ELM323

OBD (ISO) to RS232 Interpreter

Description

Since the 1996 model year, North American automobiles have been required to provide an OBD (On Board Diagnostics) data port for the connection of test equipment. This port is used to obtain emissions-related diagnostics information, and in some cases can also be used to obtain real-time vehicle operating parameters.

The ELM323 is a 14 pin integrated circuit that, with only a few external components, is able to convert between the OBD data format and the standard RS232 serial data format. 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 even create their own custom 'scan tool' by adding an interface program.

Please note that while this integrated circuit has undergone significant changes recently, it is still considered to be an experimenter's circuit. It does support a great many of the ISO9141 and ISO14230 (KWP2000) protocol vehicles, but not all of them, due to the many interpretations of these standards that are to be found in use.

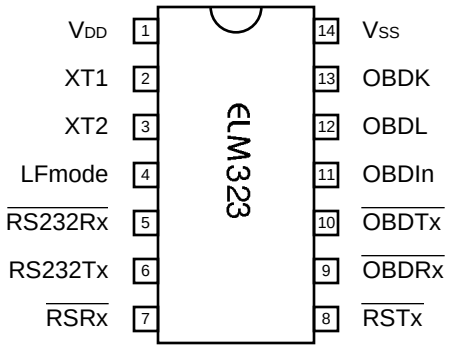
Applications

- Diagnostic trouble code readers
- Automotive scan tools
- Teaching aids

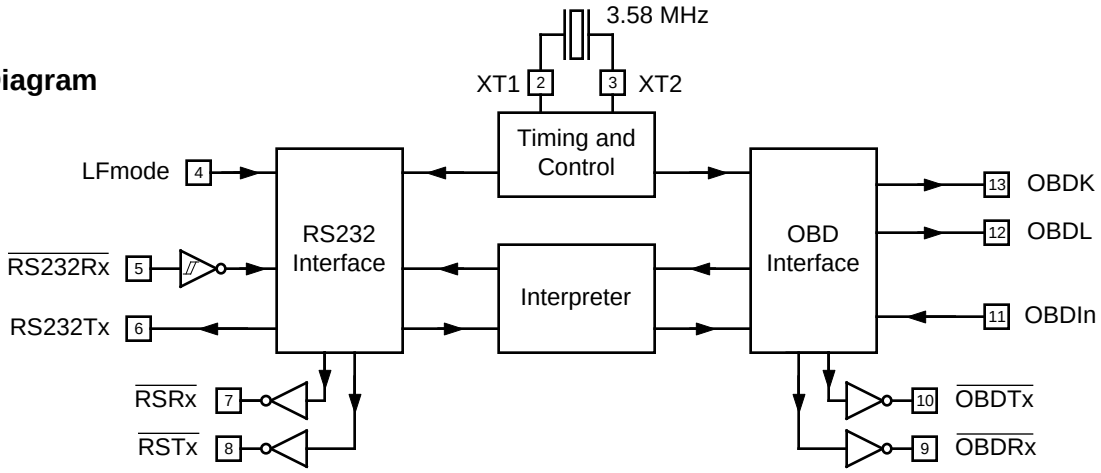
Features

- Low power CMOS design
- Crystal controlled for accuracy
- ISO 9141-2 and ISO 14230-4 protocols
- Configurable with AT commands
- Standard ASCII character output
- Four high current LED drive outputs

Connection Diagram
PDIP and SOIC
(top view)



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

LFmode (pin 4)

This input is used to select the default linefeed mode after a power-up or system reset. If it is at a high level, then by default messages 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 AT L0 or AT L1 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 resistor current 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 Applications 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

All rights reserved. Copyright 2001 to 2009 by Elm Electronics Inc.
Every effort is made to verify the accuracy of information provided in this document, but no representation or warranty can be given and no liability assumed by Elm Electronics with respect to the accuracy and/or use of any products or information described in this document. Elm Electronics will not be responsible for any patent infringements arising from the use of these products or information, and does not authorize or warrant the use of any Elm Electronics product in life support devices and/or systems. Elm Electronics reserves the right to make changes to the device(s) described in this document in order to improve reliability, function, or design.

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).

Overview

The following describes how to use the ELM323 to obtain a great deal of information from your vehicle. To some, the following information will be overwhelming, and for others it will not be nearly enough.

We begin by discussing just how to talk to the IC using a PC, then go on to explain how to change options using 'AT' commands, and finally go on to actually use the ELM323 to obtain trouble codes (and reset them). For the more advanced experimenters, there are also sections on how to use some of the programmable features of this product as well.

Using the ELM323 is not as daunting as it first seems. Many users will never need to issue an 'AT' command, adjust timeouts or change the headers. For most, all that is required is a PC or a PDA with a terminal program (such as HyperTerminal or ZTerm), and knowledge of one or two OBD commands, which we provide in the following…

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 linefeed 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 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 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 15 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

Several parameters within the ELM323 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 the settings; for example by turning the character echo off, adjusting a timeout value, or changing the header bytes. In order to do this, internal 'AT' commands must be issued.

Those familiar with PC modems will immediately recognize AT commands as a standard way in which modems are internally configured. The ELM323 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 internal configuration or 'AT' commands, and will be executed upon receipt of a terminating carriage return character. The ELM323 will usually reply with the characters 'OK' on the successful completion of a command, so the user knows that it has been executed.

