ELM323

OBD (ISO) to RS232 Interpreter

Description

Since the 1996 model year, North American automobiles have been required to provide an OBD, or On Board Diagnostics, port for the connection of test equipment. Data is transferred serially between the vehicle and the external equipment using these connections, in a manner specified by the Society of Automotive Engineers (SAE). These are serial connections, but as the voltage levels and the protocols differ, computer serial ports cannot be directly used to communicate with vehicles.

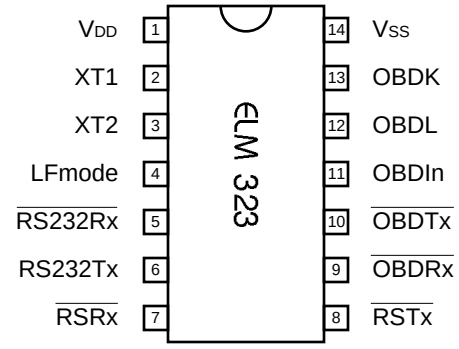
The ELM323 is a 14 pin integrated circuit that, with only a few external components, is able to interface to the OBD port, interpreting the data signals and reformatting them as standard ASCII characters. This allows virtually any personal computer or PDA to communicate with a vehicle using only a standard serial port and a terminal program. If desired, hobbyists can create their own custom 'scan tool' by adding an interface program.

This integrated circuit was designed to provide a cost-effective way for experimenters to work with an OBD system, so many features such as RS232 handshaking, variable baud rates, etc., have not been implemented. Additionally, this device is only able to communicate using the 10.4KHz ISO 9141 protocol that is commonly used by Daimler Chrysler, and many 'world built' vehicles.

Features

- Low power CMOS design
- Crystal controlled for accuracy
- Configurable with AT commands
- Standard ASCII character output
- Four high current LED drive outputs
- 10.4KHz ISO 9141 Protocol

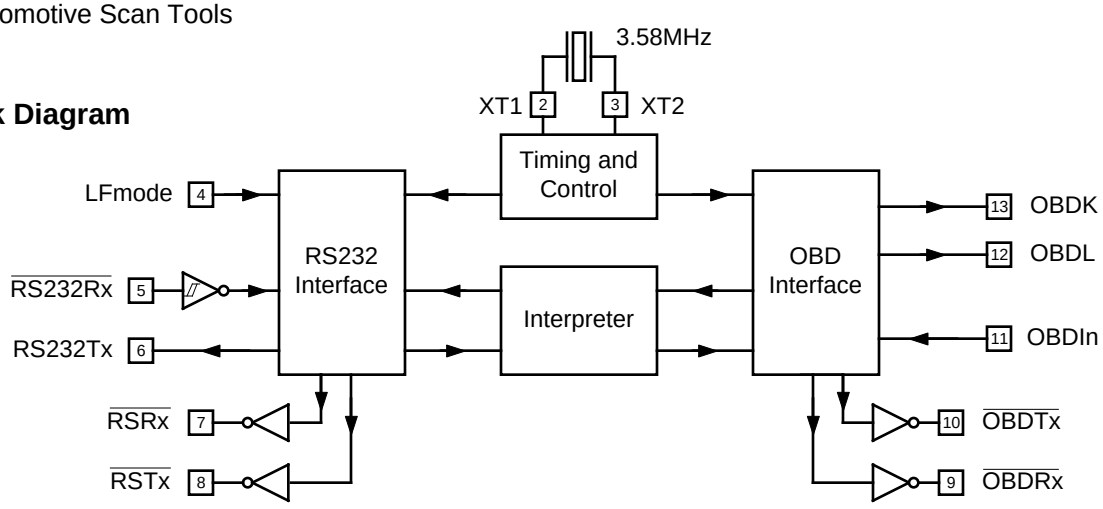
Connection Diagram
PDIP and SOIC
(top view)



Applications

- Diagnostic Trouble Code Readers
- Automotive Scan Tools

Block Diagram



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 3.579545MHz NTSC television colourburst crystal is connected between these two pins. Crystal loading capacitors (typically 27pF) will also normally be connected between each of the pins and the circuit common (Vss).

