ELM633
19200bps miniLIN Monitor

Description

LIN or 'Local Interconnect Network' is a low cost and relatively simple networking system that is used predominantly in the automotive world. Recently, it has been gaining in popularity with proposals to use it in major appliances as well. For more information, visit the LIN web site (http://www.lin-subbus.de/).

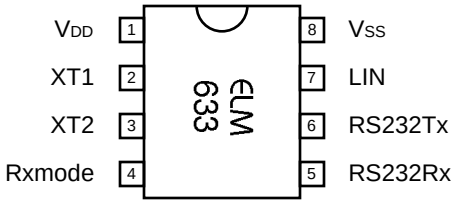
The ELM633 is a monitoring device designed for troubleshooting LIN bus systems. It is capable of continually monitoring a LIN network, translating the LIN messages to standard ASCII characters, and re-transmitting them to an RS232 system (personal computer or PDA) for display and possibly for analysis. The synchronizing, data formatting, and checksum calculations are all done for you by the ELM633.

The LIN specification has recently been updated to revision 2.0, incorporating several improvements. The ELM633 has been updated to be compatible with this new specification, as well as the previous ones.

Features

- Low power CMOS design
- Crystal controlled for accuracy
- Standard ASCII character output
- Special power-on monitor mode
- Permanently set to a 19200bps LIN rate
- High speed (57600 baud) RS232 interface
- Works with LIN1.x and LIN2.0

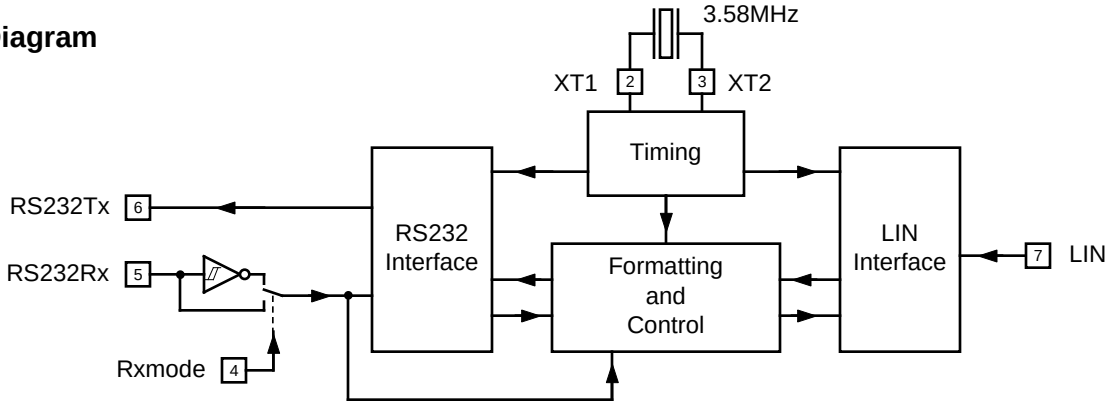
Connection Diagram
PDIP and SOIC
(top view)



Applications

- LIN logic probes
- Diagnostic PC interfaces
- Instruction triggered (breakpoint) devices
- Educational or training devices

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 be connected from each of these pins to the circuit common (Vss).

Rxmode (pin 4)

This input is used to control the inverting of the signal at the RS232Rx input (pin 5), allowing some flexibility as to how the RS232 is connected to it. Many experimenters will prefer to use only a single resistor between the RS232 interface and pin 5 to minimize costs. In that case, pin 4 need only be tied to common (V_{SS}), and the internal logic will invert the polarity of the signal for you.

Other users may prefer to use one of the standard (inverting) interface circuits such as the MAX232 series, or the SN75189/MC1489 type of IC. In these cases, the internal inversion is not required, and should be disabled by connecting the Rxmode input to a high level (V_{DD}). The interface IC can then be directly connected to the RS232Rx input (pin 5).

RS232Rx (pin 5)

A computer's RS232 transmit signal is connected to this pin, either through a resistor or through an interface IC, as discussed under Rxmode (pin 4). Refer to the Example Applications section for some typical connections.