Some of the following commands allow passing numbers as arguments in order to set the internal values. These will always be hexadecimal numbers which must be provided in pairs. The hexadecimal conversion chart in the OBD Commands section may prove useful if you wish to interpret the values. Also, one should be aware that for the on/off types of commands, the second character is a number (1 or 0), the universal terms for on and off respectively.

The following is a summary of all of the AT commands that are recognized by the current version of the ELM323, listed alphabetically. Users of previous versions of this product (v1.x) should note that their ICs will not support all of the functions shown.

AR [Automatically set the Receive address]

Responses from the vehicle will be acknowledged and displayed by the ELM323, if its internally stored receive address matches the address that the message is being sent to. With the Auto Receive mode in effect, the value used for the receive address will be chosen based on the current header bytes, and will automatically be updated whenever the header bytes are changed.

The value that is used for the receive address is determined based on several factors. If the IC is connected to a KWP2000 (ISO14230) system, the third byte of the header will always be used as the receive address. If the IC is connected to an ISO9141 system, the receive address will depend on the contents of the first header byte. If the first byte shows that the message uses physical addressing, the third byte of the header will be used for the address, otherwise (for functional addressing) the second header byte, increased in value by 1, will be used. Auto Receive is turned on by default.

BD [OBD receive Buffer Dump]

There may be times when a bus initialization is not successful, or perhaps the OBDRx LED flickers but nothing is sent on the RS232 connection. In these cases, it may be an advantage to see just what was received on the OBD connection. This command shows the entire OBD buffer contents as a length byte followed by 11 data bytes. Since not all of the data bytes are likely to be relevant, be sure to check the value contained in the length byte before interpreting the data. The format of the data returned by this command will follow the data mode in effect at the time (Packed Data or Formatted Data) so you may want to adjust that before viewing the data.

D [set all to Defaults]

This command is used to set all of the options to their default or 'factory' settings, as listed in these pages. This lets you experiment with different settings, but be able to quickly restore them all to the original settings using only one command.

To summarize the changes, E will be on, H will be off, and L will be set according to the level at pin 4. The Auto Receive mode (AR) will be selected, data will be transmitted in the standard formatted way (as if chosen by FD), and both the 'NO DATA' timeout and the period between bus idle (wakeup) messages will be reset to their default values. As well, the header bytes will be set to the prescribed values for OBDII operation, and the receive address will be adjusted accordingly. If the bus had been initiated, it will remain active.

AT Commands (continued)

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. The default is E1 (echo on).

FD [send Formatted Data]

This command requests that all of the vehicle's responses be returned as standard ASCII characters (which are readable with virtually all terminal programs). Byte values are sent as a pair of ASCII characters, each representing a hexadecimal digit, with a space character sent between each pair as a separator. Every line will end with a carriage return character and (optionally) a linefeed character, ensuring that each response appears on a new line. This is the default output mode.

FI [perform a Fast bus Init]

Issuing this command forces the ELM323 to perform a fast (KWP / ISO14230) bus initialization sequence, regardless of the present state of the bus. Note that the bus does not need to be manually initalized with this command, as it will be performed automatically by the ELM323 when required. (It will first try a slow initialization, and if it is not successful, it will then attempt a fast initialization sequence.)

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

These commands 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, which are 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 'ELM323 v2.0'). Software can use this to determine exactly which version of the integrated circuit it is connected to, without having to reset the IC.

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

Whether or not the ELM323 transmits a linefeed character after each carriage return character is controlled by this option. If an ATL1 is issued, linefeed generation will be turned on, and for AT L0, 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 level at pin 4 when the IC is reset (power-on or AT Z), or when the default values are restored (AT D).

MA [Monitor All messages]

Using this command places the ELM323 into a bus monitoring mode, in which it displays all messages as it sees them on the OBD bus. This continues indefinitely until stopped by activity on the RS232 input. To stop the monitoring, one should send any single character then wait for the ELM323 to respond with a prompt character ('>'). Waiting for the prompt is necessary as the response time is unpredictable, varying depending on what the IC was doing when interrupted. If, for instance, it was in the middle of printing a line, it will first complete the line then return to the command state, then issue the prompt character. If the ELM323 was simply waiting for OBD input, it would return immediately. The character which stops the monitoring will always be discarded, and will not affect subsequent commands.

This command has been provided as a convenience, and should be used with caution. No periodic 'wakeup' messages are sent by the ELM323 while monitoring the bus in this mode, so if the bus had been initialized before this command was invoked, the vehicle connection will likely 'go to sleep' and will have to be re-initialized. The ELM323 will not be aware that the connection was lost, however, and will likely have to be reset with an AT SW 00, or an AT Z command.

PD [send Packed Data]

This option is for those that are building a computer interface and want the fastest data transfer rate possible while still operating at 9600 baud. When selected, all of the data obtained from the vehicle will be sent as a single length byte, followed by the actual

AT Commands (continued)

response bytes that were received from the vehicle. There will be no added space characters, and no trailing carriage returns or linefeed characters inserted between messages. This provides a very compact format for data transfer.

Note that the length byte only represents the total number of data bytes following, and does not include itself. Also, if there was a data (checksum) error, this length byte will have its most significant bit set, making it appear that the length is greater than 128. If you ignore the most significant bit (or subtract 128 from the value), the other 7 bits will still provide a valid byte count for the remainder of the message.