LFmode (pin 4)

This input is used to select the default linefeed mode after a powerup or system reset. If it is at a high level, then by default lines sent by the ELM323 will be terminated with both a carriage return and a linefeed character. If it is at a low level, lines will be terminated by a carriage return only. This behavior can always be modified by issuing ATL0 or ATL1 commands (see the section on AT Commands).

$\overline{\text{RS232Rx}}$ (pin5)

A computer's RS232 transmit signal can be directly connected to this pin from the RS232 line as long as a current limiting resistor (typically about 47KΩ) is installed in series. (Internal protection diodes will pass the input currents safely to the supply connections, protecting the ELM323.) Internal signal inversion and Schmitt trigger waveshaping provide the necessary signal conditioning.

RS232Tx (pin 6)

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

LED Drive Outputs (pins 7, 8, 9, and 10)

These four pins are driven to low levels when the ELM323 is transmitting or receiving RS232 or OBD data. Otherwise, they are at a high level. Current capability is suitable for directly driving most LEDs through current limiting resistors. If unused, these pins should be left open-circuited.

OBDIn (pin11)

The OBD data is input to this pin, with a high logic level representing the active state of the OBD K line. No Schmitt trigger input is provided, so the OBD signal should be buffered to minimize transition times for the internal CMOS circuitry.

OBDL (pin 12) and OBDK (pin 13)

These are the active high output signals which are used to drive the OBD bus, using external NPN transistors. Data transfer normally occurs only by the K line, but the standards require that the L line be implemented as well in order to properly initialize the bus. See the Example Application section for more details.

V_{SS} (pin 14)

Circuit common is connected to this pin. This is the most negative point in the circuit.

Ordering Information

These integrated circuits are available in either the 300 mil plastic DIP format, or in the 150 mil SOIC surface mount type of package. To order, add the appropriate suffix to the part number:

300 mil Plastic DIP.....ELM323P

150 mil SOIC..... ELM323SM

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 to +7.0V

Voltage on any other pin with
respect to V_{SS}..... -0.6V to (V_{DD} + 0.6V)

Note:

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

Electrical Characteristics

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

Characteristic	Minimum	Typical	Maximum	Units	Conditions
Supply voltage, V_{DD}	4.5	5.0	5.5	V	
V_{DD} rate of rise	0.05			V/ms	see note 2
Average supply current, I_{DD}		1.0	2.4	mA	see note 3
Input low voltage	V_{SS}		0.15 x V_{DD}	V	
Input high voltage	0.85 x V_{DD}		V_{DD}	V	
Output low voltage			0.6	V	Current (sink) = 8.7mA
Output high voltage	V_{DD} - 0.7			V	Current (source) = 5.4mA
$\overline{\text{RS232Rx}}$ pin input current	-0.5		+0.5	mA	see note 4
RS232 baud rate		9600		baud	see note 5

Notes:

1. This integrated circuit is produced with a Microchip Technology Inc.'s PIC16C505 as the core embedded microcontroller. For further device specifications, and possibly clarification of those given, please refer to the appropriate 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. Device only. Does not include any load currents.
4. This specification represents the current flowing through the protection diodes when applying large voltages to the $\overline{\text{RS232Rx}}$ input (pin 5) through a current limiting resistance. Currents quoted are the maximum that should be allowed to flow continuously.
5. Nominal data transfer rate when the recommended 3.58 MHz crystal is used as the frequency reference. Data is transferred to and from the ELM323 with 8 data bits, no parity, and 1 stop bit (8 N 1).

Communicating with the ELM323

The ELM323 relies on a standard RS232 type serial connection to communicate with the user. The data rate is fixed at 9600 baud, with 8 data bits, no parity bit, 1 stop bit, and no handshaking (often referred to as 9600 8N1). All responses from the IC are terminated with a single carriage return character and, optionally, a line feed character. Make sure your software is configured properly for the mode you have chosen.

Properly connected and powered, the ELM323 will energize the four LED outputs in sequence (as a 'lamp test') and will then send the message:

ELM323 v1.1

>

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 displayed above is the ELM323's prompt character. It indicates that the device is in its idle state, ready to receive characters on the RS232 port. Messages sent from the computer can either be intended for the ELM323's internal use, or for reformatting and passing on to the OBD bus.

