A LabVIEW-based Accelerator Instrumentation Platform

W. Blokland
Fermi National Accelerator Laboratory[1]
P.O. Box 500, Batavia IL 60510

Abstract

An overview is given of the developed LabVIEW-based accelerator instrumentation platform and its applications. LabVIEW provides the software basis with its extensive analysis library and drivers to a variety of data acquisition hardware. An in-house developed interface integrates the platform into the Fermilab Accelerator Controls Network. Procedures have been designed to assist in the development process. The wide variety of applications implemented on this platform demonstrates its overall usefulness.

INTRODUCTION

LabVIEW

The decision of the Instrumentation Group to use the programming language LabVIEW as the main means to implement instrumentation applications is based on its ease of programming, the extensive analysis libraries, and the supported interfaces to many instrumentation platforms, e.g., VXI and GPIB. At the time, LabVIEW was only available on the Macintosh, making this the computer of choice.

Instrument Configuration

A typical application in the Instrumentation Group performs data acquisition, data-analysis, and presentation of the results under control of a console application or guided by accelerator events. The data acquisition hardware can be a stand-alone device, e.g., a GPIB scope, or a module in a crate, e.g., a VME digitizer. Commercially available and LabVIEW supported interface cards connect the Macintosh to the device or crate. The data analysis is done in LabVIEW which is sometimes extended with an add-on package.

Network interface

To make LabVIEW practical for accelerator instrumentation purposes, a communication interface was written according to the Accelerator Control Network protocol (Acnet). This interface enables the control system to read LabVIEW analysis results and set variables to control the program flow, see [1]. To complete the analysis, some applications need the values of accelerator variables such as beam current or bunch intensity. Instead of requiring a console application under control of an operator to set LabVIEW variables to these values, the interface was expanded to talk, using TCP/IP, to a Vax program that has similar access to accelerator data as a console. While this is indirect, as opposed to using the Acnet protocol, the advantage is that it uses the standard console functions such as logging, database, and safety routines, which otherwise would have to be reimplemented on the Mac. The new part is fully implemented in LabVIEW, as opposed to C, to enable other LabVIEW supported computer systems to use this part of the interface as well. While implementing the new part in LabVIEW results in a slower execution, this does not pose a problem. As opposed to the supply part, which is written in C, where many requesters for possibly large data sets must be quickly serviced, the request part is only a one-to-one link for small data sets.

Besides access by console applications using the Acnet protocol, the Mac itself can be controlled by remote control programs. Such programs, e.g., Timbuktu which uses the TCP/IP or Appletalk protocols, and PlanetX which is an XWindows client, are for use by an expert, most often the developer, to diagnose the operation or to perform studies. The console applications and generic parameter pages remain the designated way the operators control and monitor the instrumentation.

DEVELOPMENT AND MAINTENANCE

As more people started to use the platform and more applications became involved with the daily operations of the accelerator, it became clear that the current informal setup was not sufficient to provide a smooth development path to a trusted, reliable, and maintainable instrument. A beginning was made to formalize the development process to solve the problems relating to software backup, software installation, communication with other groups, documentation, system verification, diagnostics, and integration into the control system. The development and maintenance process is separated into phases with procedures associated with each phase and each transition. There are three phases defined:

1) Stand-alone phase: The developers are programming and configuring the hardware. Initial tests are performed but on a stand-alone basis. Only the developers are involved.
2) Commissioning phase: The instrument is hooked up to the control system which now sets and reads out data. Studies are done to find bugs, improve performance, and enhance capabilities. Operators are involved in a limited way.
3) Operational phase: The instrument is now part of the control system and is used by the operators. The main emphasis is on maintaining the operational status of the instrument.

An important aspect in designing the procedures was to keep them as simple as possible. They should assist the developer and not feel like a bureaucratic hindrance. The proce-

[1]Operated by the Universities Research Association Inc., under contract with the U.S. Dept. of Energy.

dures are represented as simple lists with items to complete. These checklists are available for each transition to the next phase and each change within a phase. Additional explanations are available for people using the list for the first time. See figure 1.