When the vehicle does not provide a response to a query (a 'NO DATA' condition), the response has no data bytes, but still sends a length byte with value '0'.

SH xx yy zz [Set the Header to xx yy zz]

This command allows the user to control the values that are sent as the three initial (header) bytes in each message. The value of hex digits xx will be used for the first or priority/type byte, yy will be used for the second or target byte, and zz will be used for the third or source byte. These values will remain in effect until set again, or until restored to the default values with the AT D, or AT Z commands.

For ISO9141 vehicles, the default header values are 68 6A F1, while for ISO14230 (KWP2000), they are Cn 33 F1, where n represents the number of data bytes in the message. Note that when assigning header bytes for ISO14230 systems, whatever value you provide for 'n' will be ignored by the ELM323, and the appropriate value will automatically be calculated and inserted just before sending each message.

A feature has been added to this version of the IC to allow experimenting with the header bytes, while not affecting the periodic 'wakeup' messages. That is, a separate set of header bytes can be used for the periodic wakeup messages and for those used by the standard requests. This is accomplished by first assigning header bytes (or leaving them as the default ones) then initializing the OBD bus. Whatever header bytes are being used when the bus is initialized will be 'locked in' at that point and used for all of the periodic messages until the IC is reset (or the AT SW 00 is used to turn them off). Issuing the AT SH command after the bus has been initialized will only affect the requests that follow, and will have no effect on any of the periodic wakeup messages.

SI [perform a Slow bus Init]

Issuing this command forces the ELM323 to perform a slow (5 baud) bus initialization sequence, regardless of the present state of the bus. Note that the bus does not have to be manually initialized with this command. If it is not active when a command is issued to the vehicle, the ELM323 will automatically try a slow initialization, and if that is not successful, will then attempt a fast initialization.

SR hh [Set the Receive address to hh]

Depending on the application, users may wish to manually set the address for which the ELM323 will display responses. Issuing this command will turn off the AR mode, and force the IC to only accept responses from the vehicle that are addressed to hh, ignoring all others.

ST hh [Set Timeout to hh]

After sending a request, the ELM323 waits a preset time before declaring that there was no response from the vehicle (the 'NO DATA' response). Depending on the application (and priority of the request), users may want to modify this timeout period to allow more or less time. The ST command is used to do that.

The actual time allowed before a timeout occurs is (approximately) 4 ms x the byte value passed as the hexadecimal argument. Passing a value of FF thus results in a maximum time of about 1020 ms. Note that a setting of 00 (zero) is not allowed, and will be replaced internally with the default setting value – hex 32 (decimal 50) resulting in a timeout value of 200 milliseconds.

SW hh [Set Wakeup to hh]

Once a data connection is made with a vehicle, there needs to be data flow every few seconds, or the connection will 'go to sleep'. The ELM323 will automatically generate 'wakeup' messages in order to maintain this connection whenever the user is not requesting any data. (The responses from these messages are always ignored, and not seen by the user.)

The time between these periodic 'wakeup' messages can be adjusted in 20 msec increments using the AT SW hh command, where hh is any

AT Commands Summary

hexadecimal value from 00 to FF. The maximum possible time delay of 5.10 seconds thus occurs when a value of FF (or decimal 255) is used. The default setting is 7D (or decimal 125) which provides a nominal delay of 2.5 seconds between messages.

Note that the value 00 (zero) is treated as a very special case, and must be used with caution. It is used to stop all periodic messages, by telling the ELM323 internally that the bus is no longer active. This can be convenient if the vehicle has timed out (perhaps when using the AT MA command) and you wish to inform the ELM323 of that without performing a full reset. Issuing AT SW 00 will not change the previous setting for the time between wakeup messages.

Attempting to communicate with the vehicle after issuing AT SW 00 will result in the ELM323 perfoming a bus initialization sequence.

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 in the idle state, waiting for characters on the RS232 bus.

Figure 1 shows all of the current ELM323 AT commands in one convenient chart. In order to help with the understanding of these, we have grouped the commands into four functional areas, but this has no bearing on how they need to be used – it is only for clarity. You may find this chart to be useful when experimenting with the IC.

ELM323 AT Commands

general
- **D** set all to Defaults
- **I** show the ID string
- **Z** reset all
- **<CR>** repeat last command

bus control
- **FI** Fast Init
- **SI** Slow Init
- **SW hh** Set Wake (hh*20ms)

responses
- **E1/0** Echo on/off
- **H1/0** Headers on/off
- **L1/0** Linefeeds on/off
- **FD** use Formatted Data
- **PD** use Packed Data
- **ST hh** Set Timeout (hh*4ms)

requests
- **BD** Buffer Dump
- **MA** Monitor All
- **SH xx yy zz** Set Header
- **AR** Auto Receive
- **SR hh** Set Rx address

Figure 1. ELM323 Command Summary

Bus Initiation

Both the ISO 9141-2 and ISO 14230-4 (KWP2000) standards require that the vehicle's OBD bus be initialized before any communications can take place. The ISO 9141 standard allows for only a slow (2 to 3 second) process, while ISO 14230 allows for both the slow method, and a faster alternative. In either case, once the bus has been initiated, communications must take place at least once every five seconds, or the bus will revert to a low-power 'sleep' mode.

