# ELM624

# Control L to RS232 Interpreter

## Description

The Control L or LANC interface is an industry standard introduced by the Sony Corporation for controlling audio and video devices. It uses a bit serial data format, and requires that the controller be synchronized to the controlled device – something which is difficult to do using standard serial interfaces. The ELM624 is an 8 pin integrated circuit that transparently performs this function for you.

All user interaction with the ELM624 is by standard ASCII characters over an RS232 interface. There is no special formatting required, other than perhaps an understanding of the hexadecimal numbering system, nor is there a need for a powerful PC - virtually any model with a serial port will do.

Since the ELM624 was designed to provide a cost-effective way for people to experiment with the Control L system, many features typically found in commercial devices, such as RS232 handshaking, variable baud rates, extra buffering of signals, etc. have not been implemented. Responses are kept to a minimum as well (eg. a single question mark is returned for a misunderstood command), but the general principles of operation are demonstrated and for many applications, this is all that is required.

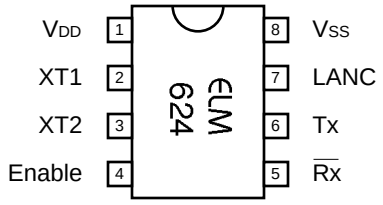
## Applications

- Video editors
- Time-lapse recording controllers
- Programmed control of A/V equipment
- Remote camera controls

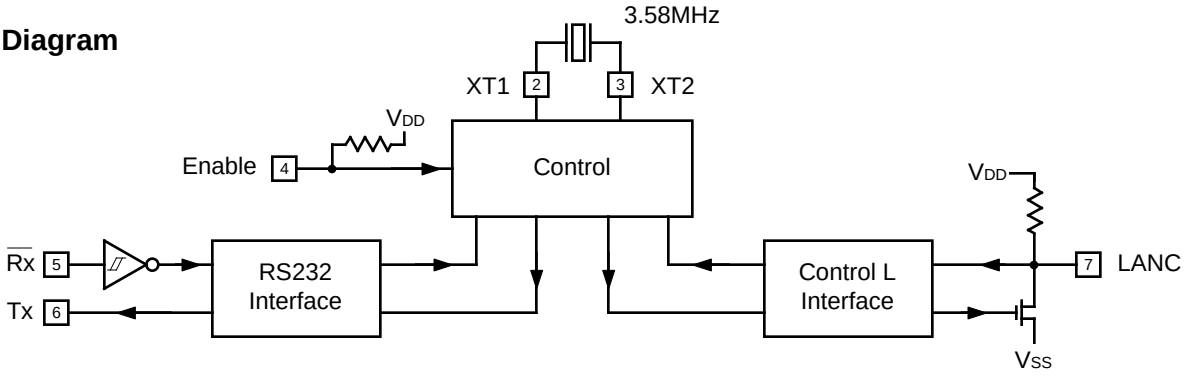
## Features

- Low power CMOS design
- Enable input allows control of multiple devices
- Configurable with standard AT commands
- ASCII output formatted as standard hex digits
- Minimum of external components required
- Most Control L formats supported
- Internal pullup resistors simplify the interface
- Crystal controlled for timing accuracy

Connection Diagram
PDIP and SOIC
(top view)




## Block Diagram



## Control L and LANC

The terms Control L and LANC mean the same thing, and are used interchangeably throughout the following.

## 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 Vss.

Enable (pin 4)

This is an active high input which allows the ELM624 to accept or provide RS232 data. When at a high level (or open-circuited), the ELM624 behaves as in previous versions, continuously providing status updates and responding to RS232 commands. When this input is low, RS232 activity is ignored by the IC, and no status updates are provided.

This pin was formerly used as an active low reset input for v1.x ICs, but has now been discontinued. Users which require the reset function can place a special request for v1.1 ICs, which are functionally the same as that described here except for the lack of an enable pin.

$\overline{Rx}$ (pin5)

The computer's RS232 transmit signal is directly connected to this pin through a single current limiting resistor (typically about 47KΩ). Internal signal inversion and Schmitt trigger waveshaping provide the necessary signal conditioning.

Tx (pin 6)

This is the RS232 data output pin. The signal polarity is compatible with most interface ICs, and the drive is sufficient to allow interfacing using only a single PNP transistor if desired. See the Example Applications section for more details.

LANC (pin 7)

This is the open drain Control L (LANC) interface pin. An internal pullup resistor is provided for a nominal drain load.

