment (every 0.293 seconds) until it stops at the maximum of 255 or FF hex (about 75 seconds). Reading this value is a convenient way of determining if the data bus is active or not, and is done by sending AT AMC.

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 once.

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, etc. 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.

AT Command Descriptions (continued)

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.

F1 hh hh hh hh hh [set Filter 1 to…]

Filter F1 can be used to selectively receive only the messages that contain specified 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 is allowed to 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 specified 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 is allowed to 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 before the timer times out, they will all be displayed.

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.

GO hh [Get One message with PID hh]

There may be instances where you wish to only display information about one particular PID, and only need one response containing it. This command does this for you, returning to the command prompt immediately after getting and printing one response.

In addition to filtering the data for the PID, this command also respects filter settings, and times out automatically, based on the AT ST setting.

Note that the GO hh command only displays the bytes associated with requested PID hh, and discards bytes associated with any other PIDs in the message.

GO 01hh [Get One message with PID 01hh]

This command is exactly the same as the GO hh command, except that it filters for page 2 PIDs (that is, for PIDs of value 0100 to 01FF).

GP hh [Get messages with PID hh]

The AT GP command is used to look through messages, and only display the bytes associated with PID hh. Any other PID values in the message are discarded. If filter F1 or F2 have been set, then only messages meeting the filter requirements will be searched for the PID.

This command 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 before the timer times out, they will all be displayed.

GP 01hh [Get messages with PID 01hh]

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 looks 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).

I [Identify yourself]

Issuing this command causes the chip to identify itself, by printing the startup product ID string (currently ‘ELM325 v2.0’). Software can use this to determine exactly which integrated circuit it is talking to, without having to reset the IC.

AT Command Descriptions (continued)

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).

M0 and **M1** [MID display off or on*]

These commands determine whether the MID byte is displayed in the response or not. Usually, it is preferred to see this byte as it tells you which device is sending the data, but there may be occasions when you do not wish to see it. If so, you may block the printing of the MID by sending AT M0.

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.

There is no automatic timeout for this function – the search must be interrupted by the controlling PC sending a single character to the ELM325.

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.

There is no automatic timeout for this function – the search must be interrupted by the controlling PC sending a single character to the ELM325.

MP hh [Monitor for PID hh]

The AT MP command is used to look through messages, and only display the bytes associated with PID hh. Any other PID values in the message are discarded. If filter F1 or F2 have been set, then only messages meeting the filter requirements will be searched for the PID.

There is no automatic timeout for this function – the search must be interrupted by the controlling PC sending a single character to the ELM325.

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 R0), the ELM325 will not wait for a reply from the vehicle after sending a request – the ELM325 will 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 general purpose functions, such as disabling peripherals or blinking an LED.

RT0 and **RT1** [Relaxed Timing off* or on]

The ELM325 always measures the time between messages to ensure that the time is at least 12 bit periods long (as required by SAE J1708). If it should

AT Command Descriptions (continued)

see a violation of this timing requirement, it will print '<PROT ERROR' (for protocol error) after the offending line, but will otherwise carry on. If you are experimenting, and wish the ELM325 to relax this timing requirement a little, then turn on relaxed timing with AT RT1.

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.

Note that sending AT SP 0 causes the default priority to be set.

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 responses 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. For more information on the TC command, refer to the 'Getting Trouble Codes' section.

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. It is assumed that you have built (or purchased) a circuit that is similar to that shown in the Example Applications section.

Do not connect your circuit to a vehicle at this time. Power it from a test source only (if you do not have a bench supply, a 9V 'transistor radio' battery works well), and connect it to your PC as discussed previously 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 v2.0'.

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 format. Converting, 100 (= 6 x 16 + 4) is 64 in hexadecimal, so we send:

```
>AT ST 64
```

Again, don't forget to press return (or enter). 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.

The timeout should now 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 a command that should have been stopped by the timer has been prematurely 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. Another way of restoring the timer to the default setting is to send AT ST 00, which is handy if you only want to reset that setting.

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 Messages

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, which is very similar). Since J1922 is no longer actively supported, this data sheet will be mainly focusing on the J1587 standard.

SAE J1587 messages always begin with a single byte called the Message ID, or MID, as shown in Figure 2 below. It is essentially the sender's address (they are always predefined, and no two devices on the bus can have the same MID), so it identifies where the information is coming from.

The next byte contains a Parameter IDentification number, or PID. It tells you what type of information follows in the message. One or more data bytes follow the PID byte and are the actual data content of the message being sent. Several PID/data groups may 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 a total of 21 bytes or less in length).

If different amounts of data is required to be transmitted with each PID, then the PID data groups will need to vary in length within the message. Since there are no extra bytes to say when a PID begins and ends, then the lengths of the PID groups must be known before a message can be understood. It is the SAE J1587 standard that defines these data lengths for us.

The J1587 standard states that PIDs 0 to 127 (00 to 7F hex) will always have only one data byte, while PIDs 128 to 191 (80 to BF hex) will always have two data bytes. To accommodate other lengths, it allows PIDs 192 to 253 (C0 to FD hex) to have a variable number of data bytes, with the number of data bytes always provided after the PID byte.

PID 254 (ie hex FE) is a special PID called the data link escape PID. The data contained within that message is manufacturer specific, and not defined by J1587.

The standard allows an additional block of PIDs to be defined. These are numbered 256 to 511 (or 0100 to 01FF in hex), and are referred to as 'page 2 PIDs'. Naturally, if those are called page 2 PIDs, the ones in the 00 to 255 range are then called 'page 1 PIDs'.

To signal that the PIDs in a message are actually in page 2, PID 255 (hex FF) is put in the first PID position as shown in Figure 3. Rather than allow a mixture of page 1 and page 2 PIDs in one message, the standard states that a message may contain only one type – if all the PIDs in a message are page 1, then there will be no hex FF byte in the first PID position, and if all the PIDs in a message are page 2, there will be a hex FF byte in the first PID position, as shown in Figure 3. Since all page 2 PID numbers begin with 01 (hex), to repeat that byte for every PID in the message would be redundant, and it is assumed when you know the message is page 2. For example, PID 0123 should be in a message with the FF in the byte position after the MID, it will be encoded as PID 23 only (the 01 is dropped).