The ELM323 can quickly determine where the received characters are to be directed by analyzing the entire string once the complete message has been received. Commands for the ELM323's internal use will always begin with the characters 'AT' (as is common with modems), while commands for the OBD bus are only allowed to contain the ASCII codes for hexadecimal digits (0 to 9 and A to F).

Whether an 'AT' type internal command or a hex string for the OBD bus, all messages to the ELM323 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 10 seconds, and the ELM323 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 ELM323 (syntax errors) will always be signalled by a single 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 ELM323 is a protocol interpreter that makes no attempt to assess the OBD messages for validity – it only ensures that an even number of hex digits were received, combined into bytes, and sent out the OBD port, and it does not know if the message sent to the vehicle is in error.

Incomplete or misunderstood messages can also occur if the controlling computer attempts to write to the ELM323 before it is ready to accept the next command (as there are no handshaking signals to control the data flow). To avoid a data overrun, users should always wait for the prompt character ('>') before issuing the next command.

Finally, there are a few convenience items to note. The ELM323 is not case-sensitive, so 'ATZ' is equivalent to 'atz', and to 'AtZ'. Also, it ignores space characters and all control characters (tab, linefeed, etc.) in the input, so they can be inserted anywhere to improve readability. Another feature is that sending only a single carriage return character will always repeat the last command (making it easier to request updates on dynamic data such as engine rpm).

AT Commands

The ELM323 accepts internal configuration commands in much the same manner that modems do. Any message received at any time, that begins with the character 'A' followed by the character 'T' will be considered an internal configuration or 'AT' command. These are executed upon receipt of the terminating carriage return character, and completion of the command is acknowledged by the printing of the characters 'OK'.

Communications on the OBD bus can generally begin without requiring the use of any AT commands, as the factory default settings should be appropriate for most applications. Occasionally the user may wish to customize settings though, such as turning the character echo off. In these cases, AT commands must be issued.

To perform the desired function simply send AT followed by the appropriate characters. For example,

AT Commands (continued)

to turn character echoing off, simply send ATE0 followed by a return character. To turn it back on, send ATE1.

The following is a summary of the commands that are recognized by the current version of the ELM323. Remember that they are not case sensitive, they can have spaces or tab characters embedded as you wish, and that for the on/off type commands, the '0' character is the number zero:

D [set all to Defaults]

This command is used to reset the E, H and L options to their default (or factory) settings. This is equivalent to issuing an ATE1, an ATH0, and also an ATL1 or ATL0. (The level at LFmode is read whenever this command is issued, and L is set appropriately).

E0 and **E1** [Echo off(0) or on(1)]

These commands control whether or not characters received on the RS232 port are retransmitted (or echoed) back to the host computer. To reduce traffic on the RS232 bus, users may wish to turn echoing off by issuing ATE0. Default is E1 (echo on).

FI [perform a Fast Initiation]

When this command is received, the ELM323 will begin a Fast Initiation sequence as described by the ISO14230 standards. Normally this would be attempted if the slow 5 baud sequence fails, but this provides a means to manually initiate the sequence.

H0 and **H1** [Headers off(0) or on(1)]

These strings control whether or not the header information is shown in the responses. All OBD messages have an initial (header) string of three bytes and a trailing check digit that is normally not displayed by the ELM323. To see this extra information, users can turn the headers on by issuing an ATH1. The default is H0 (headers off).

I [Identify yourself]

Issuing this command causes the chip to identify itself, by printing the startup product ID string (this is currently 'ELM 323 v1.1'). Software can use this to determine exactly which integrated circuit it is talking to, without resorting to resetting the entire IC.

L0 and **L1** [Linefeeds off(0) or on(1)]

The transmitting of a linefeed character following every carriage return character is controlled by this option. If an ATL1 is issued, linefeed generation will be turned on, and for ATL0, it will be off. Users may wish to have this option on if using a terminal program, but off if using a custom interface (as the extra characters transmitted will only serve to slow the vehicle polling down). The default setting is determined by the logic level at the LFmode pin on powerup or reset: if it is a '1' or high level, then the default is L1, otherwise it is L0.

Z [reset all]

This combination 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 in the idle state, waiting for characters to arrive on the RS232 bus.

Bus Initiation

Both the ISO 9141 and ISO 14230 (KWP2000) standards require that the vehicle's bus be initialized before any communications can take place. ISO 9141 allows for only a slow (2 to 3 second) process while ISO 14230 allows for the slow method, as well as a faster alternative. In either case, once the bus has been initiated, communications must take place at least once every five seconds, to keep the bus 'alive'.