A special power-on monitoring mode is entered if this input is found to be at an active level (as determined by pin 4) for the entire startup period. In this mode, the ELM633 will display the ID string, but instead of issuing a prompt character, it will immediately execute an AT MA command, reporting on all LIN bus activity.

RS232Tx (pin 6)

This is the RS232 transmit or data output pin. While at rest (no data is being sent) this pin will output a high level (V_{DD}), which is compatible with most interface ICs. It has sufficient current drive to allow interfacing using only a transistor (if desired).

LIN (pin 7)

This is the active high LIN signal input. The signal from the LIN bus is inverted and buffered, then presented to this pin. Note that this input is limited to voltages from V_{SS} to V_{DD}, so it can not be directly connected to the LIN bus. (See the Example Applications section for a typical interface circuit.)

V_{SS} (pin 8)

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 208 mil SOIC surface mount type of package. To order, add the appropriate suffix to the part number:

300 mil Plastic DIP.....ELM633P 208 mil SOIC..... ELM633SM

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, and 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}		0.8	1.4	mA	
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
RS232Rx pin input current	-0.5		+0.5	mA	see note 3
RS232 baud rate		57600		baud	see note 4

Notes:

1. This integrated circuit is produced with a Microchip Technology Inc.'s PIC12C5XX 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. This specification represents the current flowing through the internal protection diodes when the voltage connected to the RS232Rx input (through a current limiting resistance) is greater than V_{DD} or less than V_{SS}. Currents quoted are the maximum that should be allowed to flow continuously.
4. Nominal data transfer rate when the recommended 3.58MHz crystal is used as the frequency reference. Data is transferred to and from the ELM633 with 8 data bits, no parity, and 1 stop bit (8 N 1).

Communicating with the ELM633

The ELM633 relies on a standard RS232 serial connection to communicate with the user. The data rate is not adjustable and is set at 57600 baud, with 8 data bits, no parity bit, 1 stop bit, and no handshaking (often referred to as 57600 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 this connection (and the linefeed mode that you have chosen). No special software is required to 'talk' to the IC – a standard terminal program is all that is needed.