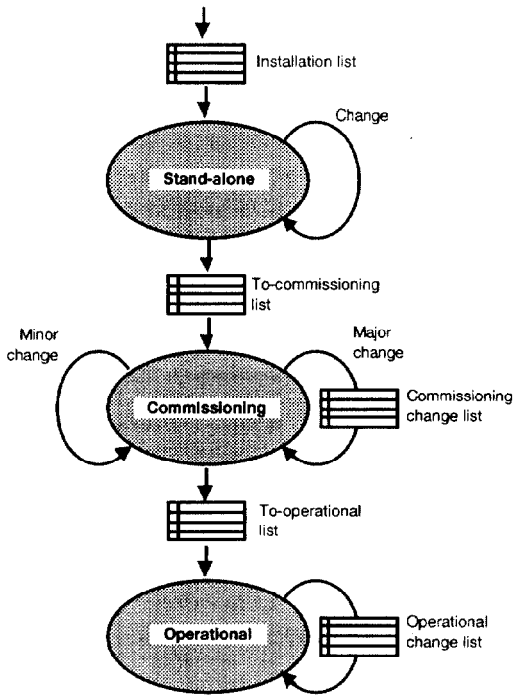

Figure 1. The development phases.

To start a project, the developer obtains the project folder which contains the lists and the documentation about the Acnet interface and Macintosh utilities to be used. As the project progresses, all notes and created documentation should be added to the folder. The first list tells which utilities should be installed so that every instrumentation computer has the same (trusted) utilities and everyone can expect the same tools to be available. Because the development process in the stand-alone phase is a continuous series of changes and additions, too many to document, there is no list for each change but a requirement to backup the software at least once a week.

To continue to the commissioning phase, the list for the transition must be completed. List items require notifying the involved groups, (at least the Operations Group), backing up software, and adding an entry in the Instrumentation Book (IB). An entry in the IB includes the status of the project, a diagram of the configuration, a description of its operation, the table of LabVIEW variables accessible by the control system, a description of diagnostic procedures, and operator training sheets. The IB is located in the Main Control Room and is intended for use by the operators. A software tool is available to help integrate the application to the control system. It generates from the same table that LabVIEW uses to initialize its side of the interface, the database file that the control system needs to register the variables accessible in LabVIEW. In this way, both sides use the same table, the developer is not required to know how to write database entry files, and the table can be used as a document of the registered variables. Adding the Acnet interface to the LabVIEW program typically takes less than a day unless an intrinsic control is required using a custom-made console application. Often though, there is no need for a specific console application and the generic ready-to-use parameter page suffices.

Any major change that the developers make must be documented according to the commissioning change list unless the change is irrelevant to the operators or programmers of the console application. This judgment is left to the developers to avoid too much paper work while an error at this stage would not halt accelerator operations. Any significant change must be noted in the IB and the version number of the software must be updated. The version number is also available as a LabVIEW variable and can be seen on-line on a console for easy reference with an IB entry.

To bring the system up to operational status, the developer must provide the means to rebuild the system from the ground up, complete the IB entry, complete the spare parts list, pass the system certification test, and notify the Operations liaison. The system certification includes a review of the analysis and a complete from-the-ground-up software installation on a different computer to prove that, in case the computer fails, it can be resurrected on a spare.

APPLICATIONS

So far, six projects have been created using the platform. All are hooked up by a tokenring card to the control system and can be remotely controlled over ethernet.

Synchrotron Light Monitor

The transverse image of the beam formed by synchrotron radiation is displayed on a Quadra 950. The light is converted by a CCD camera to video signals which are read into the Mac by a frame grabber card. A stepper motor card interfaces to the positioning motors which aligns the camera within the telescopic system. A GPIB card controls a CAMAC crate which has timing and power supply modules. The image is captured and analyzed by a LabVIEW add-on package. All hardware and software were commercially available and needed only minor modification. See [2].

Beam Line Tuner

On injection of the beam in the ring, the turn-by-turn beam positions are measured to estimate the initial phase and amplitude of the betatron oscillation. A console application page retrieves the results and calculates the beam line corrections. The Macintosh IIci is hooked up to a VXI crate through the MXI interface. The VXI crate contains VXI digitizers, cards for decoding of injection events and beamsync signals, and RF-multiplexers to switch between beam position signals. See [3].

Sampled Bunch Display

A Quadra 950 controls two GPIB scopes taking samples of the proton and pbar bunches in the Tevatron and analyzes their beam intensities. Timing signals come from a GPIB con-

trolled CAMAC decoder card. A major part of the program is the control logic to set the scope and analysis for the various states of the accelerator. The intensities are displayed in the lab wide status display and used for luminosity calculations. See [4].

