Measuring Pulse-Code-Modulation Signals with CompactRIO

Print this Page

Pulse-Code-Modulation (PCM) is a technique used to transmit digital values over radio frequency (RF) systems by encoding them as pulse-widths. In our application, a Futaba radio control aircraft radio transmits joystick and dial positions using PCM wireless communication. The Futaba receiver decodes the radio signal and outputs 8 channels of data as TTL-compatible pulses.

Our goal is to create a mechanism for reading these PCM signals with an embedded computer, interpret them in real-time as operator instructions (joystick positions), and then combine the instructions with data acquired from other sensors to create adjusted control commands for the robotic device. In this way, we are able to include additional intelligence in our robotics system, allowing more sophisticated control of our device than a human operator could achieve directly from the radio transmitter.

The new CompactRIO reconfigurable embedded system provides us with a small, embedded, high performance solution. PCM decoding is performed in the reconfigurable FPGA logic and then the data is passed to the embedded real-time processor.

Table of Contents:

About the Author

David Thomson works at the NOAA Aeronomy Lab, where he develops atmospheric chemistry instrumentation for use on high-altitude research aircraft. He is also the principal of Original Code Consulting, an NI Alliance Member which provides LabVIEW development and system integration for laboratory, R&D, and manufacturing clients.

Pulse Code Modulation Signals

This article presents a method for completing the first one-third of our task: measuring the PCM pulse widths. Since two of the eight radio channels are special cases that are processed using external hardware, we need a solution that can read six channels of PCM data. Each of these channels has pulses arriving at a rate of 70 Hz, and the channels are out of phase with respect to

each other. Each pulse varies from 1.0 to 2.0 milliseconds, with approximately 1.5 ms corresponding to a centered joystick, and larger or smaller pulses corresponding to joystick positions away from center. The data transmitted by the radio has 10 bits of resolution, meaning that for optimal measurement of the pulse widths, we require 1 microsecond resolution. The cRIO-9411 module has specifications that are ideal for this application. It supports 6 digital inputs and is compatible with TTL-level signals. Furthermore, it has a response time of less than 1 micro-second, allowing excellent measurement resolution of our pulses.

The LabVIEW FPGA code for this task is actually quite simple. As shown in Figure 1 below, the diagram for this task consists of a single loop with a cycle time of 1 microsecond. Each pulse width measurement takes the form of a counter on a shift register which is incremented every iteration of the loop for which the associated input bit is high. When the bit goes low, the number of microseconds counted is used to update that channel's indicator. With this architecture, the indicators for each of the six channels are all updated with minimal latency, even though the pulses have various phases. This system provides high-performance pulse width measurement for all six channels, while using straight-forward code.

In comparison, our previous PXI system utilized a 6602 counter-timer card. Although this card has 8 counter-timer channels, it has only three DMA channels, so only three of the radio channels could be acquired with minimal latency. In addition, the code for the 6602 was considerably more complex. Although the FPGA hardware did require several hours of time for learning and understanding the overall paradigm of its operation, once that threshold had been crossed, the programming time required for implementing this solution was a fraction of that required for creating the PXI-6602 solution.