The data lengths for page 2 PIDs mirror that of the page 1 PIDs. That is PIDs from 0100 to 017F hex will always have one data byte only, while those from 0180 to 01BF will have two data bytes. PIDs from 01C0 to 01FD will have a variable number of bytes, just like

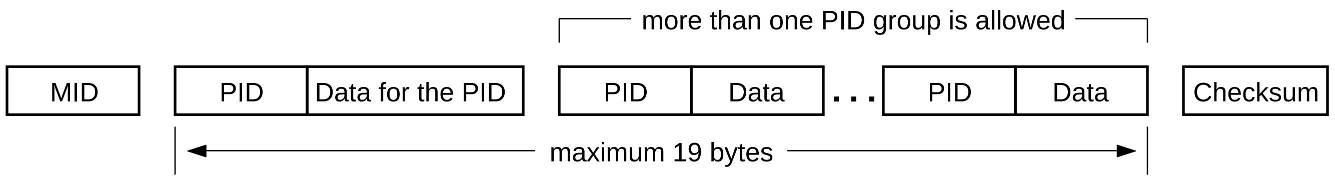


Figure 2. The SAE J1587 Message Structure

J1587 Messages (continued)

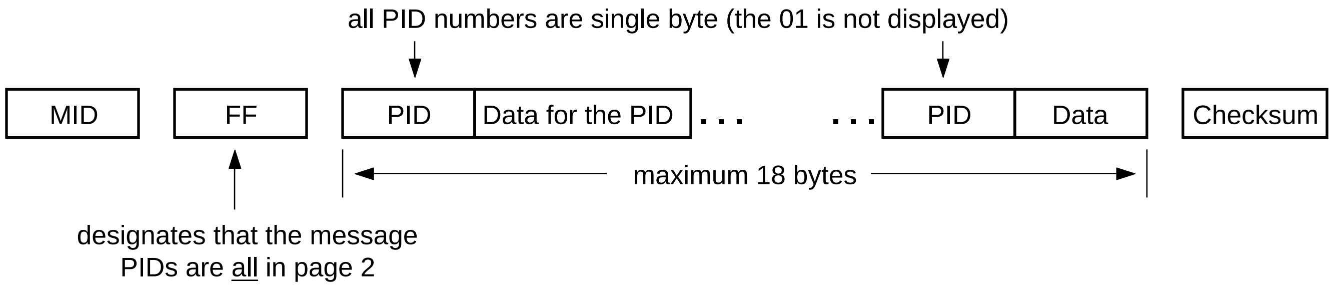

all PID numbers are single byte (the 01 is not displayed)
MID
FF
PID
Data for the PID
. . .
. . .
PID
Data
Checksum
maximum 18 bytes
designates that the message PIDs are all in page 2

Figure 3. Page 2 SAE J1587 Message Structure (PIDs 0100 to 01FE)

those from C0 to FD.

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

If you wish to learn more about the SAE J1587 standard, visit the www.sae.org web site.

About J1922

The SAE J1922 standard is another one that may be used by heavy duty trucks and buses, although it is being phased out. 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 any PIDs for defining the functions though – they do that with the MID byte. The J1708 standard defines MIDs 69 to 86 (ie. 45 to 56 hex) for use with J1922.

J1922 messages may be from 2 to 23 bytes in total length as shown in Figure 4 below. Note that the J1708 standard which this is based on actually allows up to 21 data bytes if the vehicle's engine is not running, and it's not moving. This is not an issue with the ELM325 as it is is capable of receiving an unlimited number of bytes, and is able to send as many as 21 data bytes (23 total bytes) if the automatic formatting is on.

If monitoring for J1922 messages, simply use the ELM325 commands as you would for J1587. Since J1922 messages do not contain PIDs, you can not really use the ELM325 instructions that specifically filter for PIDs, but you can use the others reliably – MA, MM, and if you set one or both filters GM and GO as well.

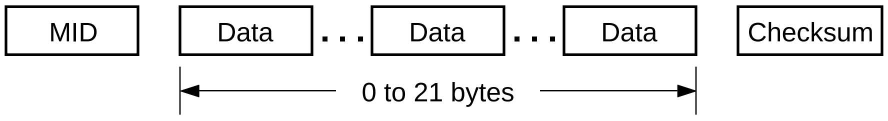


Figure 4. The SAE J1922 Message Structure

Making Requests

Using the ELM325 to send messages on the J1708/J1587 bus is very similar to sending OBD messages with our other products (such as the ELM327). Simply provide the data bytes that you want to be sent, in hexadecimal format, 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. The actual message that will be sent for you is:

```
AC 11 22 33 44 AA
```

The AC is the default MID for the scan tool, the next four bytes are the ones that you wished to have sent, and the checksum byte is AA. You can provide as many as 21 data bytes (42 digits) to be sent in this manner.

Note that the MID and the checksum are only added if automatic formatting is on. If you turn it off (with AT AF0), then only the four bytes that you provided will be sent, and the message would be received by another ELM325 as:

```
11 22 33 <DATA ERROR
```

The 44 is missing because the ELM325 does not normally show the checksum byte, and it always assumes that the last byte is the checksum. The ‘DATA ERROR’ message is printed because 44 is not the correct checksum for the other bytes, so something is wrong with the data received.

Most of the time, you will be sending requests for PID information, which follows a fairly simple format. PID 00 is used to say that you are making a request, and that is followed by the PID that you want. For example, to obtain the current battery voltage, you need to request PID 9E. Simply send:

```
>00 9E
```

and the ELM325 will add the MID and checksum, then send out the request for you.

If the ECU is not able to provide the information you want, you may see a response such as this:

```
NO DATA
```

and if it is able to reply, you might see a response like:

```
80 9E E0 01
```

The 80 byte tells you that engine #1 is providing the PID information, while the 9E says what PID is being sent, and since 80 ≤ 9E ≤ BF, the PID is two bytes long and the next two bytes are the data bytes for that PID.

All J1587 numeric data uses what is called little-endian byte order, meaning that numbers need to be read in reverse. The data provided is then 01E0 (and not E001). Converting hex 01E0 to decimal gives 480, and given that PID 9E (decimal 158) is encoded as 0.05V per bit, the voltage is 480 x 0.05 = 24.0V.

There may be times when you wish to limit the number of responses to a message sent (any message, not just a PID request). To do so, the ELM325 allows you to add a single digit after your data bytes, to tell the IC how many responses to (try to) obtain. This number can be from 0 to F. As always, the AT ST timer controls the maximum time allowed to wait for a reply, if not enough responses arrive. You will note that 0 is an acceptable number of responses as well, and acts as if you have temporarily selected AT R0.

As an example, for the battery voltage PID you might wish to obtain one reply and immediately return to the command prompt. You could do this with:

```
>00 9E 1
```

This feature is discussed further in the ‘Setting the Timeout’ section.

