ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA128

USING LABVIEW TO BUILD DISTRIBUTED CONTROL SYSTEM OF A PARTICLE ACCELERATOR

V. Aleinikov, I. Borina, A. Krylov, S. Pachtchenko, K. Sychev, FLNR JINR, Dubna, Russia

Abstract

New isochronous cyclotron DC-280 is being created at the FLNR, JINR. Total amount of the process variables is about 4000. The variety of field devices of different types is 20. This paper describes architecture and basic principles of the distributed control system using LabVIEW DSC module.

INTRODUCTION

Charged particle accelerator is a large automation subject that contains hundreds of devices and thousands of signals to control and monitor. To simplify design and maintenance of a control system it is convenient to divide it into parts or subsystems. Every subsystem is dedicated to number of common tasks. They are simply as follow: produce, inject, accelerate, extract and transport the beam to the target. Based on our experience, we decided to consider 3 subsystems: Injection (ECR source, axial injector), Accelerator (cyclotron, extraction, transport) and Low level RF. Each subsystem includes its own vacuum control, water cooling, beam diagnostics and so on.

This approach gave as:

- Independent step by step development, debugging and commissioning of every subsystem.
- Fast response of the system as a whole due to the distribution of computing among several computers.
- Modular structure is easier to maintain and upgrade.

LABVIEW? WHY NOT?

This question we asked at the start of the project. Fifteen years ago it would be too risky solution. LabVIEW has not been widely accepted for control systems of accelerators and large experimental control systems due to limitations in performance, scalability, and maintainability of large LabVIEW designs. Some of this limitations were far-fetched, some real.

Since 1999 we have been using SCADA FlexControl that runs under QNX operating system [1, 2]. Over time, the lack of support for both products forced us to consider replacing the development tool. At present, we see continuously increasing of processing power and significant evolution of the LabVIEW. For example, appearance of the Datalogging and Supervisory Control (DSC) module made it full featured SCADA. It includes tools for logging data to a networked historical database, tracking real-time and historical trends, managing alarms and events, and adding security to user interfaces [3]. It has powerful mathematical, graphical support and experience of thousands of users. Support is provided for a huge number of device drivers and protocols. It has good connectivity and openness which are very important for us due to big variety of the hardware we use.

Ten years' experience of using LabVIEW for various projects, completion of National Instrument training courses, has encouraged us to select LabVIEW as the development tool for new control system.

SHARED VARIABLES

The control system of DC-280 is a project that is distributed over a network. Its essence (all signals) can be described by means of process variables. Every subsystem consists of variables which are deployed on the dedicated host. To share data across the network or between applications, LabVIEW offers NI Publish-Subscribe Protocol (NI-PSP). NI-PSP is a proprietary technology that is optimized to be the transport for network shared variables and provides fast and reliable data transmission for large and small applications. It is installed as a service on the computer when you install LabVIEW [4]. It provides network-published shared variables that publish data over a network through a software component called the Shared Variable Engine (SVE). The SVE manages shared variable updates. Publishers send updates to a server, SVE, and subscribers receive those updates from the server. Shared variable can be connected to a front panel control or to another variable. We use this concept to provide communication between device drivers, applications on localhost and remote computers of the control system of DC-280 cyclotron.

ARCHITECTURE

The control system of DC-280 has client-server architecture. Every parameter of a subsystem is deployed as shared variable in memory of the dedicated computer where SVE hosts it.

SVE is the server for a shared variable and all references are the clients, regardless of whether they write to or read from the variable. All applications are clients or subscribers to the SVE. Being network published, variables of every subsystem can be accessed from any node. The set of all process variables are distributed across the nodes of each subsystem. They are deployed at the system start up and available for the operator panel or automation algorithms.