The ELM323 takes care of this bus initiation and the periodic sending of messages for you – it is automatic and requires no input from the user. The ELM323 will not initiate the bus until the first message needs to be sent however, by first attempting the slow method, and if that fails then trying the fast. During the

Bus Initiation (continued)

initiation process, the following message will be displayed:

BUS INIT: ...

with the three dots appearing as the process is carried out. This will be followed by either the expression 'OK' to say it was successful, or else an error message to indicate a problem. (The most common error is in forgetting to turn the vehicle's key to 'ON' before attempting to talk to the vehicle.)

Once initiated, the ELM323 does what is required to keep the bus alive, without any intervention from the user. If you have installed LEDs, you will see 'dummy' messages being sent every few seconds in order to create bus activity.

Note that the ELM323 does include some code for ISO 14230 initiation and data transfer, but at this time we cannot guarantee compliance until it is tested further. Hopefully it will work correctly when required, but we'd like to hear of any problems (and successes) that you do encounter.

OBD Commands

If the bytes received on the RS232 bus do not begin with the letters 'A' and 'T', they are assumed to be commands for the vehicle's OBD bus. The bytes will be tested to ensure that they are valid pairs of hexadecimal digits, and will be combined into bytes for transmitting. Recall that no checks are made as to the validity of the OBD command – data is simply retransmitted as received.

OBD commands are actually sent to the vehicle embedded in a data packet. The standards require that three header bytes and an error checksum be included with every message, and the ELM323 inserts them automatically for the user (they do not change in value, so are stored internally). To view the extra bytes that are received with the vehicle's messages, issue an ATH1 internal command, turning the header printing on. Occasionally vehicles will have more than one module responding to a request, and it may be useful in these cases to turn the headers on in order to determine which one responded (the third byte of the response is the address of the sender).

Most OBD commands to the vehicle are one or two bytes in length, but some can be three or more bytes long. The ELM323 is capable of sending seven data bytes (14 hexidecimal digits), the maximum number allowed by the standards. Attempts to send either an odd number of digits or too many digits will result in a syntax error – the entire command is then ignored and a single question mark printed.

Hexadecimal digits are used for all of the data exchange with the ELM323 because it is the data format used in the relevant SAE standards. It is consistent with mode request listings and is the most frequently used format used to display results. With a little practice, it should not be very difficult to deal in hex numbers, but some people may want to obtain a conversion table or keep a calculator nearby. All users will be required to manipulate the results in some way though – combining bytes and dividing by 4 to obtain rpm, dividing by 2 to obtain degrees of advance, etc. and may find a software front-end helpful.

As an example of sending a command to the vehicle, assume that A6 (or decimal 166) is the command that is required to be sent. In this case, the user would type the letter A, then the number 6, then would press the return key. These three characters would be sent to the ELM323 on the RS232 bus. The ELM323 would store the characters as they are received, and when the third character (the carriage return) is received, begin to assess the other two. It would see that they are both valid hex digits, and would convert them to a one byte value (decimal value is 166). Four header bytes would be added, and a total of five bytes would be sent to the vehicle. Note that the carriage return character is only a signal to the ELM323, and is never sent to the vehicle.

After sending a command, the ELM323 listens on the OBD bus for any responses that are directed to it. Each received byte is converted to the equivalent hexadecimal pair of ASCII characters and transmitted on the RS232 port for the user. Rather than send control characters which are unprintable on most terminals, the digits are sent as numbers and letters (eg. the hex digit 'A' is transmitted as decimal value 65, and not 10).

If there was no response from the vehicle, due to no data being available, or because the command is not supported, a 'NO DATA' message will be sent. See the error messages section for a description of this message and others.

Talking to the Vehicle

The ELM323 cannot be directly connected to a vehicle as it is, but needs support circuitry as shown in the Example Applications section. Once incorporated into such a circuit, one need only use a terminal program to send bytes to and receive them from the vehicle via the ELM323.

SAE standards specify that each group of bytes sent to the vehicle must adhere to a set format. The first byte (known as the 'mode') always describes the type of data being requested, while the second, third, etc. bytes specify the actual information required (given by a 'parameter identification' or PID number). The modes and PIDs are described in detail in the SAE document J1979 (ISO 15031-5), and may also be expanded on by the vehicle manufacturers.