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 PID 9E, simply send:

```
>9E
```

and to obtain only 1 response for parameter 9E, send:

```
>9E 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 alternator AC output voltage (PID 374 or 0176 hex), simply send:

```
>0176
```

and if you wish to obtain three responses then return immediately to the prompt, send:

```
>0176 3
```

(Yes, three responses for alternator voltage is not very probable, but this is just an example of how to use the extra digit.)

Note that the automatic formatting will only change your data bytes to PID requests 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 taken care of for you by the receive filters, even if the auto formatting is off. The next sections discuss these filters.

Receive Filtering

The ELM325 has two special purpose receive filters built in to the J1708 receiver. These filters see every message, and pass only those that have byte patterns which match their setting. Figure 5 shows a diagram of how these filters are connected, and also shows that if neither filter is on, all messages will be passed on.

Each filter allows you to define the pattern desired for the first five bytes of any message received. You either tell it which value you require, or tell it ‘X’ meaning that you do not care what the value is. An example might be easier to see. Say you wish to see all messages from Engine #1 (MID 80), that have PID 5C (engine load) in the first PID position. Simply set F1 with the command:

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

This says that you wish to see 80 and 5C in the first two positions and don’t care what the other three bytes are. 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. The filter does not have to use whole byte values as shown above – should you wish for example to see all MIDs that start with 8, and all messages that have C as the second nibble of the PID, with FD in the fifth byte position, you could set:

```
>AT F1 8X XC XX XX FD
```

Since this filter will act on all received messages, 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 the Monitor All command:

```
>AT MA
```

To compound the effects of the receive filters with the AT commands, you might also try it with other commands such as AT MM or AT MP. In order for messages to be displayed, both the filter conditions and the MID/PID receive values must all be matched.

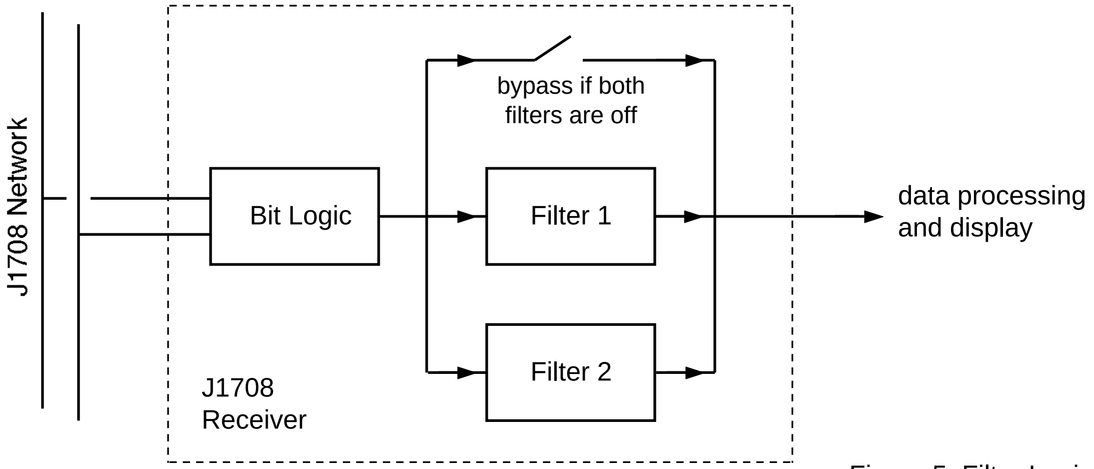


Figure 5. Filter Logic

As shown in Figure 5, the ELM325 also contains a second filter that you may define in exactly the same way as F1. If a second filter is defined, then all messages that meet either the F1 or the F2 criteria will be allowed to pass.

Either filter can be disabled by sending the AT command with no parameters. For example, to turn off filter 2, send only:

```
>AT F2
```

There is currently no way to quickly turn on a filter and restore its previous value -– you must redefine it in order to enable it.

You may wish to set filters at some point, depending on what you are experimenting with. You do not ever need to do so, however, as the ELM325 usually sets them for you, as described in the next section.

Automatic Receive Filters

Whenever you send a series of data bytes onto the J1708 bus, the ELM325 tries to determine what you are sending, and based on what it sees, it will set the F1 and F2 filters for you (the automatic formatting does not have to be on for this to happen).

The ELM325's settings may not be perfect for every situation, but they will be acceptable for most, and you always have the freedom to override them if you wish. By setting either or both of F1 and F2, you will stop the ELM325 from choosing the filter settings for you. The automatic receive filter settings will be restored as soon as you turn off your filter settings.

The following chart (Figure 6) shows what the ELM325's settings will be for different messages that you might send.

PID Request - Page 1	Filter 1	`XX PID XX XX XX`
`(MID 00 PID)`	Filter 2	`XX C0 XX PID XX`
PID Request - Page 2	Filter 1	`XX FF PID XX XX`
`(MID FF 00 PID)`	Filter 2	`XX FF C0 XX PID`
Component Specific - Page 1	Filter 1	`RxMID PID XX XX XX`
`(MID 80 PID RxMID)`	Filter 2	`RxMID C0 XX PID XX`
Component Specific - Page 2	Filter 1	`RxMID FF PID XX XX`
`(MID FF 80 PID RxMID)`	Filter 2	`RxMID FF C0 XX PID`
Diagnostic Data Request	Filter 1	`RxMID C4 XX XX XX`
`(MID C3 03 RxMID PID)`	Filter 2	`RxMID C0 XX C4 XX`
All other requests	Filter 1	`XX XX XX XX XX`
	Filter 2	(not used)

Figure 6. Automatic Receive Filter Settings When Sending

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:

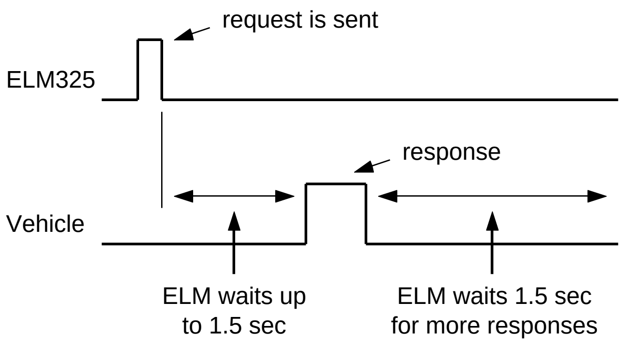


Figure 7. Message Response Timing

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, rather than sending say:

```
>9E
```

for the battery voltage, send:

```
>9E 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 to the prompt state immediately after printing the response.