DEVICE DRIVERS

In order to unify device driver development we use modular and object oriented approach with plug and play ideas. To provide simultaneous managing of many devices of the same type or manufacturer we use Driver Loaders. For example, to control of 20 power supplies from

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA128

EVPU Company, in parallel mode, the Driver Loader starts 20 clones of a single power supply device driver. Every virtual instrument runs independently with unique parameters that come from the Loader's initialization file.

LabVIEW provides the aliasing one to another shared variable with scaling. For example, using linear conversion, a raw value of ADC code can be automatically calculated to the engineering value (voltage) if value changes. At start up every device driver creates network-published shared variables for supported hardware. These raw variables are connected to engineering process variables by means of alias mechanism (See Figure 1).

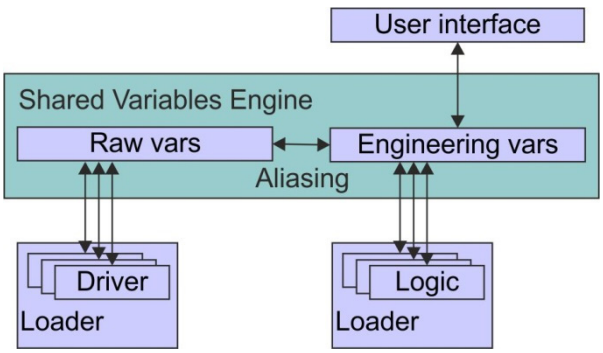

Figure 1: Data flow.

After connection to the device is established the driver cyclically reads its status and writes it to the corresponding *input* shared variables. It also receives notifications of the value change of the *output* variables, which causes the driver to send commands to the device respectively with communication protocol.

LOGICAL OBJECTS

Some objects require more complex calculations or control logic. For example, to switch the device you need to generate a 2-second pulse output signal or state value - the result of calculating several Boolean ones. For this purpose, we have created a set of control algorithms for the most typical objects of automation. This object-oriented driver uses polymorphism and supervises involved variables depending on the device type at the start up. All objects and their variables are described in the initialization file. The Loader starts services for all objects as independent processes accordingly to this file.

OPERATOR'S INTERFACE

The user interface application (UI) is intended to visualize control and monitoring of each subsystem for the operator and can be executed on any node of the control system network. It uses Data Binding technic to connect front panel controls and indicators to the process variables.

The user interface screen is divided into several areas with different purposes (See Figure 2):

- General acceleration parameters, operator name, date, time, etc.
- Menu with buttons for selecting the front panel of the subsystem for control.
- Front panel with controls and indicators of subsystems.
- The context line contains useful service information about the object under the mouse cursor.
- System events and alarms windows.

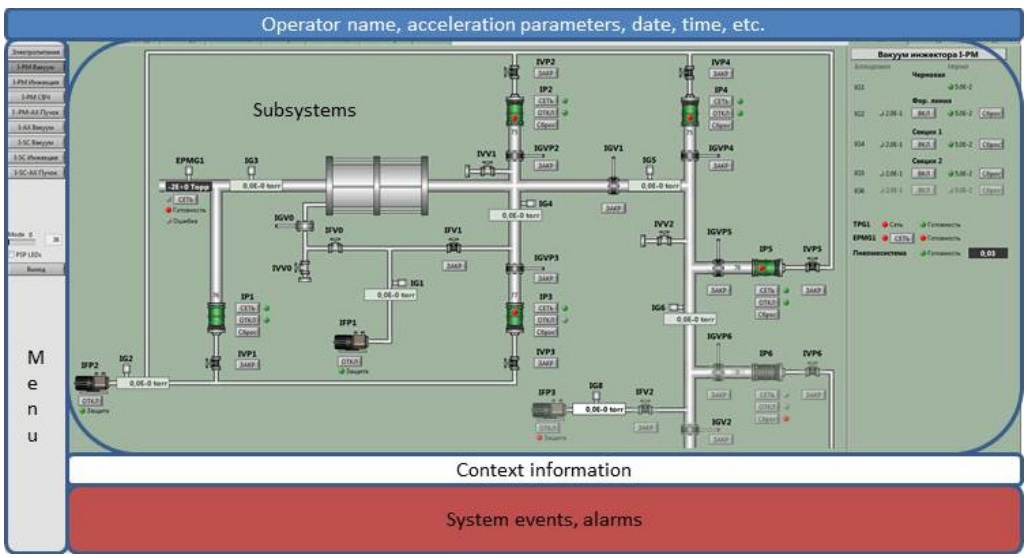

Figure 2: The user interface layout.