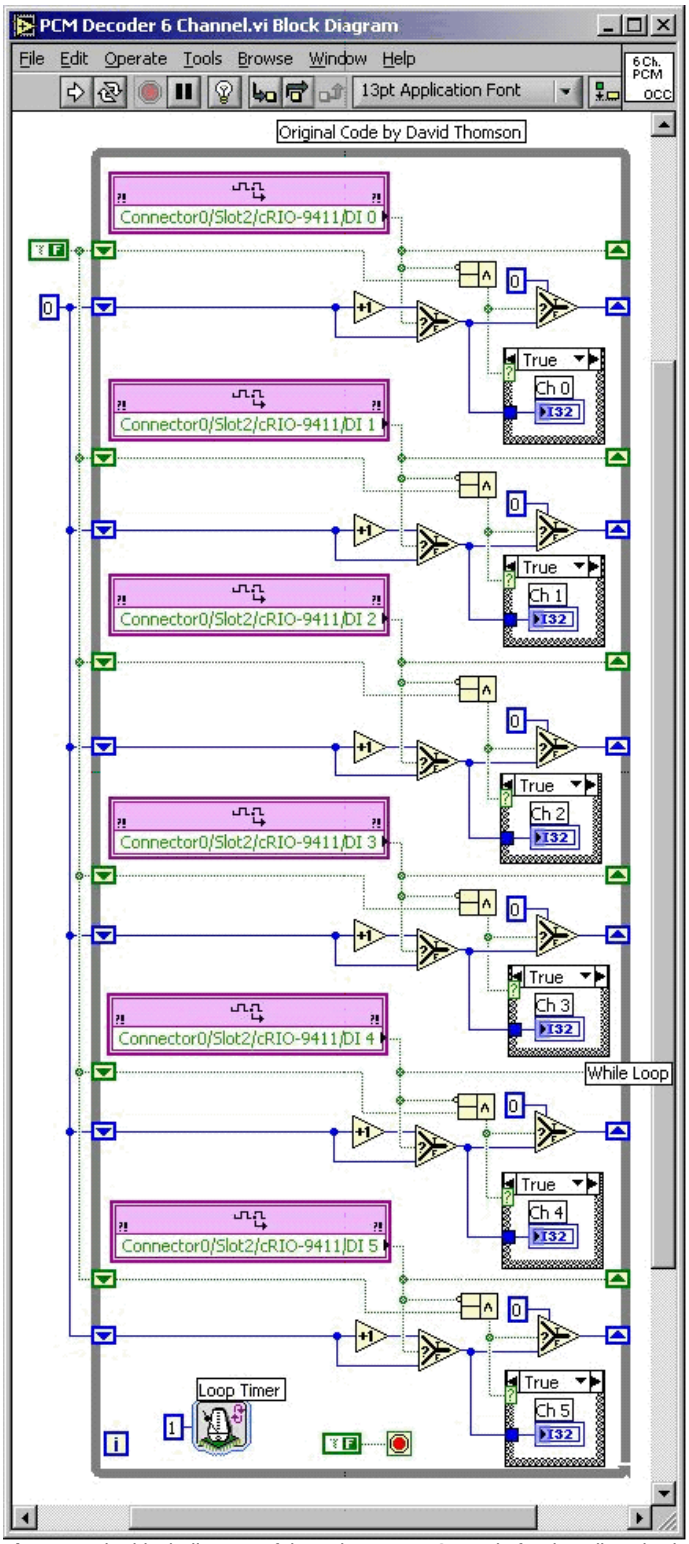

Figure 1. The block diagram of the LabVIEW FPGA code for decoding six channels of PCM data using a single 1 microsecond loop and six pairs of shift registers.

Passing Data to the Real-Time Controller

Once the PCM data has been read into the FPGA and decoded, it needs to be combined with additional data and used to create

the control signals. For relatively simple algorithms, these control calculations could be executed directly in the FPGA, for a very robust, high-speed solution. In our case, however, we used the CompactRIO embedded real-time controller to perform the control calculations. Thus we need an additional real-time VI for the real-time controller to extract the PCM data from the FPGA module. Writing this real-time VI was accomplished in less than an hour through the use of National Instruments example code. The diagram for the real-time code is shown in Figure 2. This VI loads the FPGA code through the use of the FPGA Reference node, then reads the FPGA VI's front panel controls in a loop with a cycle time of 10 milliseconds.