$V_{SS}$ (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 200 mil SOIC surface mount type of package. To order, add the appropriate suffix to the part number:

300 mil Plastic DIP..... ELM624P  200 mil SOIC..... ELM624SM

## 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 +7.5V

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 $V_{DD}$ | V | |
| Input high voltage | 0.85 $V_{DD}$ | | $V_{DD}$ | V | |
| Output low voltage | | | 0.6 | V | Current (sink) = 8.7mA |
| Output high voltage | $V_{DD}$ - 0.7 | | | V | Current (source) = 5.4mA |
| Internal pullup resistances (see note 4) | 300<br>20 | 500<br>30 | 600<br>50 | KΩ<br>KΩ | Pin 4 (Enable)<br>Pin 7 (LANC) |
| $\overline{Rx}$ pin input current | -0.5 | | +0.5 | mA | see note 5 |
| RS232 Baud Rate | | 9600 | | baud | see note 6 |
| Enable input setup time | 15 | | | µsec | |

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.
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. The value of the internal pullup resistance is both supply and temperature dependent.
5. This specification represents the current flowing through the protection diodes when applying large voltages to the Rx input (pin 5) through a current limiting resistance. Currents quoted are the maximum continuous.
6. Nominal data transfer rate. Assumes that a 3.58 MHz crystal is used as a frequency reference. Data is transferred to and from the ELM624 with 8 data bits, no parity, and 1 stop bit (8N1).

## Communicating with the ELM624

The ELM624 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 as well. Make sure your software is configured properly for the mode you have chosen.

Properly connected and powered, the ELM624 will initially display the message:

ELM624 v2.0

If you have elected to add linefeeds after line ends, the cursor will now be blinking on the next line. In addition to identifying the version of the IC, receipt of the above string is a convenient way for users to be sure that their connections are correct.

After displaying this initial message, the integrated circuit will be ready to receive commands on the RS232 lines. These commands consist of either 3 or 4 ASCII characters that are terminated with a single carriage return character. Note that there is no prompt character sent by the IC to indicate that it is in the 'receive mode'. (This is done in order to reduce the number of bytes sent and thus maximize the processing time between the bursts of data bytes.)

RS232 commands that are sent to the ELM624 can either be for internally configuring the IC, or for passing on to the LANC device. The internal commands are distinguished from the others by always beginning with the two characters 'AT', while the Control L commands will always be a series of four hexidecimal digits. This allows the ELM624 to quickly determine where commands are to be directed. See the following sections for further details.

In sending commands from a terminal program, some users may wish to insert extra characters (spaces or tabs) to improve the readability of their input. This is permitted, as the ELM624 simply ignores these characters. Also for convenience, the ELM624 is not case-sensitive, so 'ATZ' is equivalent to 'atz', and to 'AtZ'. This allows you to type commands in whatever way you prefer.

All commands must be terminated with a carriage return character (hex '0D') before they will be acted upon. A word of caution, though – if a linefeed character is sent by the computer after the terminating carriage return, it could be interpreted as the start of a new command line unless the ELM624 is expecting it. If you cannot turn the linefeeds off in your software, be sure that the ELM624 is set consistently, with linefeed generation turned on (see the ATL1 Command in the next section).

There is a possibility that data strings sent by the computer (or PDA) could be incomplete for one reason or another, and no carriage return is ever received. To avoid problems in these cases, the ELM624 will automatically abort any incomplete command after about 20 seconds. It will then print a single question mark, and wait for another command to be sent.

There are two replies that can normally be expected from the ELM624. Understood and executed 'AT' commands are acknowledged by the ELM624 sending the two characters 'OK', while misunderstood commands (syntax errors) are signalled by a single question mark ('?'). When Control L commands are executed, correct completion is signalled only by the ELM624 transmitting the four status bytes (sent as eight hexadecimal digits). There will be no 'OK' sent in this case.

One last note involves internal priorities. When the Enable input is at a high level, activity on the RS232 bus will always interrupt activity on the LANC side. In some cases it will be delayed, however, until the ELM624 completes what it is doing. This is a change from the v1.0 software where all activity was immediately aborted with RS232 activity. If your software had counted on an abort with activity, it will have to be updated.

To expand on this, if there is activity on the RS232 Rx line while the IC is in the middle of transmitting a response, it will always complete sending the line, and will send the terminating carriage return before returning to process the RS232 input. Similarly, if a command is being sent to the Control L device when an input occurs, the command will finish before the ELM624 will accept the RS232 input. In this case, no status reponse will be displayed. During the time that the ELM624 is completing what it is doing, it will not store RS232 characters – it will only note that there has been activity.