Normally, one is only concerned with the nine diagnostic test modes described by J1979 (although there is provision for more). All of these modes are not required to be supported by every vehicle, and are often not. These are the nine modes:

01 : show current data
02 : show freeze frame data
03 : show diagnostic trouble codes
04 : clear trouble codes and stored values
05 : test results, oxygen sensors
06 : test results, non-continuously monitored
07 : test results, continuously monitored
08 : special control mode
09 : request vehicle information

Within each mode, PID 00 is normally reserved to show which PIDs are supported by that mode. Mode 01, PID 00 must be supported by all vehicles, and can be accessed as follows…

Ensure that the ELM323 is properly connected to your vehicle, and powered. Most vehicles will not respond without the ignition key in the ON position, so turn the ignition on, but do not start the vehicle. At the prompt, issue the mode 01 PID 00 command:

>01 00

The first time the bus is accessed, you will see a bus initialization message, followed by the response, which might typically be as follows:

41 00 BE 1F B8 10

The 41 00 signifies a response (4) from a mode 1 request from PID 00 (a mode 2, PID 00 request is answered with a 42 00, etc.). The next four bytes (BE, 1F, B8, and 10) represent the requested data, in this case a bit pattern showing the PIDs supported by this mode (1=supported, 0=not). Although this information is not very useful for the casual user, it does serve to show that your connection is working.

Another example requests the current engine coolant temperature (ECT). This is PID 05 in mode 01, and can be requested as follows:

>01 05

The response will be of the form:

41 05 7B

This shows a mode 1 response (41) from PID 05, with value 7B. Converting the hexidecimal 7B to decimal, one gets 7 x 16 + 11 = 123. This represents the current temperature in degrees Celsius, with the zero value offset to allow operation at subzero temperatures. To convert to the actual coolant temperature, simply subtract 40 from the value. In this case, then, the ECT is 123 - 40 = 83 degrees C.

A final example shows a request for the OBD requirements to which this vehicle was designed. This is PID 1C of mode 01, so at the prompt, type:

>01 1C

A typical response would be:

41 1C 01

The returned value (01) shows that this vehicle conforms to OBDII (California ARB) standards. The presently defined responses are :

01 : OBDII (California ARB)
02 : OBD (Federal EPA)
03 : OBD and OBDII
04 : OBD I
05 : not intended to meet any OBD requirements
06 : EOBD (Europe)

Some modes may provide multi-line responses (09, if supported, can display the vehicle's serial number). The ELM323 will attempt to display all responses in these cases, but only if it is allowed sufficient time to process each. There may be occasions when the vehicle responds too quickly and lines are lost.

Hopefully this has shown how typical requests proceed. It has not been meant to be a definitive source on modes and PIDs - this information can be obtained from the SAE (http://www.sae.org/), from the manufacturer of your vehicle, ISO (http://iso.org/), or from various other sources on the web.

Interpreting Trouble Codes

Likely the most common use that the ELM323 will be put to is in obtaining the current Diagnostic Trouble Codes or DTCs. Minimally, this requires that a mode 03 request be made, but first one should determine how many trouble codes are presently stored. This is done with a mode 01 PID 01 request as follows:

>01 01

To which a typical response might be:

41 01 81 07 65 04

The 41 01 signifies a response to the request, and the next data byte (81) is the number of current trouble codes. Clearly there would not be 81 (hex) or 129 (decimal) trouble codes present if the vehicle is at all operational. In fact, this byte does double duty, with the most significant bit being used to indicate that the malfunction indicator lamp (MIL, or 'Check Engine') has been turned on by one of this module's codes (if there are more than one), while the other 7 bits provide the actual number of stored trouble codes. In order to calculate the number of stored codes when the MIL is on then, one needs to subtract 128 (or 80 hex). If the result is less than 128, simply read the number of stored codes directly.

The above response then indicates that there is one stored code, and it was the one that set the Check Engine Lamp or MIL on. The remaining bytes in the response provide information on the types of tests supported by that particular module (see the SAE document J1979 for further information).

