USING LABVIEW FOR TELEMETRY MONITORING AND DISPLAY

by

George Wells, Member of Technical Staff
Edmund C. Baroth, Ph.D, Manager of the Measurement Technology Center,
Jet Propulsion Laboratory, California Institute of Technology

Abstract

Part of the Jet Propulsion Laboratory's (JPL's) Instrumentation Section, the Measurement Technology Center (MTC) evaluates data acquisition hardware and software products for inclusion into the Instrument Loan Pool, which are then made available to JPL experimenters. As such, it acts as a focus for off-the-shelf products. The MTC also configures turn-key measurement systems that include integrated sensors, signal conditioning, data simulation, acquisition, analysis, display and control capabilities. Visual programming is used to simplify configuration of such systems.

Employment of a visual programming language is the most important factor in enabling the implementation of data acquisition, analysis, display and visualization systems at low cost. Other important factors are the use of commercial software packages and off-the-shelf data acquisition hardware where possible. Understanding the experimenter's needs is also critical. An interactive approach to user interface construction and training of the operators is also important.

An example of a telemetry monitoring and display application using two Macintosh computers aad a graphical programming language (National Instruments' LabVIEW 2) will be discussed. One computer acted as the telemetry source and the other as the, analyzer. The telemetry stream was emulated using interface boards in the computers. A schematic of the system and user interface panel will be presented. The computer programs will also be discussed as to ease of creation. The purpose of this paper is to show how using graphical-based programming software can be used for advanced data analysis like telemetry monitoring and display, as well as for emulation of a telemetry stream.

This application was created as the result of parallel development between a visual programming team and a text-based ('C') programming team. Although not a scientific study, it was a fair comparison between different development methods and tools. With approximately eight weeks of funding over a period of three months, the visual programming effort was significantly more advanced, having gone well beyond the original requirements than the 'C' development effort, which did not complete the original requirements. This application verified that using visual programming can significantly reduce software development time. As a result of this initial effort, additional follow-on work was awarded to the graphical programming team.