When controlling the ELM624 from your keyboard, you cannot choose when to interrupt the ELM624, and in fact it will make little difference. Under computer control however, one should always look for a terminating carriage return (and a linefeed character if enabled) before beginning to send a command. This will ensure that the ELM624 is in an idle period, waiting for input, and is not in fact being interrupted.

## AT Commands

The ELM624 can accept internal configuration commands at any time, in much the same manner that modems do. Any command sent to the ELM624 which begins with the letter ‘A’ followed by the letter ‘T’ is assumed to be an internal configuration (or ‘AT’) command. These commands are executed upon receipt of the terminating carriage return character, and acceptance of such a command is acknowledged by the printing of the characters ‘OK’.

The ELM624’s factory default settings should be appropriate for most applications, but some users may wish to customize their settings, such as turning the character echo off or linefeed generation on by issuing these commands.

The following summarizes the ‘AT’ commands that are recognized by the current version of the ELM624 (commands not supported by the older version 1.0 software are as noted). Note that the character ‘0’ is the number ‘zero’:

ATC0 and ATC1

These commands specify when the Control L status bytes are to be returned on the RS232 bus. With the C1 command, values are only sent when there is a change from the previous four bytes, while with C0 the response bytes are always sent. For most devices, setting C1 will have little noticeable effect, as the LANC responses usually alternate between status and time-code values, so they do continually change. The default is C1, send on change.

ATD0 and ATD1

These commands determine whether the first two Control L words (0 and 1) are to be duplicated in the next two words (2 and 3). If D0 is selected, duplication will not occur, and words 2 and 3 will be sent as zeros. Issuing ATD1 will force the words in bytes 0 and 1 to be duplicated in bytes 2 and 3. The default is D1, duplicate on.

ATE0 and ATE1

These commands control whether characters received on the RS232 port are retransmitted (or echoed) back to the host computer. To reduce traffic on the RS232 bus, and perhaps simplify some computer software, users may wish to turn echoing off by issuing E0. The default is E1, echo on.

ATI

This causes the IC to identify itself. On receiving this command, it will send the power-on identifying message (currently ‘ELM624 v2.0’) and will then return to the ready mode.

ATL0 and ATL1

Whether or not the ELM624 transmits a linefeed character when a carriage return character is sent, is controlled by this option. If an ATL1 is issued, linefeed generation will be turned on, and for ATL0 (the default) it will be off. Users may wish to have this option turned on to improve readability if their terminal program does not offer it, but should leave it off if using computer control (as the extra characters transmitted will only serve to slow the system down). Not supported by version 1.0 software.

ATRn where ‘n’ is a Hex Digit

This sets the LANC command repeat value. Although commands are only sent from the computer to the ELM624 once, they are sent to the Control L device multiple times. While the Control L standard requires that commands must be repeated at least four consecutive times to be valid, the ATR command allows this parameter to be modified for experimentation or non-standard applications. The default value is five (R5), and commands are always sent at least once (an R0 is treated as an R1).

ATZ

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.

AT?

This is a useful way to quickly obtain the current status bytes from the LANC device without having to wait while the ELM624 issues a command. This can be of advantage in cases where several devices are interconnected, and you wish to sequentially poll each device to determine its status. Not supported by version 1.0 software.

## Control L Data Transfer

In the Control L system, data is sent in groups of eight bytes at a time, with all groups repeating at either a 50Hz or a 60Hz rate depending on where the system was purchased. Information flows both ways using these eight bytes, with the first four being sent to the device being controlled, and the remaining four being responses from that device. Except for voltage level differences, each byte is transferred in a manner which is identical to the RS232 standard that is common throughout the computer industry. If the signal were inverted, it could likely be input directly to a computer's serial port, and read as 9600 baud data. The data could not be displayed directly by a terminal program however, as it would be raw data and would have to be converted to an ASCII format for display on the screen.

While receiving data is relatively easy, the sending of data to the controlled device is not. All transfers must be sent in synchronism to the eight pulses which are generated by the controlled device, and this must be done with only a few µsec of error. Responding to a real-time signal such as this is very difficult to do with many modern computer systems due to the complexity of their operating systems. The ELM624 can handle this easily however, as it is a dedicated device. It receives the data from the computer or PDA, checks for errors, then synchronizes to the time signals and sends the data out. Data received in the latter half of the frame is reformatted and sent to the controlling computer during the next frame. There is an idle period between frames (typically about 5msec) when very little activity actually occurs, so the IC uses this to prepare for the next frame, and to monitor the RS232 Rx line for any new commands.

