LANSCE-RM WIRE SCANNERS: SLIP-ENCODED SERIAL COMMUNICATION FOR MAINTENANCE DISPLAY AT THE INSTRUMENT*

J. Sedillo, J.D. Gilpatrick, LANL, Los Alamos 87545, USA

Abstract

The newest LANSCE-RM wire scanner control systems at Los Alamos National Laboratory's LANSCE particle accelerator facility utilize touch-panel displays for limited status and control at the instrument. Since the wire scanner control hardware utilizes a National Instruments CompactRIO embedded controller, no display may be interfaced to the CompactRIO via conventional means. Thus, in order to display information from the CompactRIO, a touch-panel display and computer combo unit has been added to each wire scanner control chassis with information being exchanged via a common, RS-232 interface. This paper describes the maintenance features available at the touch screen and the underlying SLIP communication scheme used for simple packetized transfer of data between the touch-panel display and the CompactRIO.

INTRODUCTION

The control system outlined in figure 1 represents the solution developed by LANSCE staff for wire scanner actuator control and data acquisition [1]. Among its assortment of features is the control system's chassis-mounted, touch-panel PC (TPC) for localized control and diagnostics of the wire scanner system.

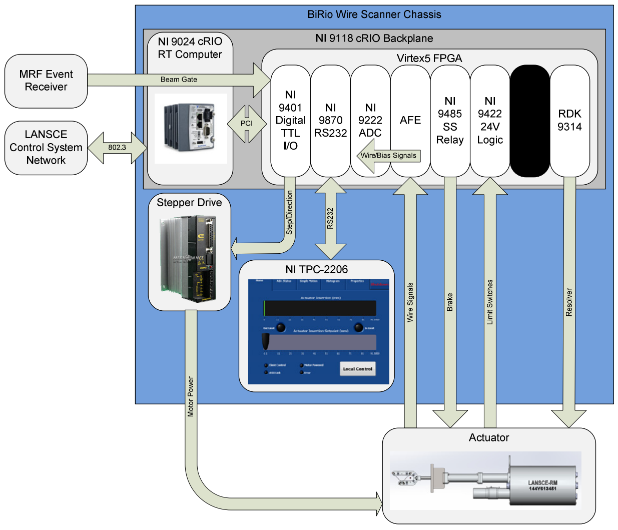


Figure 1: LANSCE-RM Wire Scanner Controller.

Although the touch panel PC offers a variety of high-bitrate interfaces (e.g. Ethernet and USB), the touch panel's RS-232, null-modem interface was chosen due to its ease of use and its common and direct interface to the compactRIO (cRIO) controller's FPGA (via the FPGA's NI 9870 RS-232 c-series module).

*Work supported by the U.S. Department of Energy.

MOTIVATION

RS-232 null-modem links offer a great deal of flexibility with how data is communicated since the hardware does not inherently provide the means to organize data meaningfully other than its FIFO approach. If context of the data may be derived from each individual byte communicated, then no expanded form of data encapsulation may be necessary. However, for more common situations where the data varies in content (e.g. varying data structures) and size (the size of the data structure in bytes), the implementation of a data encapsulation scheme will assist the receiver with making a clear determination of data boundaries by placing the data in "packets".

Data encapsulation schemes are fundamental to many familiar hardware communication systems, the most prevalent of which is 802.3 ethernet. Common encapsulation schemes leveraged on 802.3 networks include the familiar IP, TCP, UDP, etc. Given the successful standardization and implementation of these protocols, almost any one of them may be utilized for serial links implemented on hardware other than 802.3, depending on the application. Since the wire scanner's cRIO transmits a variety of data structures to the TPC and since the data can arrive in an unpredictable order, a data encapsulation method was required to assist both the TPC and cRIO to 1: understand the where a packetized data structure begins and ends, and 2: to determine how to direct the data to the proper methods for processing.

SLIP

IETF de-facto standard RFC1055, otherwise known as SLIP, was deemed the most suitable protocol for data encapsulation [3]. SLIP is an acronym for Serial-Line Internet Protocol and is a data-link layer protocol in the OSI conceptual model. SLIP was developed as a simple means of transmitting TCP/IP datagrams over serial lines when no standard yet existed and has since been replaced by PPP (Point-to-Point Protocol, RFC1661) in most applications. SLIP focuses only on packet framing, and therefore does not inherently provide a means for addressing, packet type identification, error detection/correction, or compression [2].

SLIP's simplicity made it an attractive choice for this application since encoding and decoding methods of more complex protocols would have been needlessly difficult to develop for the cRIO FPGA. Furthermore, the limitations of the SLIP protocol were of little concern since the wire scanner controller can operate without proper operation the TPC; and because the414-6541 link was short (<1 ft), low-speed (56kbps), and directly linked. These qualities ensure low probability of data corruption.

SLIP Encoding

The SLIP encoding method is generally simple to implement. Given a data structure that has been flattened into a 1-D array of bytes, the SLIP encoding method simply appends hexadecimal byte 0xC0 to the beginning and end of the array. These bytes encapsulate the data and form the boundaries of the SLIP packet (figure 2).