In this instance, there was only one line to the response, but if there were codes stored in other modules, they each could have provided a line of response. To determine which module is reporting the trouble code, one would have to turn the headers on (ATH1) and then look at the third byte of the three byte header for the address of the module that sent the information.

Having determined the number of codes stored, the next step is to request the actual trouble codes with a mode 03 request:

>03

A response to this could be:

43 01 33 00 00 00 00

The '43' in the above response simply indicates that this is a response to a mode 03 request. The other 6 bytes in the response have to be read in pairs to show the trouble codes (the above would be interpreted as 0133, 0000, and 0000). Note that the response has been padded with 00's as required by the SAE standard – the 0000's do not represent actual trouble codes.

As was the case when requesting the number of stored codes, the most significant bits of each trouble code also contain additional information. It is easiest to use the following table to interpret the first digit of trouble codes as follows:

If the first hex digit received is this,	Replace it with these two characters	
0	P0	Powertrain Codes - SAE defined
1	P1	" " - manufacturer defined
2	P2	" " - SAE defined
3	P3	" " - jointly defined
4	C0	Chassis Codes - SAE defined
5	C1	" " - manufacturer defined
6	C2	" " - manufacturer defined
7	C3	" " - reserved for future
8	B0	Body Codes - SAE defined
9	B1	" " - manufacturer defined
A	B2	" " - manufacturer defined
B	B3	" " - reserved for future
C	U0	Network Codes - SAE defined
D	U1	" " - manufacturer defined
E	U2	" " - manufacturer defined
F	U3	" " - reserved for future

Taking the example trouble code (0133), the first digit (0) would then be replaced with P0, and the 0133 reported would become P0133 (which is the code for an 'oxygen sensor circuit slow response'). As for further examples, if the response had been D016, the code would be interpreted as U1016, while a 1131 would be P1131.

Had there been codes stored by more than one module, or more than three codes stored in the same module, the above response would have consisted of multiple lines. As stated before, to determine which module is reporting each trouble requires turning the headers on with an ATH1 command.

Resetting Trouble Codes

The ELM323 is quite capable of resetting diagnostic trouble codes, as this only requires issuing a mode 04 command. The consequences should always be considered before sending it, however, as more than the MIL (or 'Check Engine' lamp) will be reset. In fact, issuing a mode 04 will:

- reset the number of trouble codes
- erase any diagnostic trouble codes
- erase any stored freeze frame data
- erase the DTC that initiated the freeze frame
- erase all oxygen sensor test data
- erase mode 06 and 07 test results

Clearing of all of this information is not unique to the ELM323; it occurs whenever a scan tool is used to reset your codes. The loss of this data could cause your vehicle to run poorly for a short time as well, while the system recalibrates itself.

To avoid inadvertently erasing stored information, the SAE specifies that scan tools must verify that a mode 04 is intended ("Are you sure?") before actually sending it to the vehicle, as all trouble code information is immediately lost when the mode is sent. Recall that the ELM323 does not monitor the content of messages, so it will not know to ask for confirmation of the mode request - this would have to be the duty of a software interface if one is written.

As stated, to actually erase diagnostic trouble codes, one need only issue a mode 04 command. A response of 44 from the vehicle indicates that the mode request has been carried out, the information erased, and the MIL turned off. Some vehicles may require a special condition to occur (eg. the ignition on but the engine not running) before it will respond to a mode 04 command.

That is all there is to clearing the codes. Once again, be very careful not to accidentally send an 04!

Error Messages

When hardware or data problems are encountered, the ELM323 will respond with one of the following short messages. Here is a brief description of each:

BUS BUSY

The ELM323 tried to send the mode command or initialize the bus, but detected too much activity to insert a message. This could be because the bus was in fact busy, but is often due to wiring problems that result in a continuously active input at OBDIn.

FB ERROR

This signifies that there was a 'feedback' error. When the K Line is first energized, a check is made to ensure that the signal is being fed back to OBDIn, and if it does not appear there, this message will be provided. Check your wiring before proceeding.

DATA ERROR

There was a response from the vehicle, but the information was incorrect or could not be recovered. In the case of a bus initialization, this error signifies that the format bytes received were not as expected, so initiation could not take place. If the error occurs during normal operation, it means that the response did not contain enough bytes to be a valid message (which can occur if the signal is interrupted during transmission).