Once it has been properly connected and powered, the ELM633 will send the message:

```
ELM633 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. The '>' character displayed above is the ELM633's prompt character, which shows that the device is in its idle state, ready to receive characters on the RS232 port.

All messages that are sent to the ELM633 must begin with the character 'A' followed by the character 'T', and must be terminated with a carriage return character. No action is taken – commands are not checked for errors, nor are they acted upon – until this terminating carriage return is received. The one exception is when a command is interrupted for some reason, and no carriage return appears. In this case, an internal timer will automatically abort the incomplete message after about 15 seconds, and the ELM633 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 ELM633 (syntax errors) will always be signalled by this same single question mark ('?'). When this occurs, it is usually due to a spelling mistake, so you often only need to repeat the input, typing more carefully.

Occasionally, errors occur if the ELM633 is busy processing LIN messages when an RS232 command message begins. In these cases, the first character of the RS232 command message will always be missed by the IC, so the remaining characters will appear to be incorrect. You should always interrupt the monitoring process with a single character (it doesn't matter which one, as it will be ignored), then wait for the prompt character (">") to appear before sending any more. This ensures that the ELM633 is ready to receive commands.

For convenience, the ELM633 has been designed to ignore spaces and control characters in the input, so if you prefer to add spaces, tabs, etc., to improve readability, then go ahead. Also, the ELM633 is not case-sensitive, so 'ATZ' is equivalent to 'atz' and to 'AtZ', which may be helpful for some people.

AT Commands

The ELM633 is controlled with short commands that all begin with the two characters 'AT' (which is short for ATtention). These two characters serve no purpose other than to help add validity to the characters that follow. Modem manufacturers have used this same technique for years, and it has become customary to refer to commands that begin with these characters as 'AT Commands'.

The ELM633 accepts several different AT commands, but only one at a time (it cannot process multiple commands on one line, as most modems can). Each is executed only upon receipt of the terminating carriage return character. Several commands do not have a visible response (AT D for example), so completion of those commands will be acknowledged by the printing of the characters 'OK'.

Monitoring of the LIN bus can generally begin without requiring the use of any AT commands, as the factory default settings should be appropriate for most situations. Occasionally, however, the user may wish to customize settings, such as turning the character echo off, and in these cases, AT commands must be used.

To perform the desired AT command, simply send the characters AT followed by the appropriate characters from the following list. For example, to turn character echoing off, simply send AT E0 followed by a return character. To turn it back on, send AT E1.

The following is a summary of the commands that are recognized by the current version of the ELM633.

AT Commands (continued)

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.

BR [display the Baud Rate]

This command asks the IC for the LIN baud rate it is currently operating at. Th ELM633 only supports one baud rate, so this command will always return with the response '19200'.

D [set all to Defaults]

This command resets all of the options to their default values which were set at the last power-up, or manual reset command. This usually only affects the echo and linefeed modes, but if the RS232Rx pin was at an active level during the last reset, the IC will immediately enter the Monitor All (AT MA) mode after this command is issued.

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

These commands control whether or not characters received on the RS232 port are re-transmitted (or echoed) back to the host computer. To reduce traffic on the RS232 bus, users may wish to turn echoing off by issuing AT E0. The default is E1 (echo on).

I [Identify yourself]