The ELM323 takes care of this bus initiation and the periodic sending of 'keep-alive' or 'wakeup' messages for you – it is automatic and requires no input from the user. The ELM323 will not perform the bus initiation until the first message needs to be sent, however, and it will do so by first attempting the slow method, and if that fails then trying the fast. During the automatic initiation process, the following message will be displayed:

```
BUS INIT: ...
```

with the three dots appearing as the slow initiation 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 that there was a problem. (The most common error encountered 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 monitoring LEDs, you will be able to see that automatic messages are being sent every few seconds in order to create bus activity.

If the user does not wish to use the two step automatic bus initiation process, they can specify that only the Slow Initiation, or only the Fast Initiation, be attempted, by issuing the commands AT SI or AT FI respectively. Note that the three dots are only printed during a slow initiation, so if AT FI is issued, they will not appear.

The chart at the right shows the automatic bus initiation process in more detail:

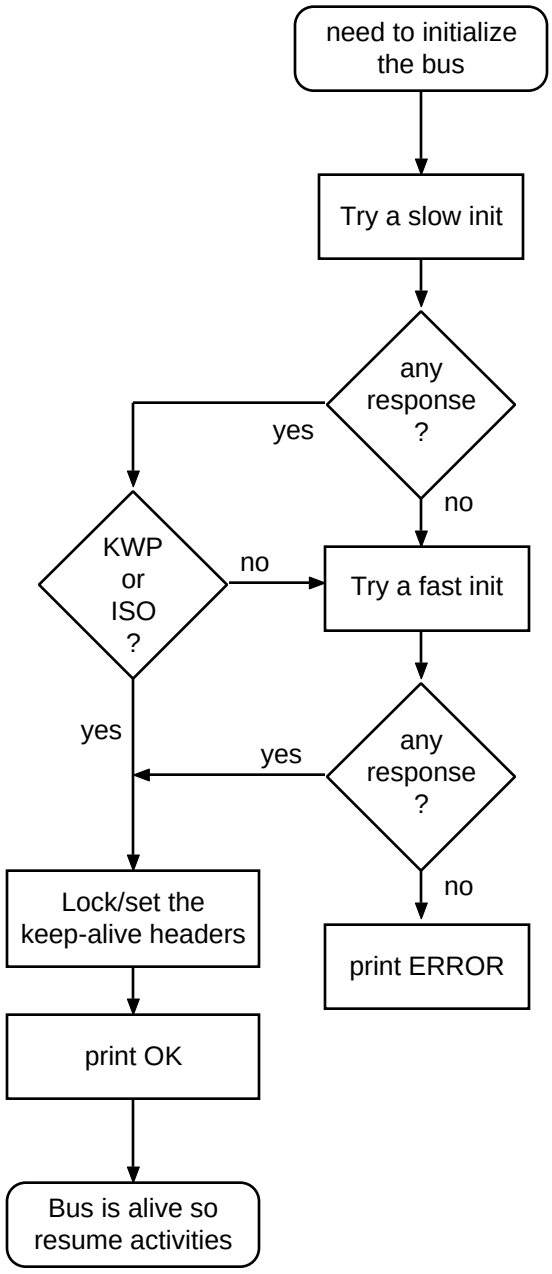


Figure 2. Initializing the Bus

OBD Commands

If the bytes received on the RS232 bus do not begin with the letters ‘A’ and ‘T’, they are assumed to be OBD commands for the vehicle. Each pair of ASCII bytes will be tested to ensure that they are valid hexadecimal digits, and will then be combined into single data bytes for transmitting to the vehicle.

OBD commands are actually sent to the vehicle embedded in a data packet. The standards require that three header bytes and an error checksum byte be included with every message, and the ELM323 adds these extra bytes to your command bytes automatically. The initial (default) values for these header bytes are usually adequate for most requests, but if you wish to change them, there is a method to do so (see the Advanced Data Retrieval section).

When receiving data from a vehicle, the extra header bytes are not normally displayed by the ELM323. Occasionally vehicles will have more than one module responding to a request, though, and it may be useful to see the extra header bytes in order to determine which ECU module responded. (The third byte of the response is the address of the sender). To view these extra header bytes, simply issue an AT H1 internal command, to turn the header printing on.

Most OBD commands are only one or two bytes in length, but some can be three or more bytes long. The ELM323 is capable of sending as many as seven data bytes (14 hexadecimal digits), the maximum number allowed by the standards. Attempts to send either an odd number of hex 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 use a table such as Figure 3, 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 to be more 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 by way of the RS232 port. The ELM323 would store the characters as they are received, and when the third character (the carriage return) was received, would 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). Three header bytes and a checksum byte 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 not sent on to the vehicle.

After sending the command, the ELM323 listens on the OBD bus for messages, looking for ones that are directed to it. If a message address matches, those received bytes will be sent on the RS232 port to the user, while messages received that do not have matching addresses will be ignored (but still available for viewing with the AT BD command).

The ELM323 will continue to wait for messages addressed to it until there are none found in the time that was set by the AT ST command. As long as messages are received, the ELM323 will continue to reset this timer. Note that the IC will always respond with something, even if it is to say ‘NO DATA’, (meaning that there were no messages at all addressed to it).

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 3. Hex to Decimal Conversion

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, and then 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 with 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 prove that the 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 hexadecimal 7B to decimal, one gets 7 x 16 + 11 = 123. This represents the current temperature in degrees Celsius, but the zero value is offset to allow for subzero temperatures. To convert to the actual coolant temperature, simply subtract 40. In this case, then, the coolant temperature 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. Some of the 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 sends information too rapidly and some intermediate lines are lost.