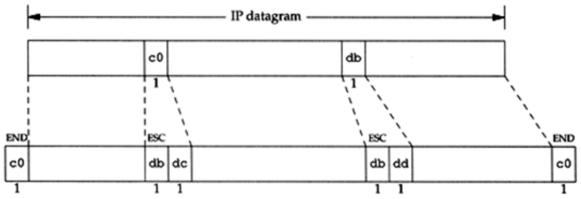


Figure 2: Slip packet layout [3].

If the 1-D array of data contains the value 0xC0, that value is replaced by 0xDBDC (two bytes). If 0xDB is encountered, it is replaced by 0xDBDD (two bytes). Through this method, special SLIP encapsulation characters and data cannot be mistaken by the SLIP decoder.

SLIP Decoding

SLIP decoding is slightly more complicated than encoding. In the ideal case, 0xC0 is the first byte a SLIP decoder encounters upon first receipt of SLIP-encoded serial data. The decoder must simply accumulate the bytes after the 0xC0 header until encountering 0xC0 again. In the not uncommon circumstance where the SLIP encoder utilized special byte patterns 0xDBDC and 0xDBDD, the SLIP decoder will replace 0xDBDC by 0xC0, and 0xDBDD by 0xDB. The data enclosed with the 0xC0 characters may then be passed to other routines for its appropriate use. In the event that a receiving system comes online after data transmission has progressed, the receiving system will simply disregard all data received until it encounters the next SLIP packet's 0xC0 header.

UTILIZATION

After implementing the SLIP encoding and decoding methods for the cRIO and TPC, a custom data encapsulation scheme was developed for incorporation within the SLIP packets. This was necessary to overcome SLIP's lack of a provision for identifying packet types.

Tag Length (1 byte)	Payload Length (2 bytes)	Tag (1-255 bytes)	Payload (1-65,535 bytes)

Figure 3: Custom packet structure.

The custom packet (Figure 3) specifies a 3-byte header and a variable-length payload. The first header byte represents a tag length in bytes. The remaining two header bytes represent the data payload length in bytes. The header is then followed by the tag and the data payload. The tag allows the receiving system to identify the data packet and direct its contents to the correct methods for proper use.

Three communication channels operate within the single, RS-232 link between the cRIO and TPC. The operational flow is shown in figure 4 and described in the sections that follow.

CompactRIO FPGA to TPC Channel

The wire scanner's CompactRIO FPGA sends wire scanner status information to the TPC. This status information includes: actuator position, limit status, sense wire continuity, remote operation status, the general error condition, network ID, beam gate, FPGA state, and the cRIO C-series module status.

The cRIO FPGA accumulates the data from its internal processes, encapsulates the data within "custom" packets with unique tags representative of the data, and then directs the custom packets through a SLIP encoding process. The SLIP packets that result are then directed to the RS-232 FIFO for transmission to the TPC. The FPGA has the least flexibility with regard to packet size since arrays dedicated to the transmission and reception of the packets must be of a fixed size prior to FPGA compilation. The corollary to this is that all packets transmitted to the FPGA from external systems must also fit within the FPGA's array size constraints.

TPC to CompactRIO FPGA Channel

The TPC sends commands to the cRIO FPGA for minimal control of the wire scanner actuator. Command information includes the actuator position setpoint and actuator operational envelope scan. These data allow for some localized control of the wire scanner actuator mechanism.

Data transmitted from the TPC to the cRIO FPGA occurs in a similar way to the "CompactRIO FPGA to TPC Channel" process. Although the TPC can dynamically allocate memory for substantially large SLIP packets, the size of the packets it transmits are of a size that are compatible with the maximum allocated array size of the cRIO FPGA's receive array.

CompactRIO RT to Touch Panel PC Channel

The cRIO RT system sends wire scanner configuration data to the TPC through the cRIO FPGA's RS-232 link. Configuration data includes:

- Actuator configuration: leadscrew pitch, stepper motor resolution, sense wire center positions, and mount angle.
- Motion control configuration: closed-loop control constants, position deadband, and stepper motor settings.
- Beam data acquisition configuration: AFE (Analog Front End) transimpedance values, ADC sampling rate, pre and post trigger samples, and macropulse waveform cumulative average count.

