

By: Linx Engineering

Project Engineers: Frank Hammerling, Paul True, Ammon Gomez



Telemetry Link Demo Components



Visual Basic Windows GUI

Introduction

One of the most popular uses for Linx products is to transfer data containing the status or value of a remotely located device to a host. The purpose of this application note is to illustrate how easily a Linx RF module can be applied to create a simplex communication link of this type. The application note focuses on a sample project, which illustrates the use of popular hardware and software to achieve a simple and cost-effective telemetry link. Because of its compact size and ease of use the Linx LC series was selected as the basis for this example.

Please bear in mind that this application note is not intended to detail a finished product. It was developed simply to provide qualified engineers a starting point for their own development journey.

Transmitter-end overview

The transmitter side of the project is self-contained and utilizes the popular PIC 16C711 micro controller from Microchip. It features a

0-5V A/D input with a potentiometer to vary the voltage, a momentary switch, and a NTC thermistor for temperature data.

Receiver-end overview

The receiver-end is intended for direct interface to a PC's serial port from which it cleverly draws its power. All data handling is performed on the PC side using a Visual Basic GUI.

Transmitter-end hardware

The Micro-Controller (μ C) has several analog inputs, where analog signals can be converted to digital data. In this application, two of the PIC's A/D converters are used to convert a signal 1.) from a NTC thermistor and 2.) from a selectable source, either a potentiometer or an analog input 0-5V. This selection is achieved by setting a jumper.

A thermistor changes resistance value with temperature. By applying it in a voltage-divider configuration, a voltage corresponding to temperature is obtained. The potentiometer is connected as a voltage divider as well. It puts a

voltage from 0-5V into the second analog input at the μ C. The two push-buttons lead to separate digital inputs at the μ C. The PIC chip simply reads all its inputs and then formats and transmits the information in a packet.

Receiver-end hardware overview

In many instances it is useful to transfer telemetry information directly to a PC for analysis and display. For this application we chose to interface a LINX LC Receiver module with a PC's RS-232 serial port. For grins and to impress you, the reader, we wanted to make the entire receiver fit into a connector shell. This meant getting creative and deriving power directly from the serial port. To accomplish this, the components derive their power from the signal lines of the serial port.

A PC's serial COM port has 3 output lines: TxD (data out), RTS (request to send), and DTR (data terminal ready). DTR and RTS can be put on both permanent logic high or permanent logic low. Since all lines at the COM port have $\pm 10V$ levels, setting DTR or RTS on either low or high will result in either -10V or +10V. In this application they are both put on +10V and tied together, to avoid a voltage drop due to receiver current. A voltage regulator is used to provide the module with stable 3V. In this configuration the LC receiver's data output will swing between 0V and 3V.

A low-power operational amplifier is used as a comparator to adapt the LC receiver's data output levels to the PC's COM port. Since the COM port's TX data line is not used and carries a permanent -10V, it becomes the negative voltage supply for the op-amp. This allows the op-amp to provide a $\pm 10V$ swing to the PC.

Keep in mind that the sample receiver circuit board is not meant to illustrate optimum RF layout or ground-plane techniques. Indeed, such a small board provides a marginal counterpoise for the 1/4-wave antenna employed. Also, in many instances, emissions from the PC may interfere with receiver operation and it may be necessary to remotely locate the receiver using a cable.

Why is a protocol needed?

It is important to note that Linx modules including the LC do not encode or packetize the data in any manner. This transparency eliminates the issues of variable latency common to traditional radio modems and gives the designer a virtually unlimited number of techniques and protocols that may be employed for such a transmission.

Effective data transfer in any RF link is best accomplished by breaking the information to be sent into small logical packets. This minimizes the impact of errors and simplifies data handling at the receiver. It is also necessary to provide a recognizable pattern to the start and end of data packets so that the packet can be accurately distinguished from noise and the receiver can "frame" the packet correctly. In the sample protocol, framing is accomplished by a start byte.

Since there is always a possibility of bursting errors from interference or changing signal conditions causing corruption of the data packet, it is always wise to employ some form of error checking. Once an error is detected the protocol designer may wish to simply discard the corrupt data or develop a scheme for correcting it. In the sample protocol a simple checksum is used. If the checksum is not valid the data is simply discarded. In our sample application the data is largely redundant; therefore, this simple method works well.

Regardless of which protocol structure is chosen, it is critical to understand the ways in which a wireless environment differs from that of a hardwired. At every point in this system, there are timing and data corruption issues that should be understood and accounted for.

About this applications protocol

For the telemetry link that serves as the basis for this application note a simple yet fairly robust protocol was developed. The programmer chose a packet structure consisting of five 10-bit bytes. The bytes are

transmitted LSB first and contain the following order and information:

Byte 1 is the start byte, which contains a counter value from 0 to 255. This value is incremented with each packet sent. The purpose of this byte is to allow the receiving computer to synchronize so that the beginning of a data packet can be correctly identified. The value is incremented to provide a first line of defense against data corruption.