Occasionally, the ELM624 will be asked to send a Control L signal on the bus, and it will be unable to sense any synchronizing pulses being sent by the controlled device. If the device is not providing these pulses for any reason, the ELM624 will simply display 'NO SYNC', and will return to the ready mode awaiting another command. If this error does occur, check your connecting cables and power supplies.

The figure below is representative of a typical frame of data that is sent on the Control L or LANC bus. Commands to the device (from the ELM624) are sent during the first half of the frame, while the second half (the last four bytes) are for feedback from the controlled device. Often, four bytes are not enough to send all of the information, so data is multiplexed over several frames. An example of this is the time codes which will be discussed in the next section.

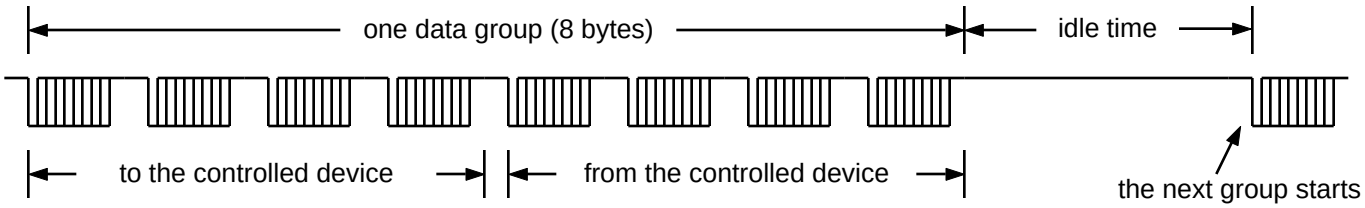



Figure 1. Control L data grouping

## Talking to Your LANC Device

Commands to the controlled device are normally only two bytes long, but are repeated by the ELM624 in the third and fourth positions when transmitted. This results in the four byte packet mentioned previously, without the user having to manually perform the formatting necessary.

The two byte commands are required to be sent from the controlling computer to the ELM624 as four ASCII characters, each representing one of the four (hexadecimal) nibbles of the two bytes. When four valid hexidecimal digits and a carriage return character have been received, the IC will assume that the characters are a valid command meant for sending to the controlled device. It will combine the first two characters to form the first byte, and the third and fourth characters to form the second byte, will synchronize to the incoming signal, and will transmit all four bytes. Note that the ELM624 makes no effort to determine whether the codes are valid commands.

After sending the command, all responses

## Talking to Your LANC Device (cont'd)

received are reported back to the RS232 port using standard ASCII characters. Control characters are never sent – the hexadecimal digit 'A' for example, is transmitted as the decimal value 65 (the letter 'A'), and not as 10 (a linefeed character).

In the Control L standard, the first control byte sent usually designates the device being spoken to. The first nibble is often the type of device (eg. 1 for VTR, or 2 for camera), and the second nibble is a unique identifier for that device (ie. the device number). These correspond to the first and second hex digits sent to the ELM624, respectively. The second control byte (hex digits 3 and 4) is the actual command for the device.

As an example, the following is a typical four character command string which might be sent to the ELM624:

1 0 3 4

It can be more easily interpreted if one separates the first few characters, and reads them as follows:

| 1 | 0 | 3 4 |
|---|---|---|
| VTR | #0 | Play |

That is 'VTR #0 Play' (as the code for play is 34). Similarly, 2130 would usually mean 'Camera #1 Stop'. Note that there does not seem to be a completely standard set of commands, so you will have to check with your manufacturer to be sure of which ones are supported by your device. Some typical control codes are:

| | | | |
|---|---|---|---|
| 30 | Stop | 36 | Rewind |
| 32 | Pause | 38 | Fast Forward |
| 34 | Play | 3A | Record |
| 2C | Eject | 8C | Counter Reset |

On receiving a valid command, the controlled device will respond with a sequence of 4 bytes which the ELM624 will resend as a series of 8 hexidecimal digits. Although the addition of space characters could make this data more readable, there is not sufficient time to do this with 60Hz systems, without losing data. The initial byte (the first two digits) will always be a status byte, having typical values as follows:

| | | | |
|---|---|---|---|
| 01 | No Tape | 02 | Stopped |
| 03 | Fast Forwarding | 04 | Recording |
| 83 | Rewinding | 06 | Playing |
| 72 | Stopped - tape beginning | | |

The next nibble (third digit) is normally used to identify the meaning of the final two bytes. For example, if the digit is a 1, it generally means that the final two bytes are status bytes, while if it is a 3 or 4, they will be time codes. A 3 will indicate the bytes are seconds and minutes, while a 4 means they are hours and days.