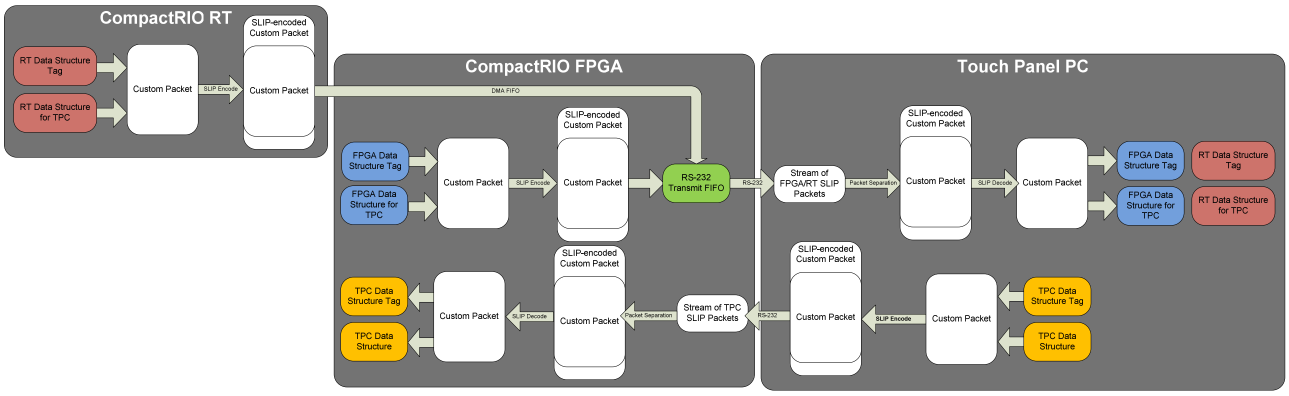


Figure 4: Overview of cRIO/TPC Communication flow.

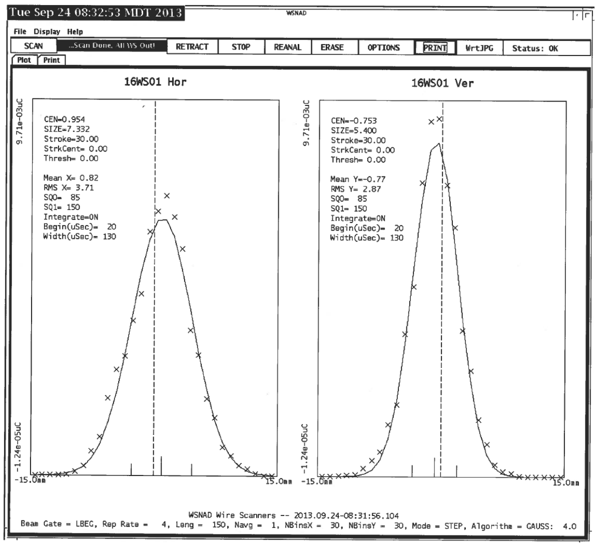
The relatively large amount of data transmitted by the cRIO RT system to the TPC made it essential that data packets produced by the cRIO RT avoid the cRIO FPGA's bottlenecks. To do this, the cRIO RT encodes its data before transmitting the resulting data stream to a DMA FIFO read by the FPGA. Once the FPGA has completed sending its own packet stream to the TPC, it checks the DMA FIFO for data content. If data exists, the FPGA simply directs the data directly into the RS-232 FIFO for transmission to the TPC.

IMPROVEMENTS

SLIP is simply one of many methods for data encapsulation over serial streams. Again, SLIP does not inherently provide a means for addressing, packet type identification, error detection/correction, or compression. This work has compensated for SLIP's lack of packet type identification and required minimal error correction since a high-fidelity data link was utilized. More demanding applications may require more features from a SLIP implementation. In such cases, other protocols that possess the necessary features may be encapsulated within the SLIP packet or used outright, depending on the protocol and its compatibility with the application.

LANSCE-RM WIRE SCANNER PROJECT UPDATE

Six new LANSCE-RM wire scanner systems have been deployed throughout the LANSCE facility: two in the 201 DTL, Three in the 805 linac, and one in the switchyard. Integration of these systems is complete with all systems fully utilizable by LANSCE's central control room. Figure 5 shows a beam profile obtained by one of the wire scanners deployed in the 805 linac. Additional systems are planned for installation as resources become available.



```
CEN=0.954
SIZE=7.332
Stroke=30.00
StrkCent= 0.00
Thresh= 0.00

Mean X= 0.82
RMS X= 3.71
SQ0=  85
SQ1= 150
Integrate=ON
Begin(uSec)=  20
Width(uSec)= 130
```

```
CEN=-0.753
SIZE=5.400
Stroke=30.00
StrkCent= 0.00
Thresh= 0.00

Mean Y=-0.77
RMS Y= 2.87
SQ0=  85
SQ1= 150
Integrate=ON
Begin(uSec)=  20
Width(uSec)= 130
```

```
WSNAD Wire Scanners -- 2013.09.24-08:31:56.104
Beam Gate = LBEG, Rep Rate =   4, Leng =  150, Navg =  1, NBinsX =  30, NBinsY =  30, Mode = STEP, Algorithm = GAUSS:  4.0
```

Figure 5: LANSCE Control Room view of wire scanner-acquired beam profiles.

REFERENCES

[1] J. Sedillo, et al. "First Test Results of the New LANSCE Wire Scanner." PAC '11, New York, March 2011 MOP236; http://www.JACoW.org

[2] J. Romkey, "A Nonstandard for Transmission of IP Datagrams Over Serial Lines: SLIP," RFC1055, http://www.ietf.org/rfc/rfc1055.txt

[3] http://recolog.blogspot.com/2012/10/serial-line-internet-protocol-slip.html