Issuing this command causes the chip to identify itself, by printing the startup product ID string (which is currently 'ELM633 v2.0'). 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 option of transmitting a linefeed character after each carriage return character is controlled by these commands. If an AT L1 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 serve no real purpose in such a case). The default setting is L1 (linefeed characters are sent after every carriage return).

MA [Monitor All]

This command causes the IC to immediately begin monitoring the LIN bus for messages, displaying all that it finds. It can be interrupted at any time by activity on the RS232 receive input.

MR hh [Monitor for a Response to hh]

This is a special version of the AT MA command. Only identifier/command bytes matching the hex characters 'hh' will be displayed, and all others will be ignored. This can be useful if trying to track down a troublesome switch problem, or for triggering another action on a specific input.

Z [reset all]

This combination causes the chip to perform a complete reset, as if power was cycled off and then on again. All settings are returned to their default values, and if the RS232Rx input is not held at an active level for the entire reset time, the chip will be put in the idle state, waiting for characters to arrive on the RS232 bus. Note that the level at the Rxmode input is measured again at this time, so the RS232Rx input may be affected if the level at pin 4 has changed since the initial power-up.

The LIN Standard

The cost and the weight of wiring harnesses in automobiles have been a concern to manufacturers for some time. To reduce both of these, manufacturers have begun adopting network bus structures to allow sharing of common information (and common copper). While several topologies (CAN, etc.) were developed for the high-speed information requirements, these systems were a definite overkill for some applications, particularly when interfacing with humans. Seeing this need, the LIN Consortium was formed, and the Local Interconnect Network standard was developed.

The LIN standard uses a bit serial protocol that is in most respects identical to the one used for personal computers. This keeps costs to a minimum as almost 'off the shelf' parts can be employed, and the learning time for developers is quite short, as the techniques are familiar. The one difference between standard RS232 and the LIN protocol is that the LIN interface uses a synchronizing signal.

The LIN synchronizing signal consists of a period of at least 13 consecutive bit times that are all in the active ('0') state. This would normally never occur in an RS232 system as a start bit and 8 data bits could only give a maximum of 9 active bits. The 13 bit long start signal (known as the 'Synch Break' signal) is always initiated by a bus master processor, to signal that a data transfer is about to follow. Once the Synch Break occurs, the message bytes are all sent in the same manner as for standard RS232.

Each message that is sent is split into two parts, the header and the response. The bus master always generates the header, while the response can be provided by either the master or by a slave processor on the bus (depending on whether the master is trying to send information or obtain it).

The first byte of the header is known as the 'Synch Byte', and it is always the byte value 55. That value was chosen because it creates a pattern of alternating '1's and '0's that can be used by the slave devices to perform an internal timing calibration. This allows inexpensive RC oscillators to be used for the slave processors, reducing costs.

Following the Synch Byte, the master will always send an Identifier Byte which describes the information which is required, or that which is to follow. One can think of it as the command byte.

The response field occurs after the ID byte, and will generally consist of two, four, or eight data bytes, followed by a single checksum byte. However, the standard does allow certain commands to initiate messages of arbitrary length so be aware of this when developing software to process the LIN data.

The following figure may prove helpful in visualizing a typical LIN message:

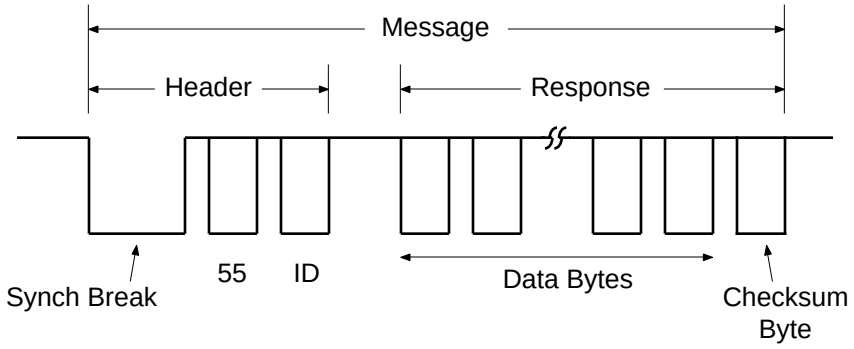


Figure 1. The LIN Message Structure

Monitoring the LIN Bus

Data bytes that appear on the LIN bus can assume values from 0 to 255. These cannot be displayed using a PC terminal program, however, since many of these values are not printable characters. In order to make them readable, the ELM633 re-formats every byte as a pair of hexadecimal digits, using standard ASCII characters. A typical request made of the ELM633 would appear as:

```
>AT MA
49: 6C 8F 04
```

The AT MA is the user's request to 'monitor all', while the 49: 6C 8F 04 is what the ELM633 found on the LIN bus. Note that the initial Synch Byte is always received, but is never displayed (it is always 55).

The identifier byte (49 in this example) always appears first, and is separated from the data bytes by a colon character (":"), while the pairs of hexadecimal digits following represent the data bytes that were received. The final pair of digits on each line is the checksum byte (04 in this case).

Should the checksum byte not match the value calculated internally by the ELM633, the error will be flagged by printing a single question mark at the end of the line. For an example, if the slave driver in the above case was weak, allowing an extra '1' to appear in the first data byte of this example, the output might typically look like:

```
>AT MA
49: 6C 8F 04
49: 7C 8F 04?
```

The question mark at the end of the second response line alerts you to the fact that an error is present, but the position of the error cannot be determined from this information – you will only be able to say that it is somewhere in the response. Another type of error that could occur is when a slave fails to respond to the master. In this case, you would typically see only the identifier, followed by a single question mark:

```
>AT MA
49:?
```

The question mark gets printed in this case because 49 is not a valid checksum byte (all bytes must add up to FF).

Often, this would be followed by a timeout so the output could appear as:

```
>AT MA
49:
[T]
```

As you experiment, you will likely find that timeouts are a very common occurence. They simply mean that there has been no activity for some time, often due to the system going into a low-power sleep mode. Error messages are described further in the next section.

This data is typical of what might be experienced in many systems. The current version of the LIN standard (2.0) allows for the possibility of an arbitrary number of data bytes however, so while the ELM633 has no limitations on what it can display, the user should be prepared for this possibility. This could involve simply allocating enough buffer space, processing the data faster than it arrives, or by simply ignoring lines that are longer than a predetermined length.

The ELM633 also supports one other type of monitoring command, which is useful when you know the specific responses that you wish to view. It is the AT MR command, which requests that only specific responses to ID bytes be displayed, while ignoring all others. For instance, a 'monitor all' command might have resulted in:

```
>AT MA
D3: C0 00 3F
49: 6C 8F 04
92: D4 00 2B
D3: C0 00 3F
.
:
```

Only a portion of the data stream is shown, as it can typically be quite lengthy. However, if the user was only interested in responses that began with 92 for instance, they need only issue an AT MR 92 command to filter the information for them. The above would have looked like:

```
>AT MR 92
92: D4 00 2B
```

This command is often helpful when trying to diagnose a particular problem.

Error Messages

There are only a few error messages that the ELM633 will generate in response to data problems. They are kept short so that their sending does not interfere with the LIN data monitoring. Here is a brief description of each:

[S] [Sync error]

If the ELM633 is unable to synchronize to the LIN data stream, or if the received data bytes seem to be of incorrect length, a synchronizing error occurs. This might be for several reasons – if the LIN bus rate is something other than 19200bps, if the data stream is not of the LIN format and sync breaks or bytes could not be found, or possibly if something occurred during data transfer and the stop bits did not occur when expected. The ELM633 will report this error, and immediately try to re-synchronize to the bus (and will continue to do this indefinitely, or at least until interrupted by the user).

[T] [Timeout error]

This message is printed if there have been no recognizable messages for at least 25000 bit times (referred to as a 'Bus Idle Time-Out' in the LIN specification). This timeout normally occurs when the LIN system has been put to sleep.

When a timeout occurs, the ELM633 will continue to monitor the bus, and will resume resending the LIN data when it appears again.

? [General error]

This is the standard response for a general error, often due to a misunderstood AT command received on the RS232 bus (usually due to a typing mistake). It can also occur if the checksum byte that was received was not correct for either a classic (LIN1.3) or an enhanced (LIN2.0) system.

Example Applications

Figures 2, 3 and 4 on the following pages show various ways that the ELM633 might typically be connected to monitor a LIN system. They show a full implementation; a less expensive, but just as capable one; and finally, a minimal circuit that can essentially be built into a cable to automatically translate LIN data to ASCII RS232 data.