<DATA ERROR

The error checksum result was not as expected, indicating a data error in the line pointed to (the ELM323 still shows you what it received). There could have been a noise burst which interfered, or a circuit problem. Try resending the command.

NO DATA

There was no response from the vehicle, even after the message was repeated three times. It may be that the vehicle has no data to offer as there is no stored information, that the mode requested is not supported, or that it was attending to higher priority issues and ignored the request. Try issuing a 01 00 command to be sure that the vehicle is ready to receive commands.

?

This is the standard response for a misunderstood command received on the RS232 bus. Usually it is due to a typing mistake.

Example Application

The SAE J1962 standard dictates that all OBD compliant vehicles must provide a standard connector near the driver's seat, the shape and pinout of which is shown in Figure 1 below. The circuitry described here can be used to connect to this J1962 plug without modification to your vehicle.

The male J1962 connector required to mate with a vehicle's connector may be difficult to obtain in some locations, and you could be tempted to improvise by making your own connections to the back of your vehicle's connector. If doing so, we recommend that you do nothing which would compromise the integrity of your vehicle's OBD network. The use of any connector which could easily short pins (such as an RJ11 type telephone connector) is definitely not recommended.

The circuit of Figure 2 on the next page shows how the ELM323 would typically be used. Circuit power is obtained from the vehicle (via OBD pins 16 and 5) and, after some minor filtering, is presented to a five volt regulator. Note that a few vehicles have been reported not to have a pin 5. On these, you use pin 4 instead of pin 5. The regulator powers several points in the circuit as well as an LED (for visual confirmation that power is present).

The remaining two connections to the vehicle (OBD pins 7 and 15) are for the two data lines prescribed by the ISO 9141 and ISO 14230 standards. To meet the standards, the ELM323 controls both lines through the NPN transistors shown, with the pullup resistors connected to their collectors. The 510Ω value for these resistors is specified in the standards, and substituting for a larger value would only increase rise times, likely making the circuit inoperable. Reducing the value could cause circuit damage, so try to keep as close as possible to the 510Ω. Note also that 1/2W resistors should be used (although 1/4W 240Ω + 270Ω will work, too).

Data is received from the K Line of the OBD bus and inverted by the PNP transistor shown before being applied to pin 11 of the ELM323. This transistor raises the threshold voltage to about 4V from the inherent 2.5V with the CMOS input of the ELM323. This helps to increase noise immunity while reducing transition times at the input pin, due to the amplification.

A very basic RS232 interface is shown connected to pins 5 and 6 of the ELM323. This circuit 'steals' power from the host computer in order to provide a full swing of the RS232 voltages without the need for a negative supply. The RS232 pin connections shown are for a 25 pin connector. If you are using a 9 pin, the connections would be 2(RxD), 5(SG) and 3(TxD).

RS232 data from the computer is directly connected to pin 5 of the IC through only a 47KΩ current limiting resistor. This resistor allows for voltage swings in excess of the supply levels while preventing damage to the ELM323. A single 100KΩ resistor is also shown in this circuit so that pin 5 is not left floating if the computer is disconnected.

Transmission of RS232 data is via the PNP transistor shown connected to pin 6. This transistor allows the output voltage to swing between +5V and the negative voltage stored on the 0.1µF capacitor (which is charged by the computer's TxD line). Although it is a simple connection, it is quite effective for this type of application. Note also that the ELM323's pin 4 has been tied to V_{DD}, so that by default linefeed characters will be sent whenever a carriage return is sent.

The four LEDs shown (on pins 7 to 10) have been provided as a visual means of confirming circuit activity. Resistors are shared among Tx and Rx LEDs as they will not be on at the same time (the ELM323 is not capable of true multitasking). The OBD bus may be in an initialization phase while data is being sent or received on the RS232 bus, though, so separate resistors are shown for these two groups.

Finally, the crystal shown connected between pins 2 and 3 is a common TV type that can be easily and inexpensively obtained. The 27pF crystal loading capacitors shown are typical only, and you may have to select other values depending on what is specified for the crystal you obtain.