Hopefully this has shown how typical requests proceed. It has not been meant to be a definitive guide on modes and PIDs – this information can be obtained from the manufacturer of your vehicle, from the SAE (www.sae.org), ISO (www.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 of this byte provide the actual number of stored trouble codes. In order to calculate the number of stored codes when the MIL is on, then, subtract 128 (or 80 hex). When 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 (AT H1) 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 for this mode – 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 extra bits in the first digit 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.

More than one ECU module can respond to requests such as this, so be prepared to possibly receive several lines of responses. To determine which ECU is reporting each line would require turning the headers on with the AT H1 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 the codes. The biggest problem with losing this data is that your vehicle may run poorly for a short time, while it performs a recalibration.

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 the 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 they 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 message is sent when a 'feedback' error is detected. When the K Line is first energized, a check is made to ensure that the signal is seen at OBDIn. If it does not appear there, this message is displayed. 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 required, so initiation could not continue. 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 a data 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 re-sending the command.

NO DATA

The IC waited for the period of time that was set by AT ST, and detected no response from the vehicle. It may be that the vehicle had no data to offer, that the mode requested was not supported, or that the vehicle was attending to higher priority issues and could not respond to the request in the allotted time. Try adjusting the AT ST time to be sure that you have allowed sufficient time to obtain a response.

?

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

Monitoring the Bus

Some vehicles use the OBD bus for information transfer during normal vehicle operation, passing a great deal of information over it. A lot can be learned if you have the good fortune to connect to one of these vehicles, and are able to decipher the contents of the messages. Other vehicles cannot be initialized, and instead continually send information; the only way to read the data from them is by monitoring everything that is being sent, and extracting the useful data.

To see how your vehicle uses the OBD bus, you will have to enter the ELM323's 'Monitor All' mode, by sending the command AT MA from your terminal program. Once received, the IC will continually display information that it sees on the OBD bus, regardless of transmitter or receiver addresses. Note that the periodic 'wakeup' messages are not sent while in this mode, so the bus may 'go to sleep' in a short time.

The monitoring mode can only be stopped by sending a single character over the RS232 connection to the ELM323. Any single character will interrupt the IC, returning it to the command mode (waiting for an input). Note that the character you send will be discarded, and will have no effect on any subsequent commands. The time it takes to respond to this interrupting character will depend on what the ELM323 is doing when it is received. The IC will always finish a task that is in progress (printing a line, for example) before returning to wait for input, so you should always wait for the prompt character ('>') before continuing to issue other commands.

If the headers are not currently displayed, simply typing AT MA shows only the contents of messages, not the transmitter and receiver addresses. To show who is sending to whom, you will need to first turn headers on (AT H1) before beginning to monitor (AT MA).

There is a slight possibility that OBD messages could be missed by the ELM323 while it is retransmitting a previous OBD message on the RS232 connection. This is due to the fact that the ELM323 is a single-tasking microprocessor that does not have hardware to buffer all of the OBD data in the background while it is performing other tasks. Should an OBD message begin while the IC is 'talking' on the RS232 bus, several bytes may be missed, and a '<DATA ERROR' message will likely be displayed. Usually the vehicle ECUs provide tens of milliseconds between messages and this is not a problem, but we are just warning that if an ECU should be transmitting data at a very high rate, it may overwhelm the ELM323, and DATA ERRORs could result. If this occurs, you may want to reduce the amount of RS232 data sent by turning linefeeds off, and using the 'packed data' mode. Most users will never encounter this problem, and so this limitation will not be noticeable.

Computer Control – Using Packed Data

If a person is simply asking a vehicle for the current Diagnostic Trouble Codes, speed is normally not an issue, as data is displayed (essentially) as quickly as it can be read. If interfaced to a computer, however, speed may be very important.

The 'Packed Data' mode is a convenient means to effectively triple the ELM323's data transfer rate while maintaining the 9600 baud connection. Once entered (with AT PD), all OBD messages will be returned as a single length byte followed by the actual data bytes. There are no space characters sent between the bytes, and no carriage returns or linefeeds between messages either – the data is simply retransmitted exactly as received from the vehicle. While no longer readable by a person, computers will understand this information, and will gain speed through both reduced transfer and reduced data conversion times. The ELM323 does not function any differently when in this mode – if the headers are to be displayed, they are sent, if in monitoring mode, data is continually sent, etc. The only difference is in the format in which the OBD responses are returned to the controlling computer.

Often there is no response from the vehicle for a particular request. When in the default (formatted data) mode, this is shown with 'NO DATA' being printed, but while in the Packed Data mode you will only receive a single length byte of value 0 (zero).

While rare, errors may occasionally be detected in the vehicle's data. Normally, a '<DATA ERROR' would be printed for this, but in the Packed Data mode, checksum errors are identified by setting the most significant bit of the length byte. Because of this, one should always check the length byte for a value of 128 or greater before processing the remainder of the message.

Advanced Data Retrieval – Setting the Headers

Prior to v2.0, the ELM323 used a fixed format for the message headers, so that only one type of data (the mandated emissions-related information) could be retrieved. Starting with v2.0, the header bytes can be specified by the user, allowing for the direct retrieval of a great deal more data. Note that only the OBDII diagnostic codes have been mandated, so there is no requirement for all vehicles to support these extra capabilities, and some do not.