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 'number of responses' 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, to obtain one response and return immediately.

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, especially if you provide the number of responses for it to get.

Listening to a Vehicle

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’ the data. For example, to monitor all data on the bus, send:

```
>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 monitor for a particular MID only with the Monitor MID command. You can monitor for all messages being sent by the engine (MID 80), by sending:

```
>AT MM 80
```

Every line returned will begin with an 80. If you should wish to only see the PID data (and not all of those 80’s), you can turn off the display of the MID using the AT M0 command.

Note that the ELM325 does not show the checksum byte 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:

$$\underbrace{\texttt{80}}_{\substack{\text{MID}\\ \text{(engine \#1)}}}\quad \overbrace{\texttt{5C}\quad\texttt{66}}^{\substack{\text{PID 5C}\\ \text{(\% load)}}}\quad \underbrace{\texttt{BE}\quad\texttt{E0}\quad\texttt{2E}}_{\substack{\text{PID BE}\\ \text{(rpm)}}}$$

The first byte (80) represents the engine #1 MID, while the second byte (5C) is the first Parameter ID (PID) byte. Since 5C is in the range from 00 to 1F, one data byte (the 66) follows the PID 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. As mentioned in ‘Making Requests’, J1587 numerical data is always little-endian, meaning that the least significant byte occurs first, and the data needs to be read backwards (so the value is 2EE0). Hexadecimal value 2EE0, is 12000 decimal, and with 1/4 rpm per digit, the engine 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 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.

Listening for Specific Data

If you tried the AT MA command in the previous section on a live vehicle, or with a simulator, you will have quickly found that there was a lot of data being continually sent by the ECU(s). In fact, most of the data of interest is broadcast continually, and easily available, so you rarely need to request data.

Consider the previous example. If you were interested in monitoring for the engine rpm PID (BE), you might be able to use one of the filters with the MA command. Recall, when we asked for only the engine messages, we saw:

```
>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.
```

So something like this might work for us:

```
>AT F1 80 XX XX BE XX
OK

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

Note that instead of using MM, we used MA, and put the filter for MID 80 in the F1 setting. Either way works.

The problem with this approach is that you will miss some messages, since the BE is not always in the same byte position. Version 1.0 of the ELM325 was limited to this method of gathering data, sometimes requiring that the user send AT MA with no filters set, and then parse all of the data on the host PC. It offered a little help with the AT MP command in that it set the filters for you, but it could not parse the message for a particular PID.

The new ELM325 now offers an improved version of the AT MP command, that can search for PID values for you. If presented with the previous data, the new AT MP command would give the following response:

```
>AT MP BE
80 BE E0 2E
80 BE 30 2F
80 BE 30 2F
etc.
```

You will note that the printed lines only show the PID data that you asked for. It is not obvious that this has been done at this point though. In order to see an indication that the lines are only partially represented, you would have to turn on the checksum display (with AT C1). Doing this with the same data would result in the following:

```
>AT MP BE
80 BE E0 2E - F2
80 BE 30 2F - A6
80 BE 30 2F - A3
etc.
```

The '-' character shown above before each checksum byte tells you that there is information missing. This is so that you do not assume that what is displayed by the ELM325 is all that there was in the message. If one line should have no information hidden (i.e. there were no other PIDs in the message), then there would be no '-' shown before that checksum byte.

The AT MP command does not filter for a particular MID. In this example, we have assumed that only the engine was transmitting the BE messages. For some PIDs, you might also want to filter for a particular MID as well, and to do so you would need to set one of the filters like this:

```
>AT F1 80 XX XX XX XX
```

You do have to enter all of those X's - the filter requires five bytes in total for the setting. A few people have asked that we make this variable so that 1 to 5 bytes can be provided, and it is on the list for a future firmware update.

Page 2 PIDs may be monitored in exactly the same way, except that the PID number that you provide is four digits long. For example, to monitor for PID 368 (i.e. 0170 hex), you would send AT MP 0170, and the replies would be the same format as above, except for one thing – the second byte is always FF (as it is a page 2 PID).

As with all monitoring commands, you must stop the AT MP monitoring by sending a single character to the ELM325. It does not matter what the character is, as it is discarded. The next commands discussed do not require a 'stop' character, as they use the AT ST timer to stop themselves.

Getting Specific Data

The ELM325 will either 'monitor' for information for you, or it will 'get' information. The difference between the two is basically that the monitoring commands (MA, MM, and MP) do not stop on their own, while the 'get' commands (GM, GO and GP) are stopped automatically by an internal timer.

This section discusses how to use each of the 'get' commands to display J1708/J1587 data.

The AT GM command is actually quite a handy command to use when you wish to watch for specific messages, setting the filters to detect the patterns. When you send a message (see 'Making Requests'), the ELM325 actually sets the filters per Figure 6 then performs a GM for you.

Just a warning as you experiment – while this command uses the AT ST timer to stop a search, you need to remember that it is reset every time a message is received and printed. If there are many messages being broadcast that pass the filters, then the timer may be reset so often that it does not time out. If that happens, you may interrupt the flow of messages by sending a single ascii character to the ELM325. That will interrupt the ELM325, causing it to stop and display

```
STOP?
```

This message is generated because it is not normal to stop a 'get' command with a key press, and the ELM325 is saying so. If you manually stop a 'monitor' command, no error message is printed because that is the normal way that those commands function.

There will be times when you wish to prevent such a large stream of data, and the need to manually stop it. In these cases, the AT GO (get one) command is most helpful. Instead of sending AT GM, simply send AT GO. Like the AT GM, the AT GO uses whatever filters you have defined in order to select the desired response.

The latest ELM325 introduced a couple of other 'get' commands that are most helpful when trying to find specific PID information. The first of these is the AT GO hh command. The 'hh' part is the PID number that you wish to search for. Using the example from the previous section, and sending AT GO BE, we might see:

```
>AT GO BE
80 BE E0 2E
```

and with the checksum on:

```
>AT GO BE
80 BE E0 2E - F2
```

The GO command may also be used with page 2 PIDs – simply put 01 before the PID number (for example AT GO 0170).

A similar command that instead retrieves multiple responses is AT GP (get PID). It is somewhat like both AT GM and AT GO in that it keeps retrieving messages until the timer stops it (like AT GM), but it also filters for a PID value (like AT GO hh).

To use the GP command, you only need to provide the PID number:

```
>AT GP BE
80 BE E0 2E
80 BE 30 2F
80 BE 30 2F
etc.
```

this display of data carries on until it is stopped by the timer. As with the GO command, if you wish to search for page 2 PIDs, simply add an 01.