This completes the description of the circuit. While it is the minimum required to talk to an OBD equipped vehicle, this is a fully functional circuit. You may want to expand on it, though, providing more protection from faults and electrostatic discharge, or providing a different interface for the RS232 connection to the computer. Then perhaps a Basic program to make it easier to talk to the vehicle, and a method to log your findings, and…

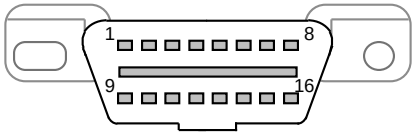


Figure 1. Vehicle Connector

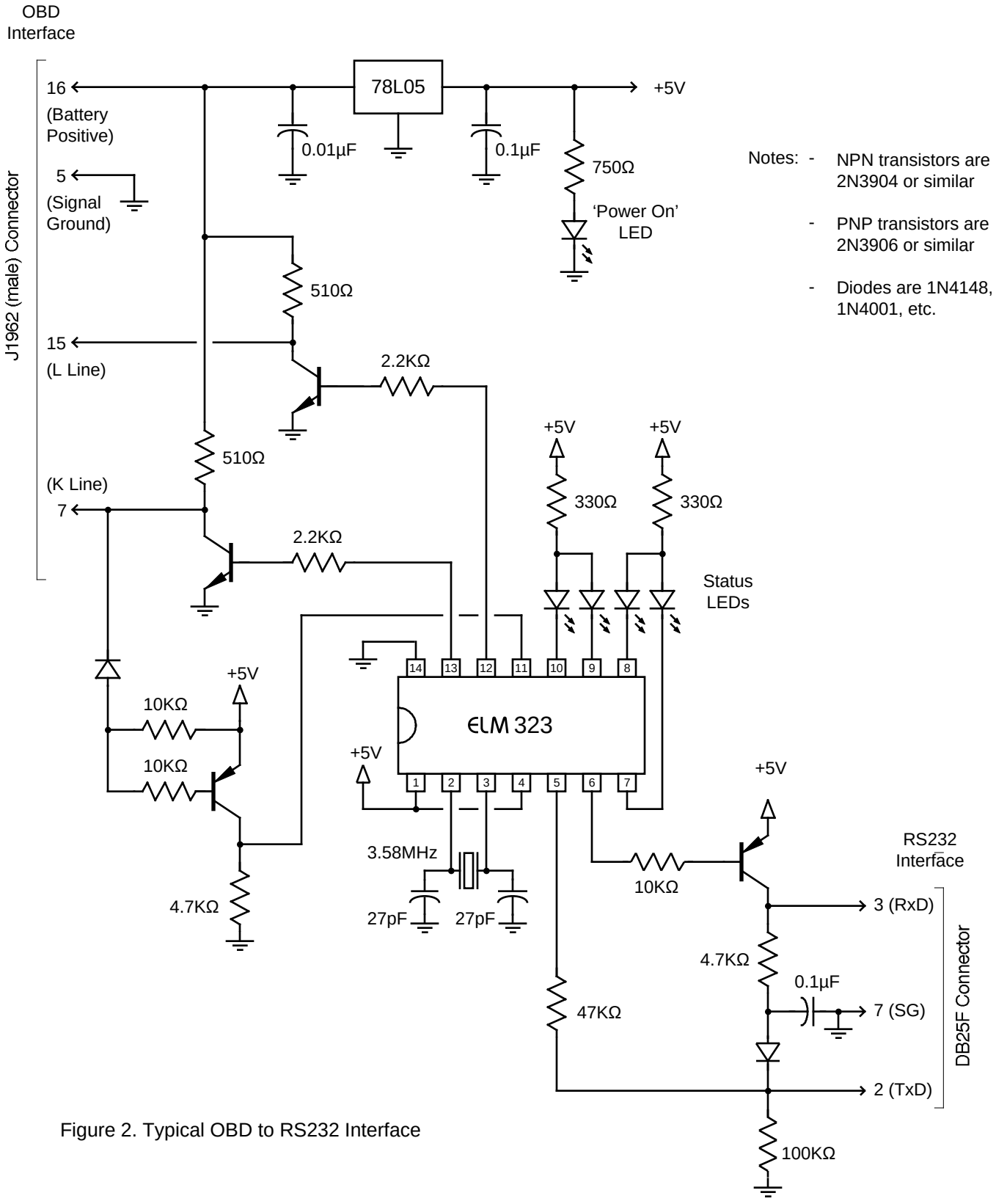


Figure 2. Typical OBD to RS232 Interface