The LIN standard does not specify what type of connector is to be used, but it does specify that the physical interface will essentially be an enhanced implementation of the ISO 9141 standard currently used in automobiles. It provides a three wire connection, with power, data, and common. You will have to choose a connector (or clips) as appropriate for your application.

Figure 2 on the following page is a complete implementation, which uses a standard RS232 interface IC to simplify the connections to the RS232 bus. This is necessary because the ELM633 logic levels are not compatible with RS232 levels. Note that pin 4 must be set to a logic high level in this case to disable the internal inverter. One way of remembering the correct level to set pin 4 to is to consider the logic level that pin 5 (RS232Rx) normally rests at, and set pin 4 to that same level. The RS232 pin connections shown are for a 25 pin female connector, as would often be found on an external modem. If you are using a 9 pin connector to match a cable from your computer, the connections should be changed to 2(RxD), 5(SG) and 3(TxD).

Since the LIN standard provides for a (nominal) 12V supply at the connector, that convenient source has been used to power the ELM633 circuit. This voltage is reduced to the required 5 volts by the 78L05

Example Applications (continued)

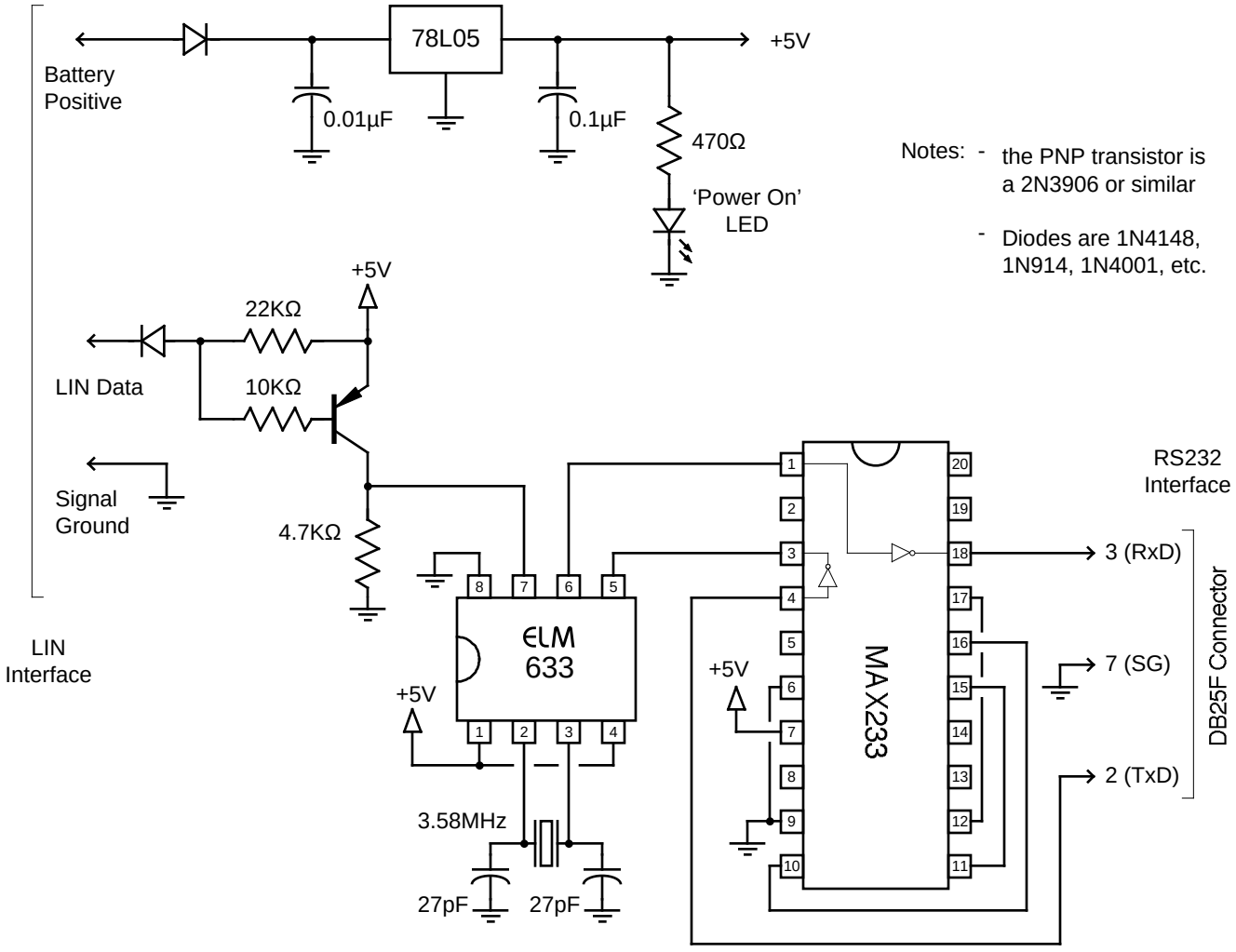


Figure 2. LIN to RS232 Interface

regulator shown, and filtered with the capacitors. An LED provides visual feedback to show that the supply is available.

The PNP transistor that is shown connected to the LIN input (pin 7) serves mainly to level-shift the input signal, and to invert it. A typical CMOS input will switch state at about half the supply level (2.5V), which might cause problems with noise in an automotive environment. By providing a transistor buffer, the input threshold is effectively raised to about 4V, increasing the noise immunity while adding amplification, signal clamping, and inversion. Note the use of the diode on the input. This prevents possibly damaging backfeeds from entering the circuit, protecting the transistor and the other components.

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

The circuit of Figure 3 provides an alternative to the previous one. It is just as functional and does not

Example Applications (continued)

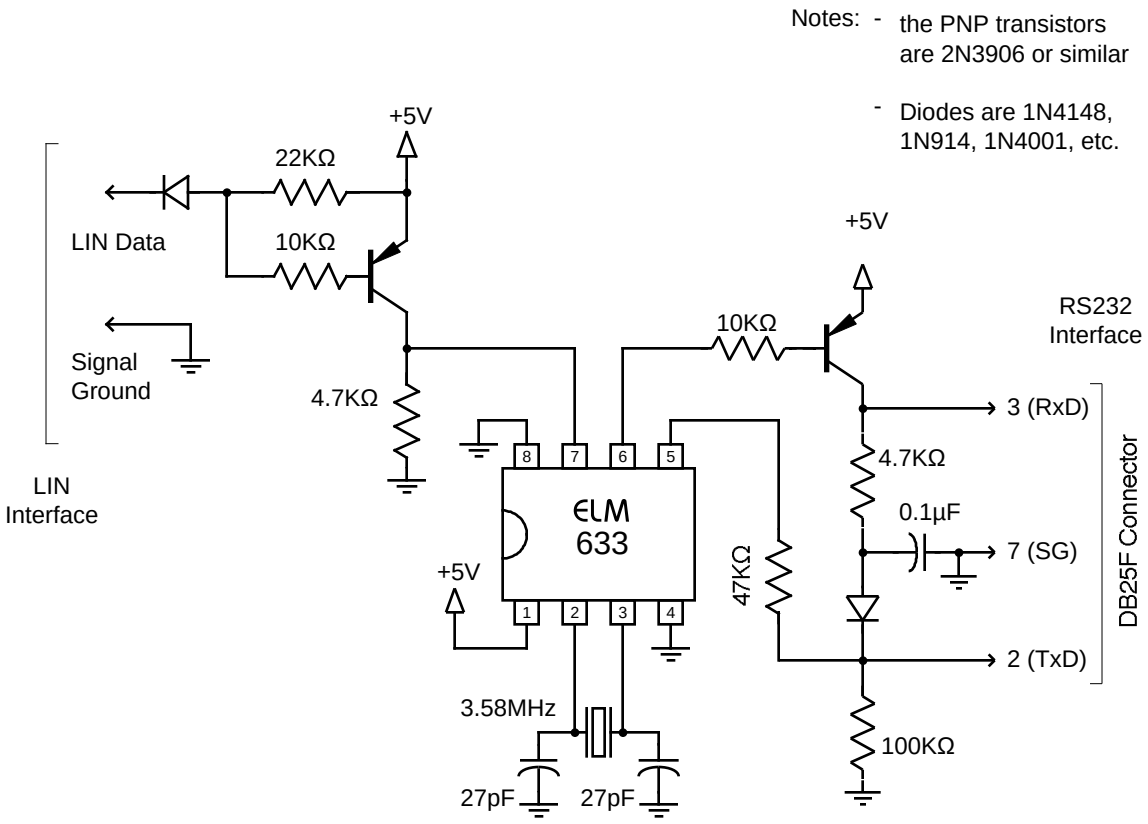


Figure 3. An Alternative LIN to RS232 Interface