The emissions related diagnostic trouble codes that most people are familiar with are described in the SAE J1979 standard (ISO15031-5), and are really a specific instance of the modes allowed by the J2178-4 standard, which provides for information transfer by what is known as 'functional addressing'. For the OBDII mandated diagnostics, requests are actually made to the functional addresses 6A (for ISO9141) or 33 (for ISO14230), with whatever processor that is responsible for that function answering the request. Theoretically, many different processors can respond to a single functional request, each contributing their own data.

To retrieve information beyond that of the OBDII requirements, either the functional or the ECU physical 'address' needs to be known. For example, consider that you want to request that the processor responsible for Engine Coolant provide the current Fluid Temperature, and you do not know its address. You consult the J2178 standards and determine that Engine Coolant is functional address 48. Combining this with the knowledge that the ELM323 does not support in-frame responses (so it only allows message types 8 to 15), and a scan tool is normally address F1, you may decide to set the three header bytes to A8 48 and F1. This is done with the Set Header command, which is used as follows:

```
>AT SH A8 48 F1
```

The three header bytes assigned in this manner will stay in effect until changed by the next AT SH command, a reset, or an AT D. If the default Auto Receive mode is in effect when the header bytes are set, the ELM323 will also adjust the receive address as appropriate – since the first byte tells us that this is a functional address, then for ISO9141 systems, the receive address will automatically be set to the functional address plus one (49). For ISO14230 systems, the physical address of the sender (F1) will be used for the receive address. If you decide that this is not appropriate for your case, you can always set another receive address, using the AT SR command.

Having set the headers, all one needs to do is issue the secondary ID for fluid temperature (10) at the prompt. If the display of headers is turned off, the conversation could look like this:

```
>10

10 2E
```

The response to ID 10 is the byte 2E in this case. You may find that some requests, being of a low priority, may not be answered immediately, possibly causing a 'NO DATA' result. In these cases, you may want to adjust the timeout value, perhaps first trying the maximum (with AT ST FF).

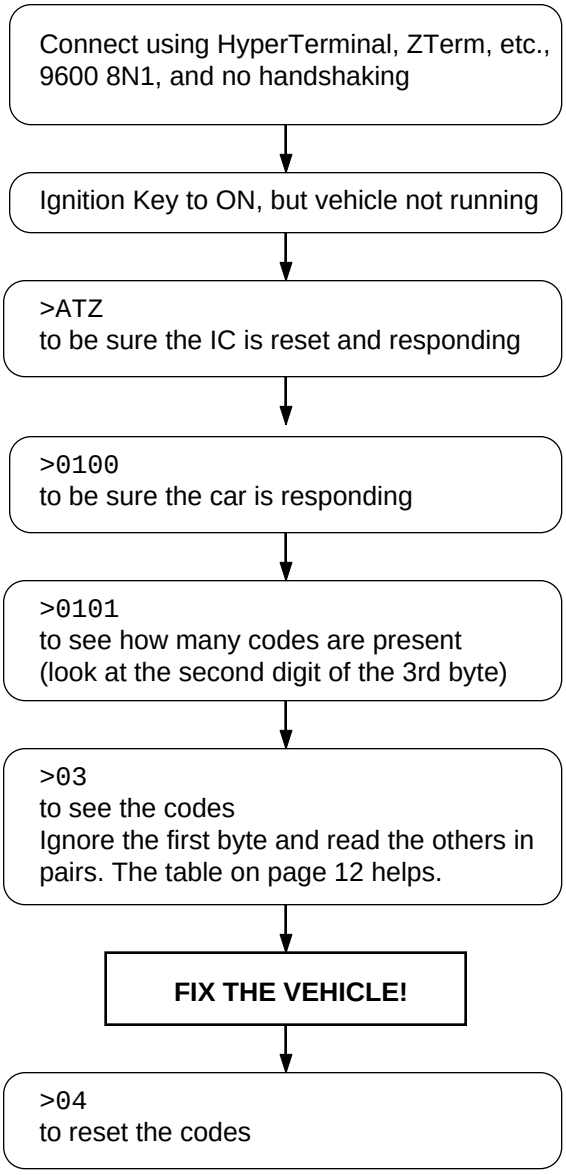
Using the physical addressing modes described by the J2190 standard involves an almost identical process. The main difference is that you must know the physical address of the device that you want to speak to (it is always the third byte of any message sent by that device), rather than the functional address. One caution to note with physical addressing is that there are modes which can initiate the constant sending of data, and if the ELM323's timeout is set longer than the time between responses, the ELM323 may send responses forever. In these cases, just like in the Monitoring All mode, a single character will have to be sent to interrupt the stream of data.

Advanced experimenters will be aware that the ISO14230 standard also specifies that the first header byte must always include the length of the data field. The ELM323 will calculate and insert these six bits automatically for each message, no matter what you provided for them in your header definition. It does not however, alter the two most significant bits of that first header byte.

Finally, please note that the headers that are in effect when the bus is initiated will remain locked-in for all of the following 'keep-alive' messages, no matter what is issued afterwards using AT SH. These same header bytes will also be the ones used for the 'start communications' message that is generated by the ELM323 during a fast initiation. This is provided as a feature to allow experimentation with the header bytes once a connection is established, but it may cause confusion if you first try to set the headers, then initiate the bus, only to find that the initiation fails. Until you are familiar with your vehicle, it is recommended that you send a simple command (01 00 or such) to first start bus activity, then try changing the header bytes.