Ion Profile Monitor

For both vertical and horizontal planes, a 30 point beam profile from micro-channel charge pickups is sampled. All 20,000 turns of the Booster cycle are captured by VME digitizers. CAMAC modules provide timing signals and control of the high voltage power supply of the pickups through a GPIB interface. The analysis determines the position and emittance for each turn. The data, 2.5 Mbytes, can be displayed for any turn individually or as a whole cycle, (decimated for display), in a color intensity plot. The Quadra 650 takes about 15 seconds to retrieve the data and display the results. See [5].

Collision Point Monitor

Each side of the collision region has a horizontal and vertical pair of beam position pickups. A GPIB multiplexer directs the signals from beam position pickup plates to a GPIB scope. The LabVIEW program retrieves the data and determines the position and time of a proton and pbar bunch to derive the collision point. The results are accessed through a standard parameter page on a control console.

Flying Wires

Transverse beam profiles are obtained by flying wires through the beam and digitizing the resulting beam loss intensities and current wire positions. A Macintosh Quadra 950 controls the wires' movement using a Nubus motor controller card. In a VME crate, controlled through a MXI interface, digitizers sample the beam loss signals on each turn of the beam. The LabVIEW program reads the data in and analyzes the profiles to determine the emittance of the beam. Work is in progress to speed up analysis and create a console application to present the results to the operators.

EXPERIENCES

Because LabVIEW is a graphical language, one is saved the time normally spent debugging syntax errors. Additional time is saved by the availability of many drivers for the data acquisition hardware. Very little time is spent on the display or communication at the Macintosh side due to the built-in graphics. If, however, the generic parameter page does not suffice, a console application must be written, (in C or Fortran), which can take extra time. Thus by using available hardware and the LabVIEW software, we found that most time, up to eighty percent, is spent on programming the analysis routines and the control flow of the program.

Besides the availability of LabVIEW for the Macintosh other useful utilities are available as well, e.g., screen snapshot takers, installer programs, and remote control programs.

A disadvantage of the Macintosh is the lack of a real-time kernel. The main factor that alleviates this lack is that our applications are geared towards monitoring, not controlling. Results are presented to the operators who don't require a fixed time interval in which an answer must be ready. Time-critical operations, such as timing the data acquisition, can all be done in hardware. The LabVIEW program only checks to see if the data acquisition is complete and if the device must be reprogrammed to change the measurement setup. It is possible to do time-critical operations if one could assure that nothing else than the LabVIEW program would be running on the Macintosh. However, we use remote control programs like Timbuktu and PlanetX. These can be used anytime and interrupt a time-critical operation.

We found that system crashes are often due to hardware incompatibilities, e.g., accelerator cards and expansion chassis, and software incompatibilities, e.g., system extensions. The solution was to replace or not use the offending parts. Crashes due to program errors are normally found and corrected for in the stand-alone or commissioning phase. To handle the crashes at remote locations, the Macs can be rebooted by cycling the power using a remote controlled switch. The most common reason to reboot a machine is after an unsuccessful change in the LabVIEW program or because an upgrade of a utility proved incompatible with the particular Mac. To date the longest uninterrupted operation of a Mac was several months.

The procedures for the development and maintenance process are new and we just started applying them to the projects. We expect to make modifications as needed.

All developers had a favorable impression of the LabVIEW-based platform and estimated that it would be much harder and more time-consuming to implement a similar application on an ASCII language based system.

ACKNOWLEDGMENTS

I like to thank the many people in the Instrumentation, Controls, and Operations Groups for their advice and help.

REFERENCES

[1] W. Blokland, "An Interface From LabVIEW To The Accelerator Controls Network", Proc. of Accelerator Inst. Workshop, Berkeley, USA, 1992, pp. 320-329.

[2] A. A. Hahn and P. Hurh, "Results From An Imaging Beam Monitor In The Tevatron Using Synchrotron Light", HEACC'92, Hamburg, Germany, July 1992, pp. 248-250.

[3] W. Blokland, "A VXI/LabVIEW-based Beamline Tuner", PAC'93, Washington.

[4] E. Barsotti, "A longitudinal Bunch Monitoring System using LabVIEW and high-speed oscilloscopes", To be published at 1994 Accelerator Instrumentation Workshop in Vancouver.

[5] J. Zagel, "Booster Ion Profile Monitor using LabVIEW", To be published at 1994 Accelerator Instrumentation Workshop in Vancouver.