Byte 2 contains temperature information from the thermal probe. The span from 0-255 is used to represent temperatures from 20-220 F. It is important to recognize that the voltage read by the A/D is compensated for in software to linearize the thermistor's temperature curve.

Byte 3 contains the voltage present at the transmitter's A/D input. The span from 0-255 is used to represent voltages from 0-5 volts DC.

Byte 4 uses the two LSB to represent the status of the transmitter's two switches.

Byte 5 contains the checksum information. In this case the checksum is the 100H (256) minus the sum of all bytes plus an offset. The offset insures that a checksum value will always be present even if all 0's are being sent.

Each packet is separated by a sync space of at least one byte-length. This insures that the UART will always be able to synchronize, with the worst error being the loss of the first packet.

At the transmitter end of the link the PIC microcontroller bit-bangs an output pin to send slightly in excess of 18 packets per second. The actual bytes are sent at 2,400 bps; however, when the sync spaces are taken into account the effective data rate is about 900 bps.

On the receiver end the receive data is transferred directly to the PC's serial port. Once the host PC sees data from the telemetry

receiver it will attempt to synchronize. Generally the UART is not lucky enough to settle on an actual start bit and so the first byte is generally stored in error. The UART will become correctly synchronized with the beginning of the next packet thanks to the sync-byte period previously discussed. This achieves packet-level synchronization that allows the bytes to be stored correctly; however, the beginning data byte must still be found. This is achieved by comparing the value of the first valid byte with the value of the sixth. If the difference is 1 (a result of the counter increment) it can be assumed with high probability that the start packet has been found. The worst-case period for determining this would be 5 byte-lengths.

The incoming bytes are stored as ASCII values and then converted to integers. Depacketizing and byte interpretation is handled via software written in Visual Basic.

In certain instances it may become necessary to use a RS232 extension cable for connecting the receiver part to your PC. This could be by reason of interference with radiating parts of the computer, especially the monitor. In testings with laptop computers, mass interference from the monitor occurred because the receiver device was mounted too close to the monitor.

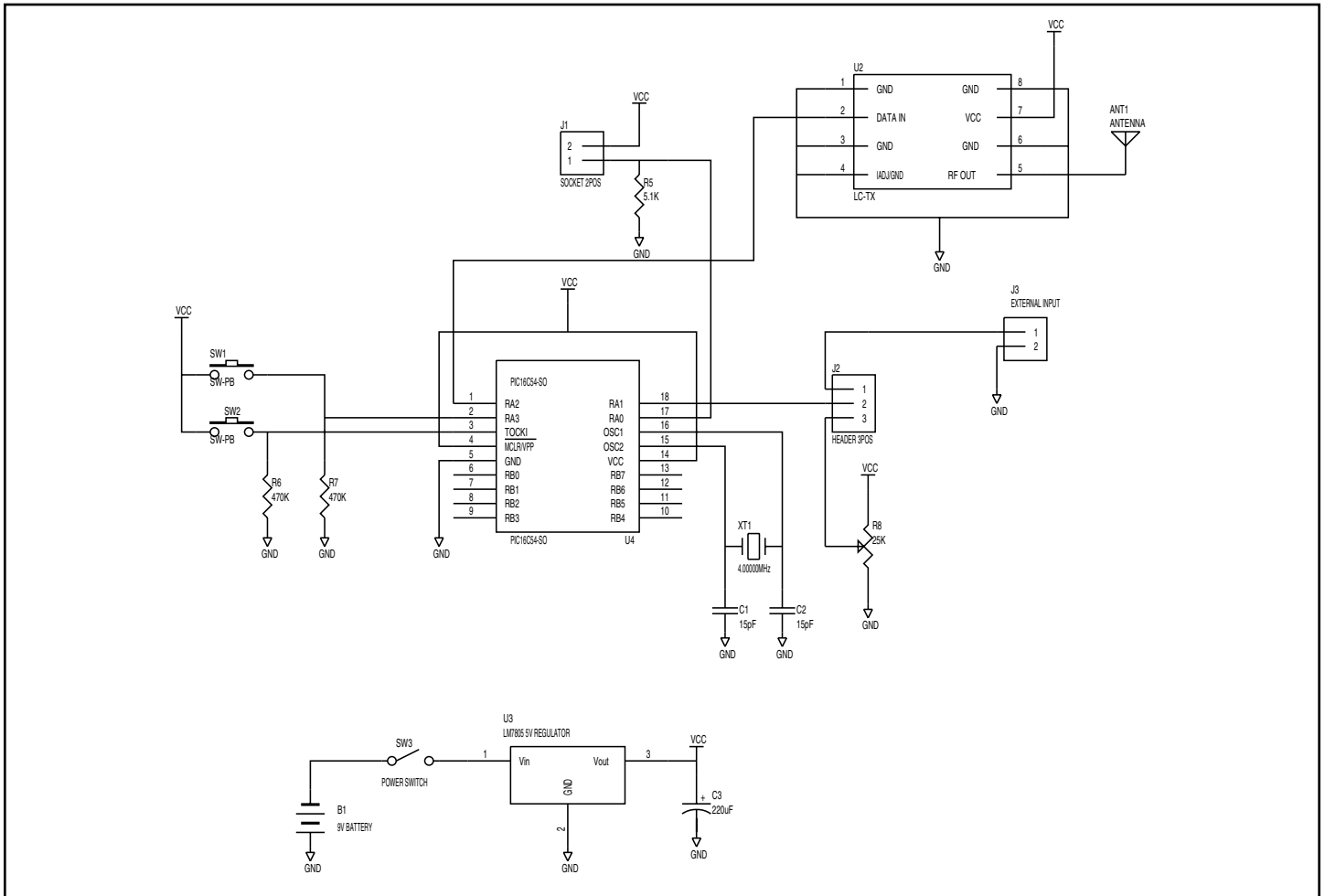
This application note has demonstrated the implementation of a simple serial communication link between a freestanding microcontroller-based device and a PC serial port. The Visual Basic and PIC microcontroller code may be downloaded at:

www.linxtechnologies.com.

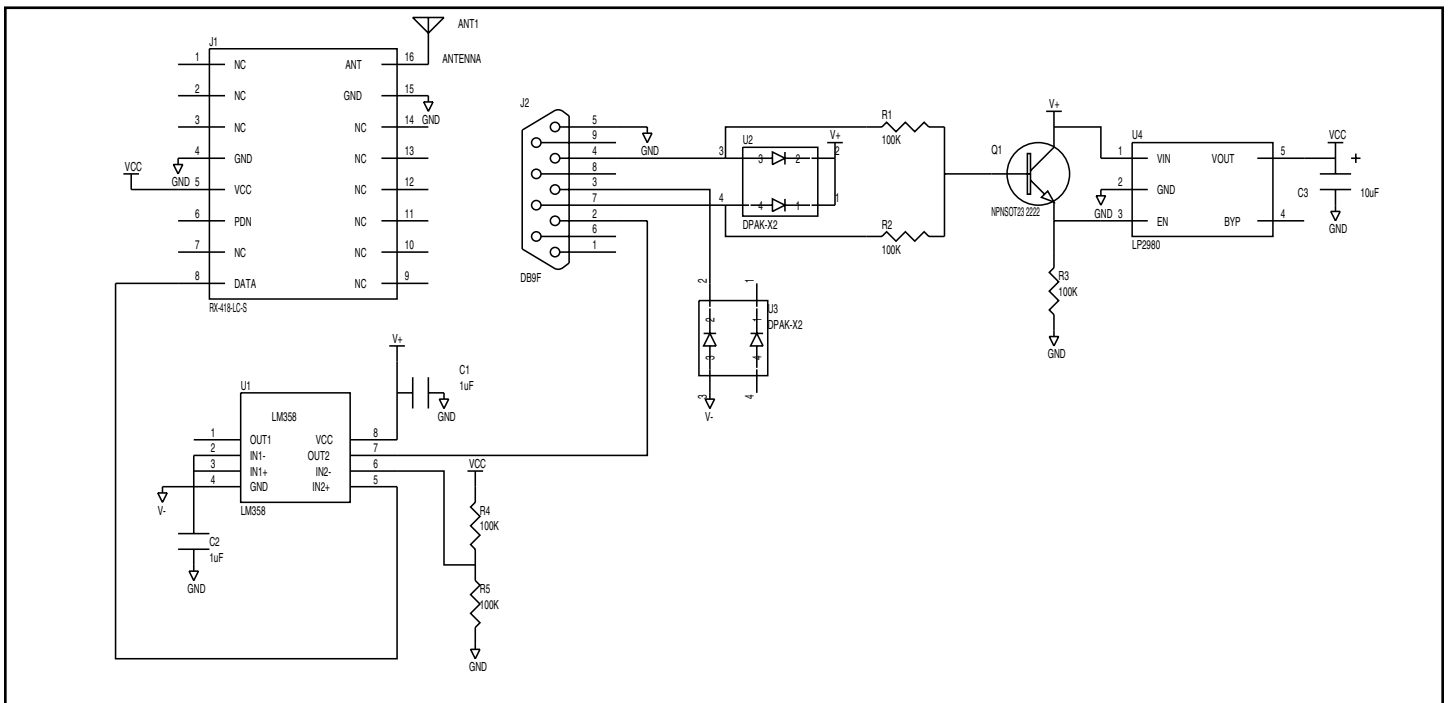
A kit containing all components is also available from RF Digital @ (818) 541-7622

FAX: (818) 541-7644

www.rfdigital.com



Demonstration telemetry transmitter schematic



Demonstration telemetry receiver schematic