The 'get' commands do not filter for a particular MID, as was the case with the monitoring commands. This means that if you wish to be sure that you are getting information from only one source, you need to set a filter for that MID. To only see messages with a MID of 80 then, you would need to set one of the filters like this:

```
>AT F1 80 XX XX XX XX
```

Since you now know that the MID is always 80, you might turn off the display of that value:

```
>AT M0
```

and then the previous GP result would look like:

```
>AT GP BE
BE E0 2E
BE 30 2F
BE 30 2F
etc.
```

That's really all there is to the 'get' commands. As you can see, they are not very complicated.

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). For your convenience, the ELM325 has a special function built in just for this purpose. To monitor for currently active trouble codes that are being broadcast, 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, and will simply perform a 'get' using them. Any responses printed will of course match your settings and will not necessarily represent trouble codes though, 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 does not do anything that you can not do, it just saves you a few steps. 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 found.

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 very easily as well – all you need to do is send a request for PID 194 (C2) like this:

```
>C2
```

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

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 or 'sections' for the receiver to assemble into one complete response.

Messages with multiple parts (or 'sections'), will usually use PID 192 (C0) to identify that the message is mult-part. There is another type of long message transmission that we are not going to discuss here, but if you wish to learn more about it, look up 'Connection Management' along with PIDs 197 and 198.

Figures 8a and 8b on the next page show how the C0 PID is used. Figure 8a shows that the first message section (section 0) is slightly different than all of the others. It has one extra byte ('Bytes') that shows how many bytes there are in the complete response. All of the next sections 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 16 sections in one message (0 to F), so the maximum number of data bytes that can be sent with this scheme is 14 + 15 x 15 = 239. If a vehicle needs to send more than that at one time, then it needs to use the other method mentioned previously.

If you recall Figure 6 in the 'Automatic Receive Filters' section, the ELM325 sets both F1 and F2 for most messages. For the F1 filters, the requested PID number appears in the very first PID position, which is the standard reply that you would expect for a message with 1 to 18 data bytes. The F2 setting is a little different, however, with the first data byte set to C0. From Figures 8a and 8b, you can now see that F2 is looking for multisection responses.

When you receive a multisection response, it will need to be reassembled into a single message. First, make sure that you have all of the sections – the number of the last one is provided by the most significant nibble of the 'Sections' byte, while the section number for each message is provided by the

Multiline Responses (continued)

least significant nibble. Be sure to pay attention to the MID values too as more than one MID might possibly respond.

Note that each section (or segment) of a message is sent by the MID device in sequence, but since there is no hand-shaking mechanism, individual sections can not be resent. If there should be a problem with one section, you must request the PID again, and can not have a single section resent.

Also, while numeric data is sent least significant byte first, alphanumeric data (letters) are sent with the most significant character first.

There aren't that many multiline responses that you will encounter – perhaps the most common would be the VIN (PID 237 / hex ED) or a component ID (PID 243 / hex F3), but now you know how to interpret them when you do encounter one.

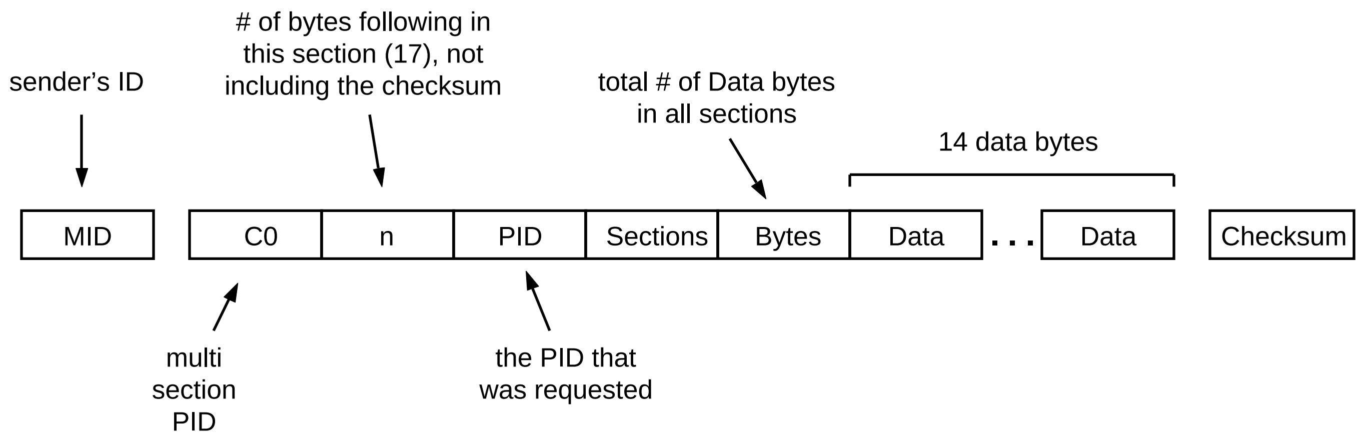

sender's ID
# of bytes following in this section (17), not including the checksum
total # of Data bytes in all sections
14 data bytes
MID
C0
n
PID
Sections
Bytes
Data
. . .
Data
Checksum
multi section PID
the PID that was requested