The fourth hex digit (least significant nibble of the second byte) generally gives various system status information such as:

b3- Counter memory on
b2- Low battery
b1- Record tab status
b0- Invalid command received

Often it is easier to follow an actual example to see how the codes look. Assume that the command 1034 (VTR#0 Play) has just been sent to the ELM624. It will format the data, transmit it the number of times that is specified by the ATRn command, then begin reading and sending status bytes back to the controlling computer. The first few lines of the response could typically look like:

0 2 1 2 2 4 5 0
0 2 3 2 4 9 1 1
0 2 4 2 0 1 0 0

Many, many more bytes would actually be sent by the ELM624, as each successive line differs from the previous, and the IC always transmits a line when it differs from the previous one. Sending a single space character to the ELM624 will stop the endless stream of data and place the integrated circuit into the ready state, able to receive more commands. Almost any character will serve to stop the data stream, but the space character is the most convenient, and the most likely to be ignored by other devices connected to the system. This could be a concern if operating multiple devices in systems like that shown in the Example Applications section. If you are using an older computer system, there may seem to be a delay between the typing of the space character and the stopping of the 'endless' display on the screen. This is not due to lack of response on the ELM624's part, as it always responds within 20 msec. It is most probably that your display is not fast enough to keep up with the data as it is received, so be patient.

Depending on when your 'stop' character is actually received, you may or may not see it echoed

## Talking to Your LANC Device (cont'd)

onto the screen. This is because if it occurs during the eight byte data frame, it will be noted by the system but not stored or echoed, while if it occurs during the inter-frame idle time, it will be seen and echoed properly. There is also a very small time during the transition between these two that part of the byte may be received so it could be interpreted incorrectly, causing an error. This is unfortunate, but unavoidable when operating manually from a terminal program. To avoid these problems when under computer control, one should always wait until the trailing carriage return of a status line is received, before beginning to send a command. This way, one can be sure that the ELM624 is between frames, and that there is about 5 msec available in which the RS232 command can begin without interrupting any other activity.

Returning to the previous discussion, let us analyze the response. The first line above starts with the byte '02', which shows that the device is currently stopped. The next nibble '1' says that the last four bytes will provide status information, while the fourth nibble, the '2', shows that recording is not allowed due to the tab (b1 is set). The status bytes (24 and 50) provide several details such as record switch on (2), no PCM (4), and mono sound (5). Each controlled device is likely to have its own unique interpretation of these bits.

The second line shows that the device is still stopped (02), that the following information will be ss:mm (3), and that recording is still not allowed due to the tab (2). The time code bytes show 11 minutes and 49 seconds.

The third line is similar, but states that hh:dd (4) follow. Combining the two time response lines (the '3' and '4' lines), one can deduce that the counter is currently showing 00 days, 01 hours, 11 minutes and 49 seconds. Some counters are simply a linear reading and do not reflect the actual time duration, so make sure of the interpretation when trying this on your device.

After looking through your terminal program's receive buffer, or by issuing an AT? if your buffer is empty, you should see something like this a short time after the previous bytes were reported:

0 6 1 2 2 4 5 8
0 6 3 2 4 9 1 1
0 6 4 2 0 1 0 0

Comparing this to the other response, one can see that the device is now in play mode (06), with the time/counter still reading 00:01:11:49, as it has just begun to play the tape. The 8 in the last position of the first row indicates that there is a 'memory mark' here to show a significant spot on the tape. Don't forget to issue a 1030 to stop the tape.

It is beyond the scope of this document to detail the Control L standard in any more detail, but hopefully this has been enough to get you started. Be sure to check with your manufacturer for specific information on the model(s) that you are using.

## Talking to Many ELM624s - Using the Enable

Controlling many devices is similar to controlling one, except for the work involved in selecting the appropriate ELM624, and deselecting the others. The selecting and deselecting is carried out using the Enable input on each ELM624.