require a MAX233, using a few discrete components instead. The circuit uses a diode and a capacitor to 'steal' a negative voltage from the host computer's transmit line, and uses that in turn to provide a bipolar swing for the ELM633's transmit. The transistor on pin 6 simply provides the necessary data inversion, while protecting the rest of the circuit from the negative voltage.

For this circuit, 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 ELM633. A single 100KΩ resistor is also shown in this circuit so that pin 5 is not left floating if the computer is disconnected. When wiring this circuit, it is advisable to keep the leads around pin 5 as short as possible in order to reduce stray capacitance, since this capacitance will interact with the 47KΩ resistor to degrade the input signal. While this is not generally much of a concern with slower interfaces, the high 57600 baud speed of this circuit means that extra care must be taken. Note that the RS232Rx input will normally be at a low level in this case, since there is no inverting interface IC. To compensate for this, the pin 4 input has been set to a low level.

Although this interface is a very simple one, it is quite effective (and inexpensive).

The final example on the following page shows how to take advantage of the power-on monitor mode.

Example Applications (continued)

This mode is entered into automatically if the RS232Rx input is found to be at an active level for the entire startup time period. This is the period of time that it takes for the ELM633 to stabilize on a power-up or reset, prior to issuing the first 'ELM633 v2.0' message. Since pin 4 is shown connected high, if pin 5 is held low for at least the initial period, the device will immediately enter the AT MA mode, as if a user had sent the command. This is convenient if one wants to monitor all LIN bus data, but does not particularly want to manually have to initiate the monitoring. This would be useful if making a 'smart' adapter cable, for example. Circuit power is from three button cells, which provide 4.5V to the circuit through a momentary action pushbutton switch. The 4.5V is at the lower level of recommended voltages, but is still within specified limits, so works fine.

These examples have hopefully provided enough information to get you started with LIN devices. The LIN standard shows considerable promise for low-cost circuits that can be used in lower speed networks such as for man-machine interfaces. While they are relatively new and not too frequently found at this time, they are beginning to turn up everywhere…

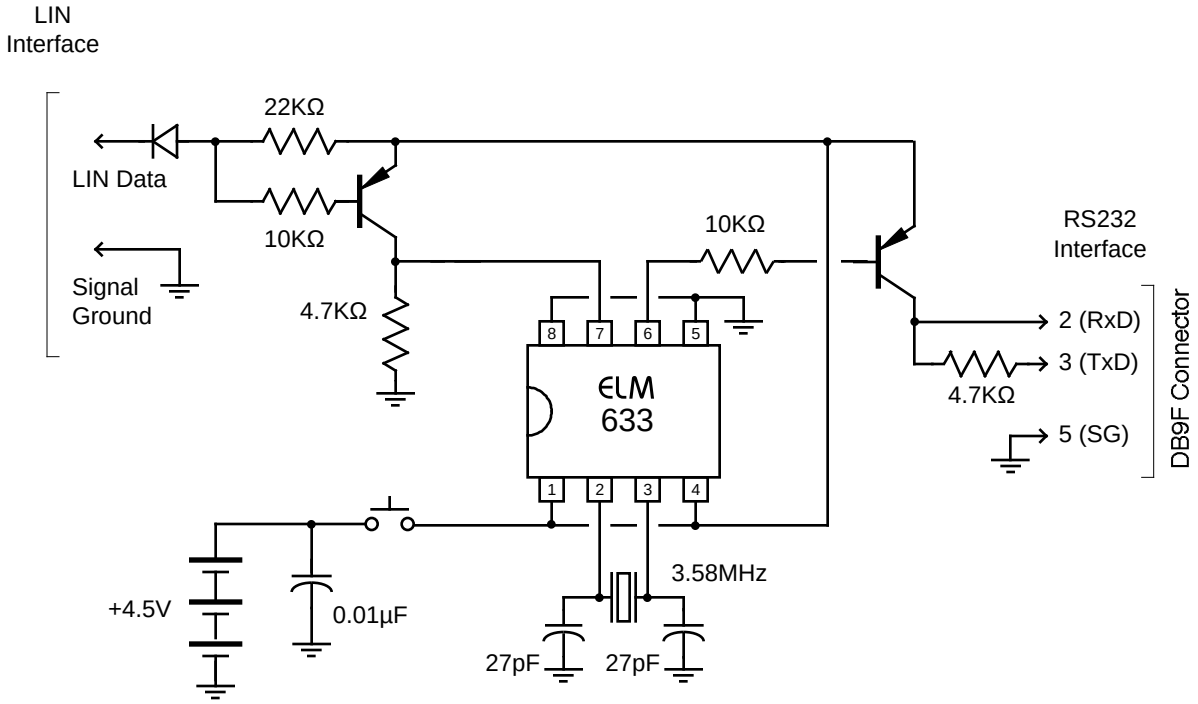


Figure 4. LIN Logic Probe