Figure 8a. Multisection Response - First Section

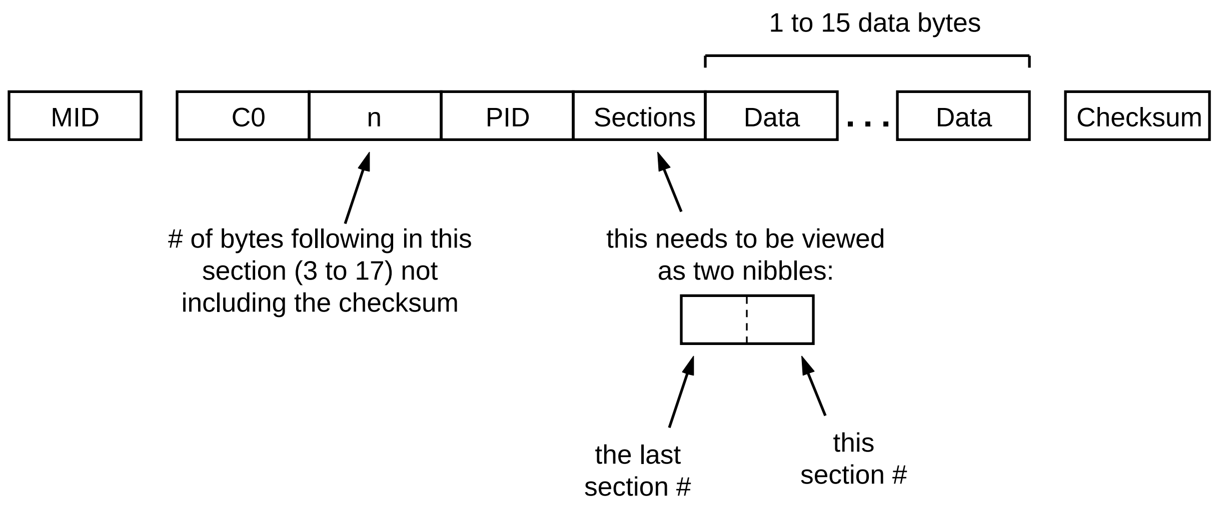

1 to 15 data bytes
MID
C0
n
PID
Sections
Data
. . .
Data
Checksum
# of bytes following in this section (3 to 17) not including the checksum
this needs to be viewed as two nibbles:
the last section #
this section #

Figure 8b. Multisection Response - All Other Sections

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 for it.

The ELM325 uses the 'Off-board Diagnostics #1' device MID by default (hex value AC, or decimal value 172). If you wish to send messages using another MID, it is easily changed 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 support any Programmable Parameters (like the ELM327 or the ELM329), so a preferred value (other than AC) can not be stored in EEPROM.

Sending Any Message

The ELM325 allows you to send any message that you want, as it does not limit what you send in any way, with one exception. That exception was noted before - that the automatic formatting will change your data bytes if you send a single byte or send two bytes where the first one is 01, as it thinks that you wish to request PID data. A few examples will help with this.

Assume that you wish to send the data bytes 12 34 56 78 in a message. To do this, simply send:

```
>12 34 56 78
```

and what will actually be sent is:

```
AC 12 34 56 78 40
```

as the MID and checksum are automatically added for you when automatic formatting is on (it is by default).

What if you had wanted to send this from MID 99 though? Then you would need to set the MID to 99, using the AT SM 99 command, before you send the data. The message sent in that case would be:

```
99 12 34 56 78 53
```

OK, so what if you should want to send 01 F0, from MID C9? Setting the MID is easy enough, but sending data byes 01 F0 would make the ELM325 think that you wish to send a page 2 PID request. In this case, you need to turn off formatting and do the work yourself. To turn off auto formatting, send:

```
>AT AF0
```

Now, you need to calculate your own checksum. To do this, you need to sum up all of the values in the message, using a hex calculator:

```
C9 + 01 + F0 = 1BA
```

The checksum is the two's complement of the sum, 'mod 256'. This can be done in several ways, one of which is to only use the last two digits ('BA'), and subtract them from hex 100:

```
Checksum = 100 - BA = 46
```

Sending your message now only requires providing the bytes at the prompt:

```
>C9 01 F0 46
```

With a little practice, it is not too difficult to send any message with the ELM325. You only need to be aware of the single byte/two byte with 01 restriction.

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, or a single character from the PC. 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 you need to be aware of this.

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 ST timer, and experimented with the filters, or maybe tried to see what happens if the MID byte is changed. How do you reset these changes?

To reset only the filters to their initial state, simply send:

```
>AT F1
```

and:

```
>AT F2
```

to have the ELM325 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 (for echo, linefeeds, etc), so should only be used when you truly want all of 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 in exactly the same way as a power off/on is the full reset command:

```
>AT Z
```

It takes approximately one second for the IC to perform a full 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.

The Activity Monitor

The ELM325 continually monitors for data on the J1708 input, whether you are seeing it displayed or not. Every time that it sees a new byte on the J1708 data bus, it resets an internal counter called the Activity Monitor Count.

The Activity Monitor Count is automatically incremented by the ELM325 roughly every 0.3 sec, so it actually always measures the time since the last J1708 byte was received. Software does not let the counter go beyond FF (decimal 255), so the maximum time measured is about 75 seconds.

You may read the value in the Activity Monitor counter at any time by sending the AMC command:

```
>AT AMC
```

and use the result as required. Typically you might set a threshold value, say CD which is 60 seconds, and make a decision if the AMC is greater than this. Note that the current version of the ELM325 does not offer a sleep mode – it always remains active.

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 9 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.

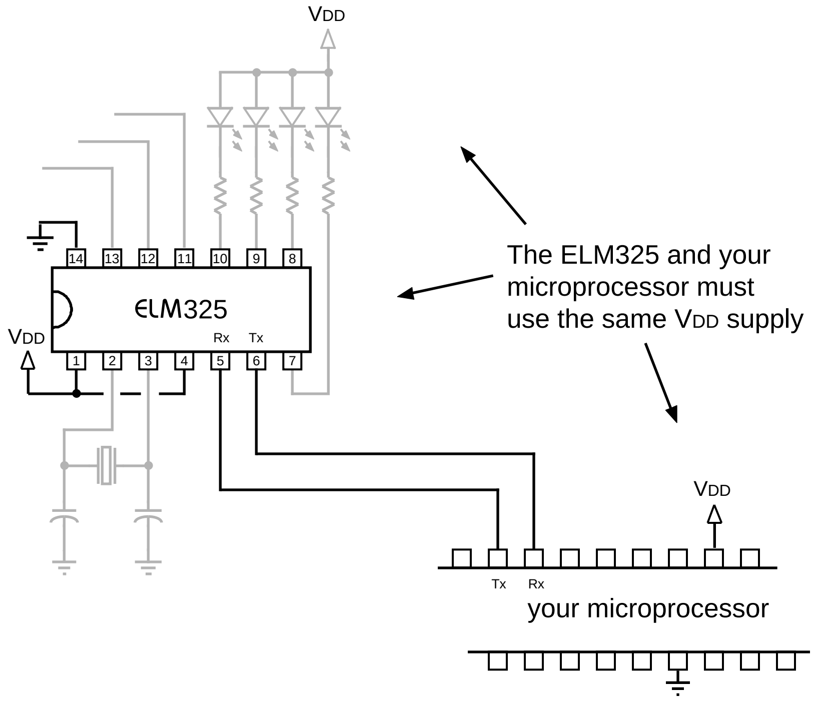

VDD
14
13
12
11
10
9
8
ELM325
VDD
Rx
Tx
1
2
3
4
5
6
7
The ELM325 and your microprocessor must use the same VDD supply
VDD
Tx
Rx
your microprocessor

Figure 9. 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 10.