As long as the Enable input is at a high level a few microseconds before you begin to issue a command, that command will be processed and acted upon. This initial time period is known as the setup time and is specified in the Electrical Characteristics section. Once the first command character begins, the logic level at the Enable input can then be set to a low level, without affecting the reception or processing of the entire command. However, responses such as the status bytes, or an 'OK', will not appear if the enable input is found to be low when they would normally be sent.

There is no restriction on how often one uses the Enable input – a typical system with several devices will naturally try to poll each device as often as it can to maintain 'real-time' control. One should keep in mind that commands typically take about 100 msec to complete, however, so attempting to poll a device more often may not be of much benefit.

This type of intelligent enable simplifies the control circuitry in most cases, reducing your design to basically a channel selector, and one ELM624 per channel. The Example Applications section shows one possible way of doing this by controlling three devices, through one RS232 interface.

## Design Considerations

The ELM624 is an experimenter's integrated circuit, so it has a few peculiarities that must be considered in any design.

Foremost, one should note that the LANC pin is connected directly to the LANC or Control L bus. This presents two areas of concern. One, if the controlled device is powered before the ELM624, there will be a backfeed through the (inherent) LANC pin protection diode into the ELM624 circuitry. This can cause unexplained and possibly erratic operation, so always power the ELM624 circuitry before the controlled device. This is likely required anyway, as many devices use the presence of a voltage to determine that a controller has been connected to its LANC port.

The other concern related to the LANC pin is the fact that this is a CMOS integrated circuit, and that 'latch-up' can occur if induced currents from external sources are not controlled. To reduce the possibility of latchup, try to reduce exposure by keeping connecting cables as short as possible, and consider placing a small value resistor (100Ω to 220Ω) in series with this pin. The Example Applications section shows such a resistor connected to the LANC pin.

On the PC side of things, the main consideration with the RS232 interface is the fact that the receive signal is inverted from what would normally be expected. This simplifies the RS232 circuitry, but may cause some confusion. Precautions should also be taken in the circuit design, to allow for the possibility that this input may be left floating due to a disconnected serial cable. Typically, this only requires a large-valued resistor between the RS232 TxD pin and Vss, as shown in the Example Applications section. This is the same interface that is used on our popular OBD II interface ICs, so the information provided on our web site under 'Auto Info' would likely be useful, if you are having trouble.

As a final reminder, one should recognize that the ELM624's RS232 interface does not employ any hand-shaking signals, so the controlling computer must be careful to wait for a complete response before issuing the next command, or else characters could be lost. This isn't too difficult to do if one monitors the data stream for a carriage return character, and optionally a linefeed, before issuing the next command (and making sure you allow time for the stop bit to finish).

## Example Applications

Figure 2 shows how the ELM624 would typically be connected in a minimal circuit. A very basic RS232 interface is provided, and the LANC connection is unbuffered, as shown.

There are two types of Control L or LANC connectors commonly used in the industry. One is the mini-DIN type connector (similar to S-VHS or Apple serial port connectors), and the other is the 2.5mm stereo plug type. Both are shown in Figure 2, so choose the one that is appropriate for your application. As previously mentioned, a 100Ω resistor is installed in series with the LANC line for helping to prevent latch-up, and to some extent for protection of the ELM624 from direct shorts to the supply (most Control L connectors provide DC on one of the pins, which could be a source of damaging energy). If a 2.5mm stereo plug is used, the tip is generally the LANC signal, the small ring is often a DC supply, and the sleeve is the circuit common or $V_{SS}$.

Note that no additional pullup resistance is shown from the LANC pin to $V_{DD}$, as the ELM624's internal resistance is usually sufficient when coupled with pullup resistors in the controlled device. Occasionally this may not be enough to provide a full logic swing, however, and an additional resistor (typically 4.7KΩ) may be needed between pin 7 and $V_{DD}$ (+5V).

Circuit timing is derived from the 3.58MHz reference crystal shown connected to XT1 and XT2 (pins 2 and 3). These crystals are quite common and inexpensive television types that should not be difficult to obtain. Although 27pF capacitors are shown here, you may have to vary these slightly depending on the recommended load capacitance of your crystal.