Quick Guide for Reading Trouble Codes

If you don't use your ELM323 for some time, this entire data sheet may seem like quite a bit to review if your 'Check Engine' light does eventually come on. The following provides a quick procedure which may prove helpful in that case (note that the '>' is the ELM323's prompt character):



Example Applications

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 4 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 that 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 5 on the next page shows how the ELM323 could typically be used. Circuit power is obtained from the vehicle (via OBD pins 16 and 5) and, after some capacitive 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, possibly 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 (and that 1/4W 240Ω + 270Ω resistors work well, 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, because of 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 standard 9 pin connector. If you are using the older 25 pin style, please refer to the web site help pages for the equivalent pins.

RS232 data from the computer is directly connected to pin 5 of the ELM323 through 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). Using the computer's own supply guarantees that the RS232 voltage levels will be compatible. 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 Figure 5. While it is the minimum required to talk to an OBD equipped vehicle, it is a fully functional circuit. Page 19 shows one more example circuit – that of an OBD monitor.

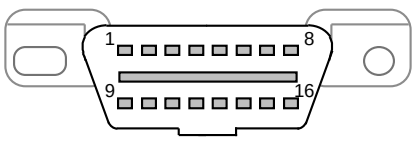


Figure 4. Vehicle Connector

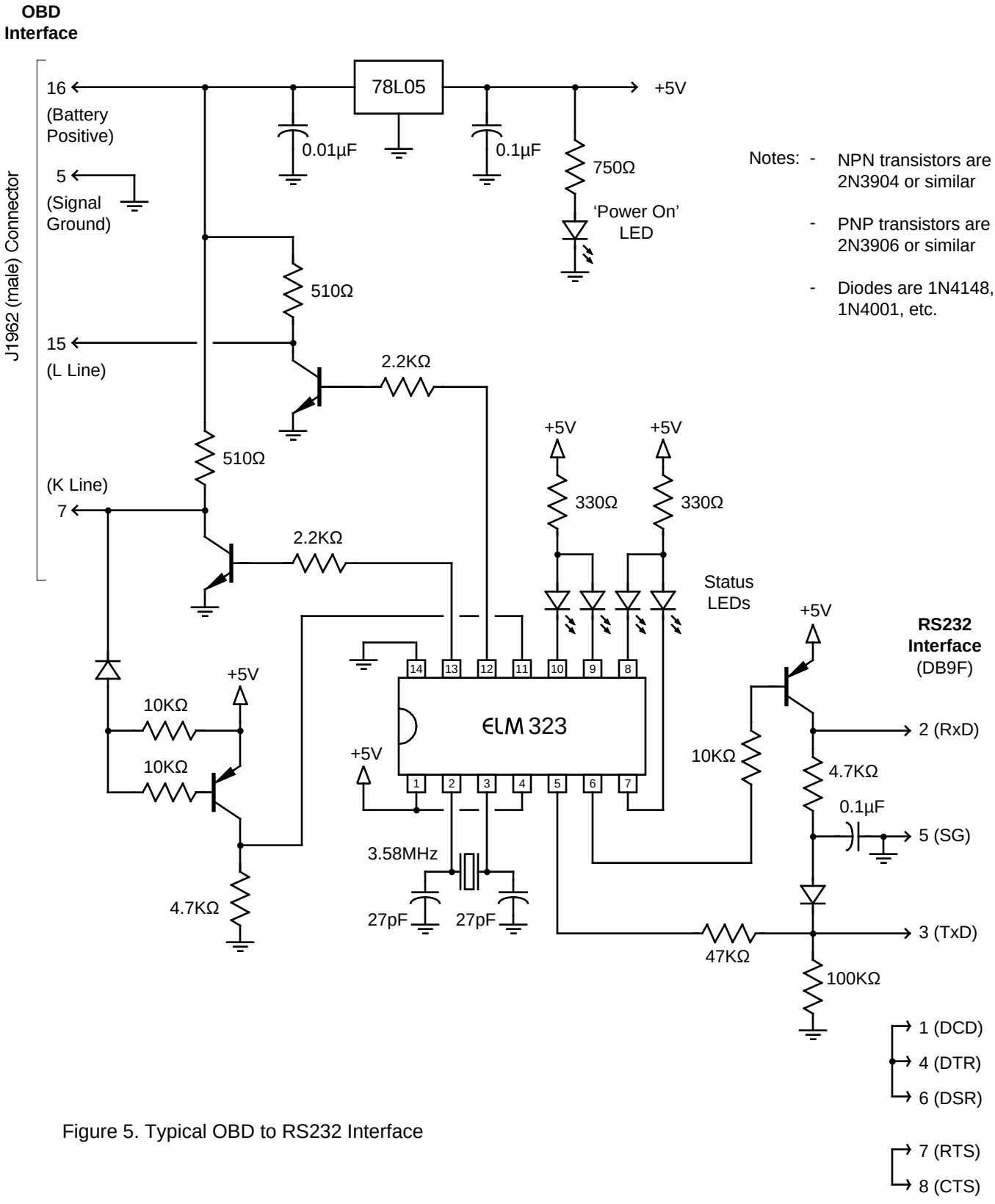
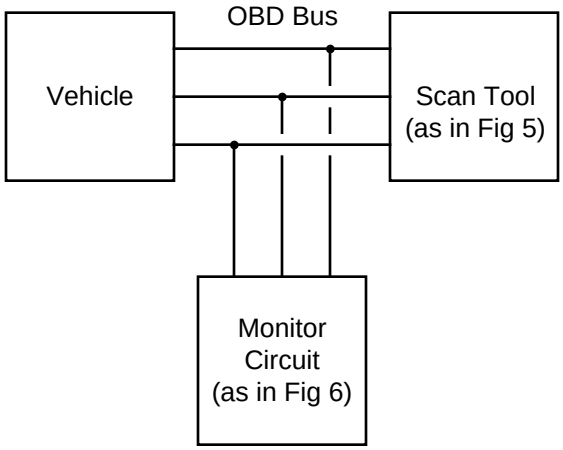