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 verify 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 10 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 10 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)

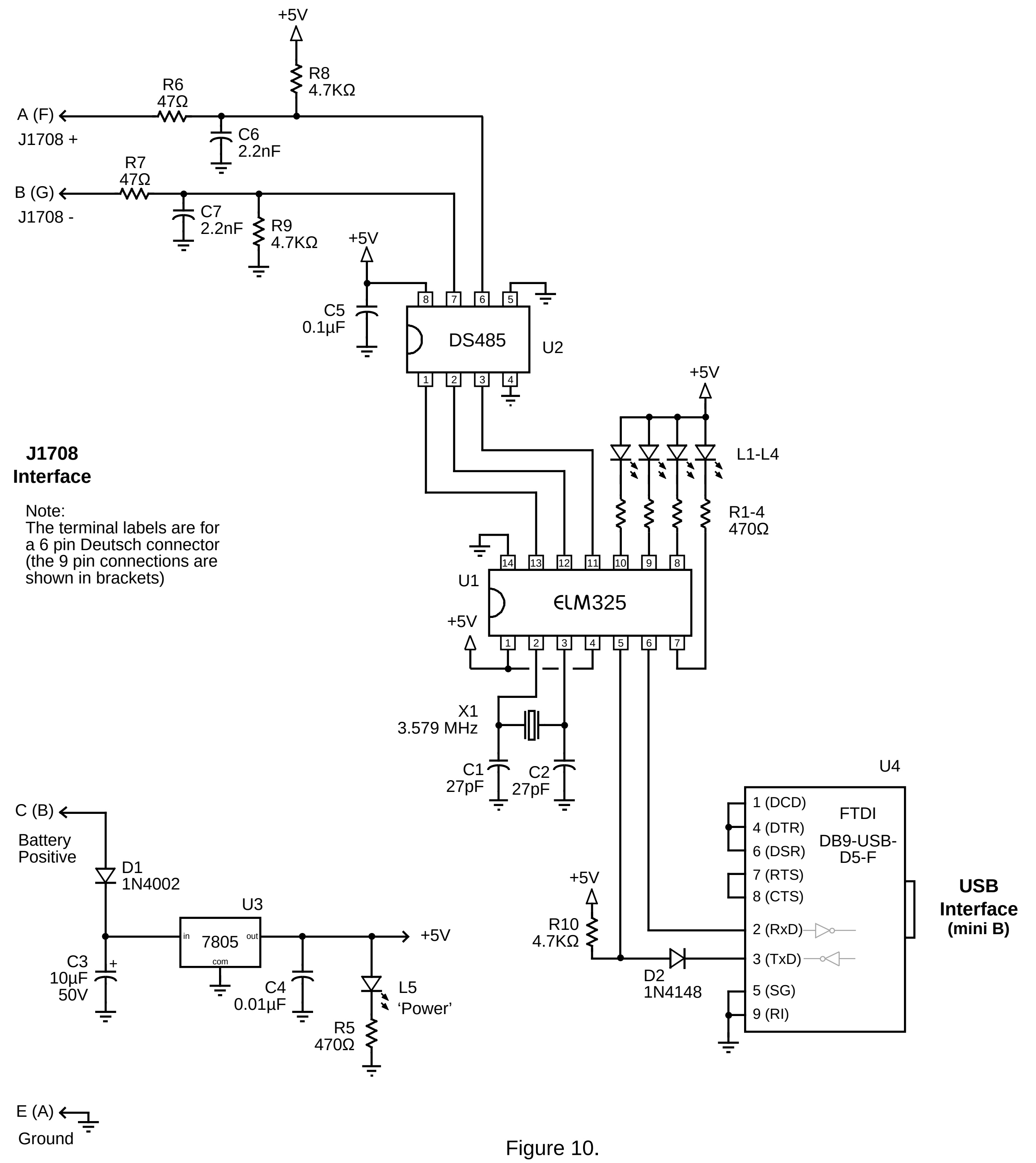


Figure 10.
A J1708 to USB Interpreter

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 Terminal by Bray), 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 12. It is identical to that in Figure 10, 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 = 1N4002
D2 = 1N4148
L1, L2, L3, L4 = Yellow LED
L5 = Green LED
U1 = ELM325
U2 = DS485 (RS485 transceiver)
U3 = 7805 (5V, 1A regulator)
U4 = FTDI DB9-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 11. Parts List for Figure 10

Example Applications (continued)