For connecting to the computer, a simple RS232 interface is used. It is not as elaborate as some, but is sufficient for this purpose. Generally, the RS232 interface is operated in half-duplex mode, with the ELM624 listening to the computer, and then the computer waiting for a response. Since both devices are not likely to be 'talking' at the same time, we are able to 'steal' a bias voltage from the computer for the ELM624 circuitry to use, as shown. A diode provides the return negative from the computer's TxD line that the RxD can use, while a small capacitor provides charge storage. The capacitor size shown should be sufficient in most cases, but if you have trouble with

## Example Applications (continued)

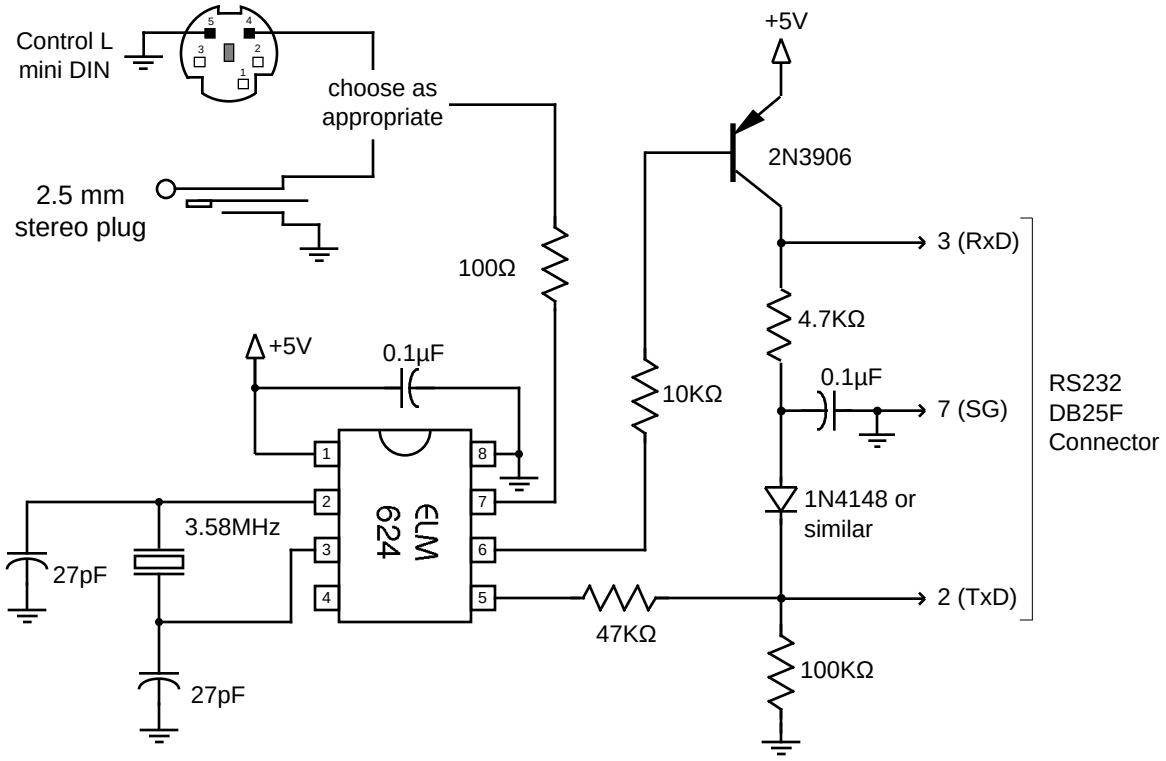



Figure 2. Control L (LANC) to RS232 Interface

characters being echoed, etc., you might want to increase its size to several µF. It is assumed that this circuit will be built into a project box with a standard DB25 female connector, which is commonly used for external modems. The RS232 pin connections shown are appropriate for such a 25 pin connector. If you are using a 9 pin, the connections would be 2(RxD), 5(SG) and 3(TxD).

Figure 3 shows a natural extension of the above, allowing the control of three LANC devices through the one RS232 connection. The ELM624s have been connected with their RS232 pins essentially in parallel (be careful not to directly connect the Tx lines to one another, though), and an ELM621 is used as the device to selectively enable which one you want. The ELM621's ability to respond to ASCII commands to control its port pins works well in this application.

After power up (or an ATZ reset), the ELM621 will configure all three of its port pins as inputs. Since there are 4.7KΩ resistors connected from these pins to $V_{SS}$, the ELM624 Enable inputs will all be at approximately 0V, and the three ELM624s will all be disabled. Note that the ELM621 Tx line is not used, so one will not see the 'ELM621 v1.0...' message on powerup.

To begin talking to the ELM624s, one must first issue the command ATCAOA, which will be received by the ELM621. It will act as commanded, clearing all three pins and setting them all as outputs. After that, the ELM624s can be selectively enabled by sending ATS1, ATS2, or ATS3 as appropriate, or disabled with an ATCA command. For more information on these ELM621 commands, it may be a good idea to download its data sheet.