Figure 5. Typical OBD to RS232 Interface

Example Applications (continued)

As a final example, we provide an OBD monitor. There are times when it would be convenient to be able to simply monitor the OBD bus for one reason or another – personal learning, monitoring others in teaching environments, and also there are apparently some vehicles produced that continually send OBD information, so can not be 'read' in the standard way.

For these situations, a simplified version of the circuit in Figure 5 can be used as shown in Figure 6 below. The K and L line bus transmit interfaces have been removed as they are no longer required (and would only serve to load the bus down). The simplified three-wire interface is connected to the OBD bus as shown at right, and an AT MA command is issued (refer to the Monitoring the Bus section for more information on that command). That's all there is to it!



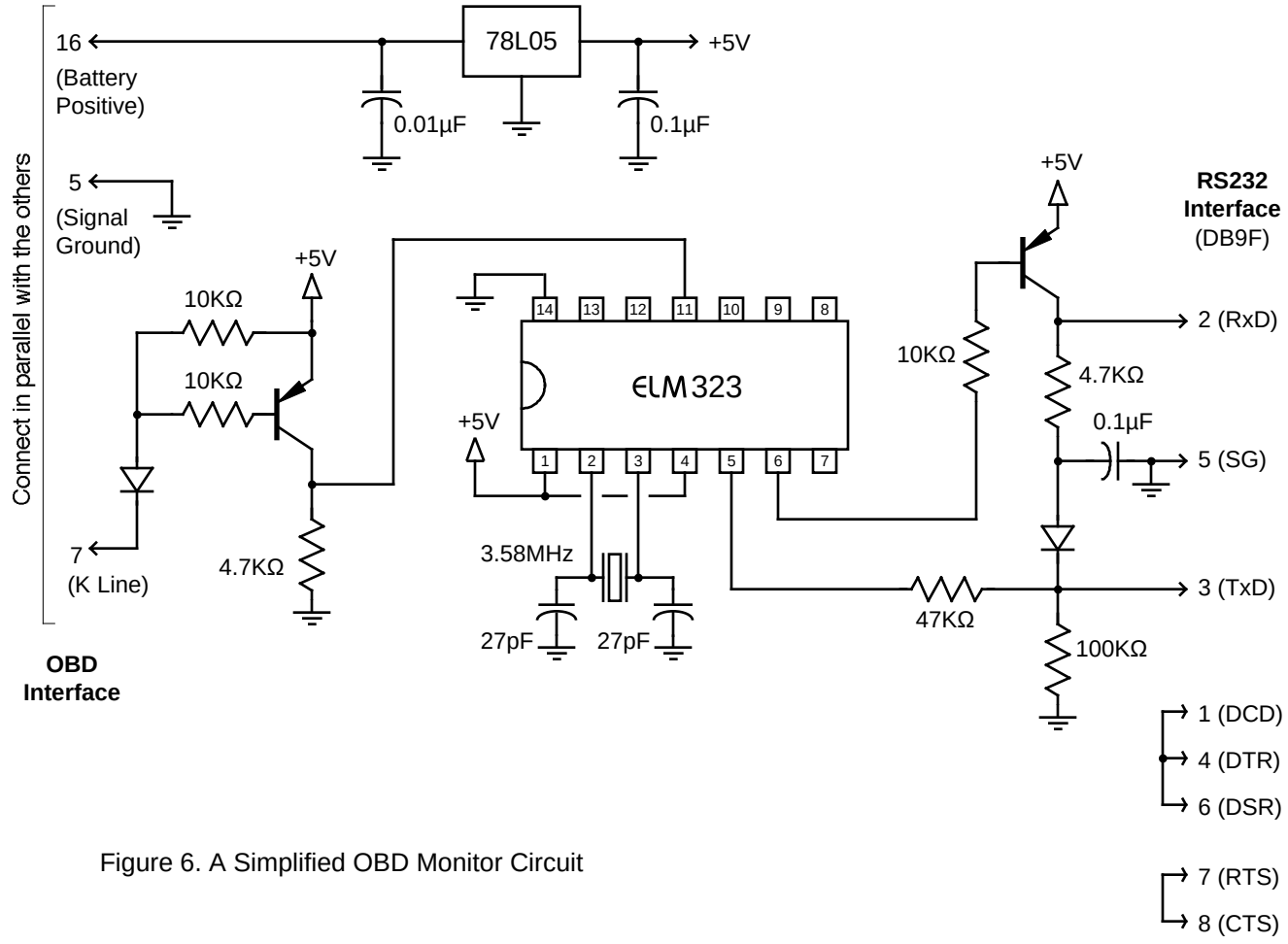


Figure 6. A Simplified OBD Monitor Circuit