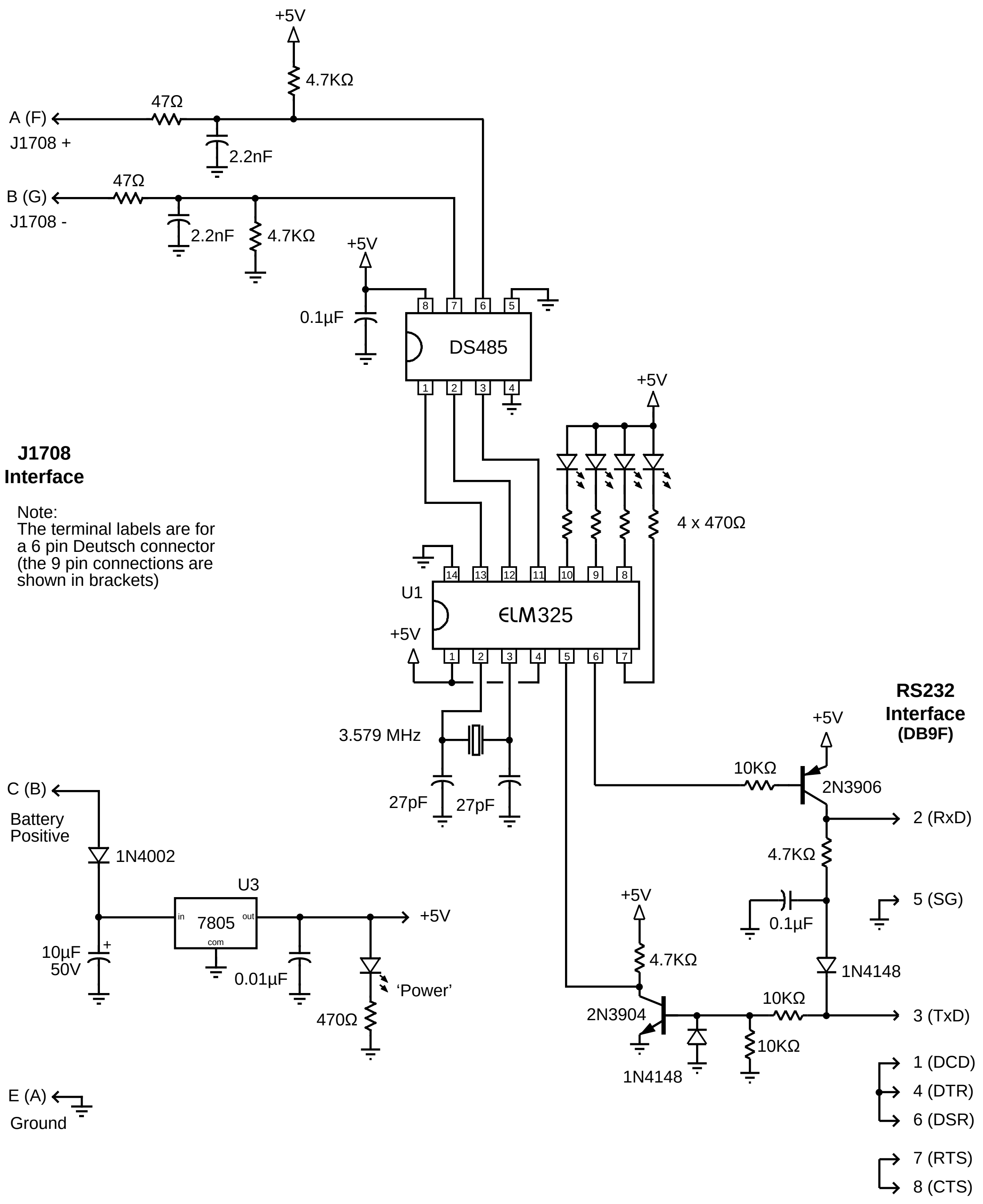


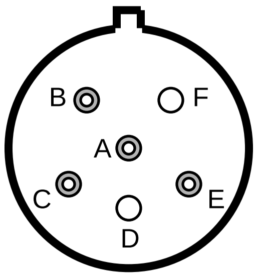
Figure 12. An RS232 Serial Interface

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.

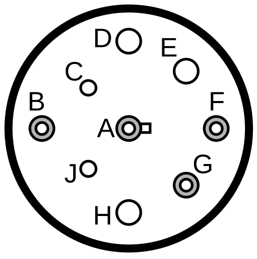
The following shows the pin configurations for the 'Deutsch' connectors, and lists part numbers for each. Should you wish to make your own connectors, you will need to buy a shell (or casing) as well as the metal pins to insert into it.

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 at the mating (non-cable) side

	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 AT62-210-1231-1820 (AT62-210-1231-1820-ND)



9 pin 'Deutsch' test connector, looking at the mating (non-cable) side

	Deutsch Industrial	Amphenol Sine Systems
Shell	HD16-9-1939S	AHD16-9-1939S (889-1059-ND)
# 16 Socket Pins	1062-12-0122	AT62-16-0622 (889-1467-2-ND) or AT62-16-0122 (889-1049-1-ND)

ECU Simulators

We are occasionally asked about bench testing circuits and this generally turns to recommendations for ECU simulator devices. We do not offer any recommendations, or reviews of any type concerning particular products, but we do occasionally pass on links for devices that our customers might consider.

There are a few lower cost simulators available for J1708/J1587 networks that we are aware of, and list here. The manufacturers will be the best source for information:

Au Group Electronics
(www.AuElectronics.com)

SAE J1708/J1587 Simulator
www.auelectronics.com/System-J1708Simulator.htm

Dafulai Electronics
(www.dafulaielectronics.com)

OBD2/J1708/J1587 Simulator DFLSOBD2
www.dafulaielectronics.com/Pages/OurProducts.aspx

Ozen Elektronik
(www.ozenelektronik.com)

J1708-J1587 ECU Simulator mOByDic1500
www.ozenelektronik.com/j1708-j1587-obd-ecu-simulator-p.html

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 two 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 value that was sent by 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).

Error Messages and Alerts (continued)

<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. As a temporary work-around, you might try using the AT RT1 command (see page 12).

<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.

Version History

The following summarizes some of the differences between the ELM325 firmware versions.

v1.0

First version available. Support for J1708/J1587 (and J1922) communications.

Features
- automatically adds MID and Checksum to data bytes (up to 21) for you and sends
- single nibble sets number of responses to get
- user settable priority
- resends automatically on collision detection
- two filters can select data streams
- 32 byte RS232 transmit buffer

AT Commands:
<CR>, AF0/1, C0/1, D, E0/1, EM0/1, F1, F2, GM, GO, I, L0/1, MA, MM, MP, R0/1, RE0/1, S0/1, SM, SP, ST, TC, WS, Z

v2.0

New Features:
- new byte handling code searches for PIDs
- filters now use the checksum byte if it is visible (AT C1)
- new activity monitor measures the time since the last J1708 byte received
- AT L0/L1 code completely revised
- several improvements to other internal code

New AT Commands:
AMC, GP hh, GP 01hh, GO hh, GO 01hh, M0/1, SP 0, RT0/1,

Upgrading Versions

A popular question that we receive for all of our products is “Can I upgrade my firmware with a file download?”. The answer to this is no, the ELM325 can not be upgraded in this way - your integrated circuit must be replaced.

Replacing an ELM325P integrated circuit is not that difficult, if it is in a socket. Be sure to note the orientation of the chip before removing the old one. Integrated circuits that have been soldered in place require more skill to change and should be left to those with experience in doing so.

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.

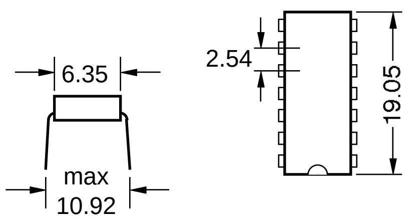
The ELM325SM package shown at right is our surface mount option. It is 3.90 mm wide (or 150 mils) and is known as a narrow Small Outline IC (or SOIC) package. We have chosen to simply refer to it as an SM (surface mount) package.

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:

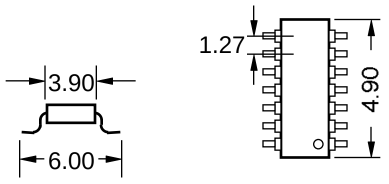
• *Microchip Packaging Specification documents.*
At the home page (www.microchip.com), expand the Design Support tab, then choose Documentation and Packaging Specifications. Choose PDIP or SOIC.

• *PIC12(L)F1822/PIC16(L)F1823 Data Sheet*,
At the home page (www.microchip.com), expand the Design Support tab, then choose Documentation and Data Sheets. Search Products 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)