Once selected, talking to the selected ELM624 occurs as if no ELM621 were present (and since the ELM621's transmit line is not used, it cannot object to us doing so). To control several devices, one simply sends the commands to each sequentially, enabling and disabling the devices as appropriate between the commands.

This opens up even more doors for the creative experimenter, and once again, is only limited by your imagination…

**Example Applications (continued)**

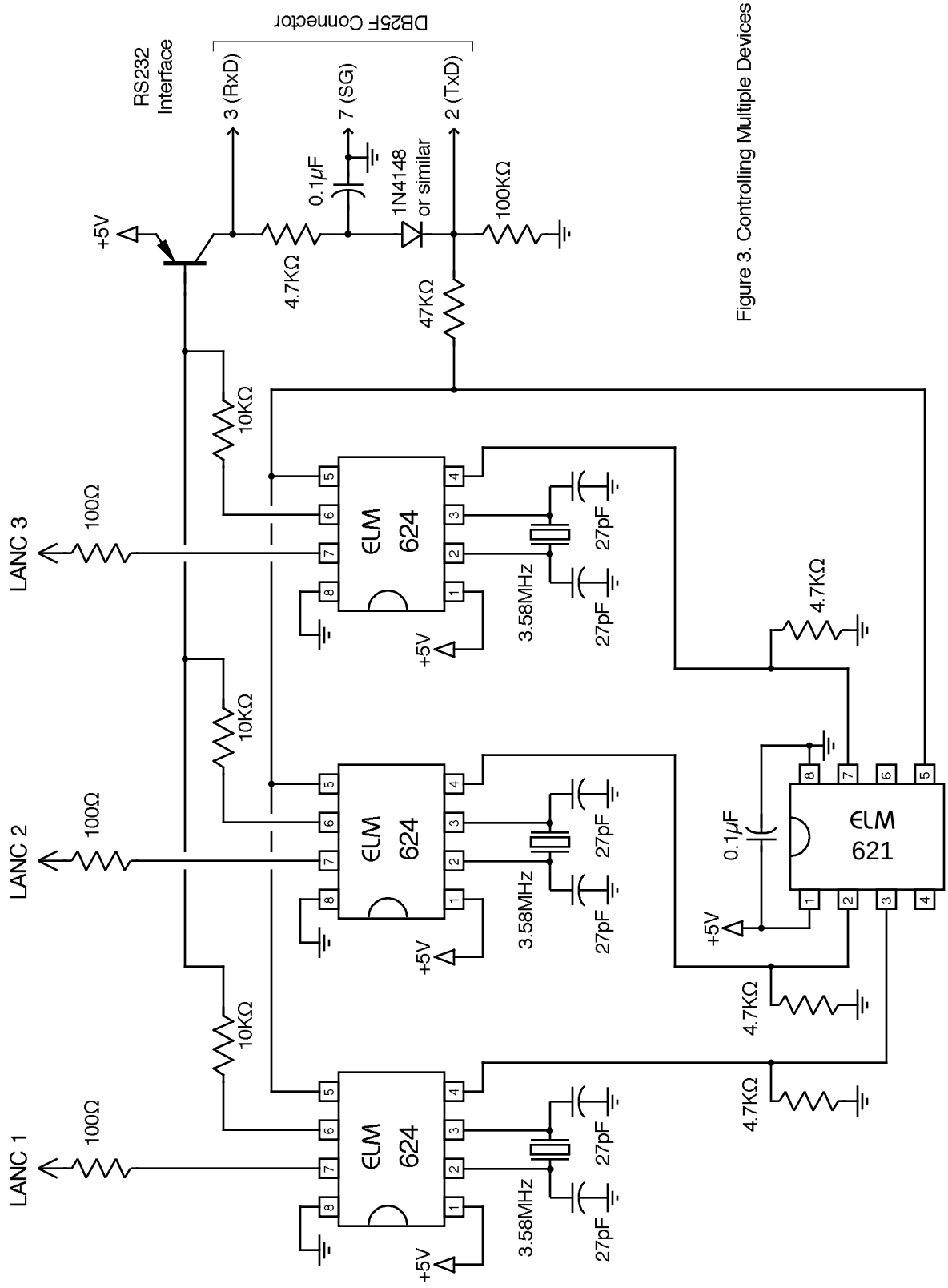



Figure 3. Controlling Multiple Devices