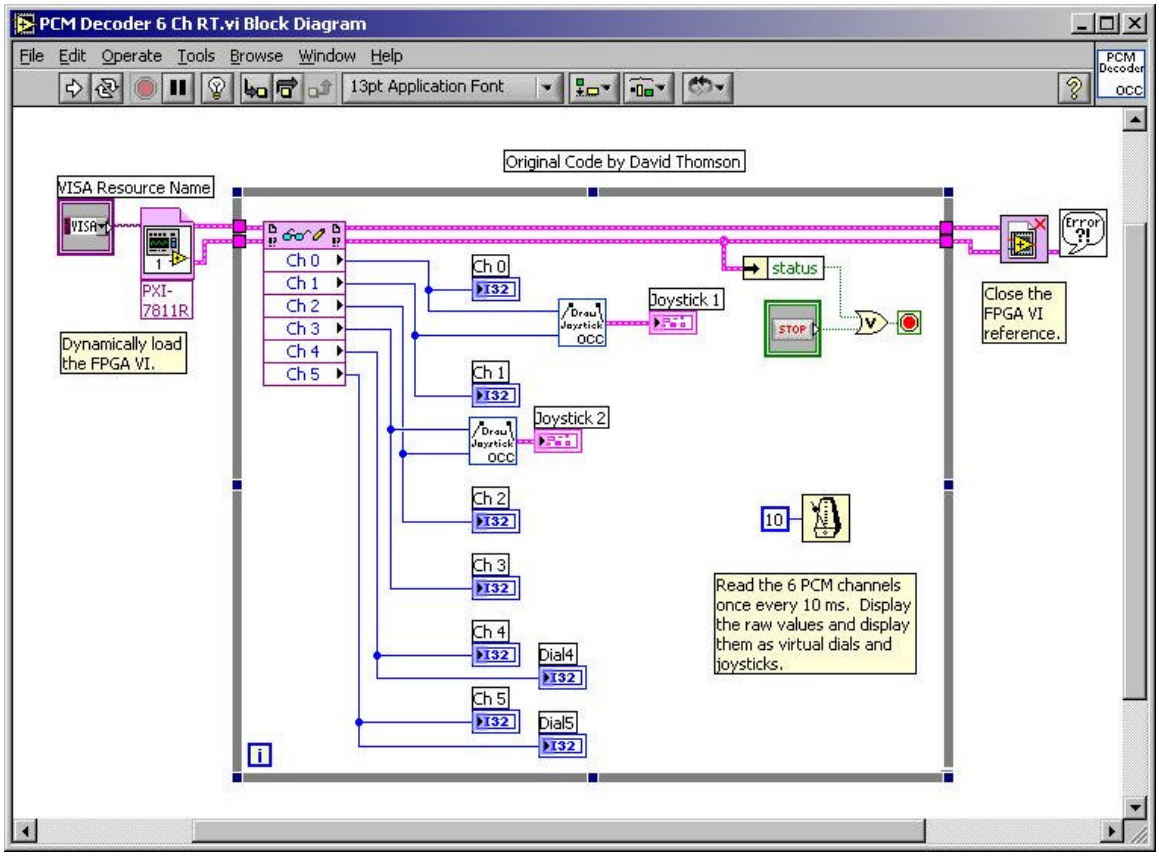

Figure 2. Block diagram of the LabVIEW Real-Time code that reads the PCM data from the FPGA. This example VI only displays the PCM data as virtual joysticks and dials, but future versions will apply various control algorithms to this data and then send it back to the FPGA for generation of the calculated control signals.

Conclusions

As can be seen in Figure 3, the front panel of the LabVIEW Real-Time program displays the PCM data as virtual joysticks and dials. The Futaba radio is shown on the left side of Figure 3, so that the correlation of the real and virtual joysticks can be seen.

This project quickly and easily demonstrated the feasibility of using CompactRIO to create a high-performance PCM decoder. Since the CompactRIO FPGA system is a new, advanced technology, some time was required to understand how to perform application development, including the use of the Embedded Project Manager, management of channels names, targeting the FPGA hardware, and compiling, downloading, running, and linking to the FPGA LabVIEW code. However, once the development process was understood, the current solution was built in a very short amount of time. Furthermore, the resulting system provides a significant increase in performance compared to the previous version built on PXI and LabVIEW Real-Time technology.

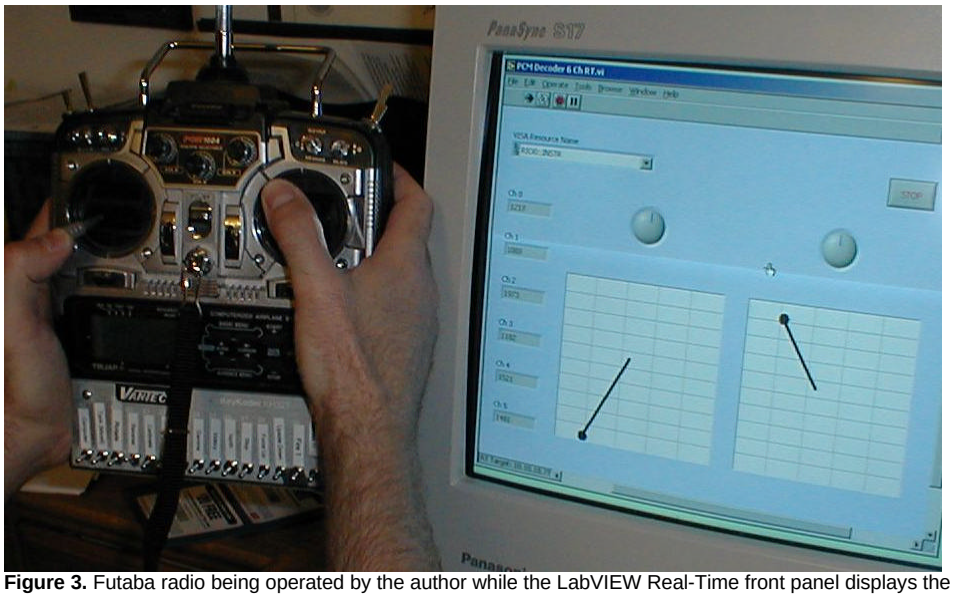

Figure 3. Futaba radio being operated by the author while the LabVIEW Real-Time front panel displays the measured joystick positions.

CompactRIO Example Code

See Also:
Example Program: Measuring Pulse-Code-Modulation Signals with CompactRIO

Print this Page