The display of objects is realized in the form of graphical elements similar to real devices. Their state is displayed by the colour and shape of the picture.

In addition to the standard services offered by the user interface for the operator (visualization of the control process, printing of reports, alarms, etc.) we have implemented useful functions for servicing and repairing the system in the event of a malfunction.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA128

When a driver loses connection to its device, all controls and indicators that represent this device become disabled and grayed out (See Figure 3).

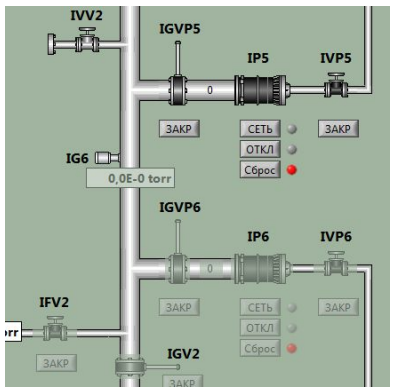

Figure 3: Online and offline states of objects.

If the operator position mouse cursor over an object, the context string displays actual values of raw variables on which that object depends, for example, ADC code.

For safety reasons the interlocking system has a lot of signals that block the operation of some important objects. The operator can easily check the list of all actual interlocks in the popup window by using context menu on the selected object (See Figure 4).

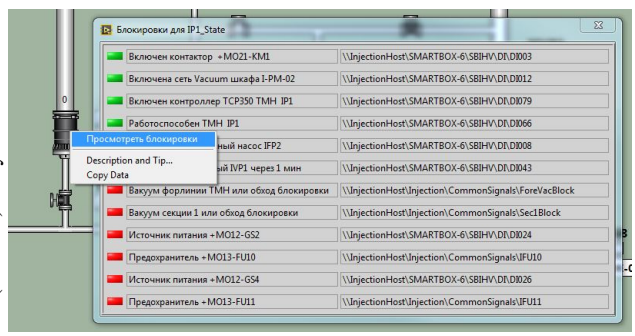

Figure 4: Popup window with the list of interlocks.

These functions were obtained thanks to our experience in the development of visualization programs for control systems of operated cyclotrons. They greatly simplify the work of engineers to find and eliminate any hardware problems.

CONCLUSION

The control system of DC-280 cyclotron as well as the software for it is in a state of development. Most of the technologies have already been worked out. More than ten device drivers were developed and tested. The software for Low level RF and the Injection subsystems have been created and tested in emulation mode. The Accelerator subsystem is creating now.

Since the system is growing, we still have to develop software for creating and remotely monitoring the historical database, the ECR source spectrum acquisition program and many other applications.

The UI contains about 1000 controls and indicators now. Simultaneous work of visualization and more than a hundred processes of drivers on one computer (2.5 GHz AMD, 4-cores) did not reveal delays in the response and interruptions of communication with the equipment.

REFERENCES

[1] V. Aleinikov, S. Paschenko, “Using commercial SCADA in control system for ECR CyLab.” PCaPAC 2000, Hamburg, Germany.

[2] V. Aleinikov, A. Nikiforov, “QNX based software for particle accelerator control system of FLNR”, NEC’2007, Varna, Bulgaria.

[3] National Instruments. *LabVIEW Datalogging and Supervisory Control (DSC) Module*, http://www.ni.com/labview/labviewdsc/

[4] National Instruments. *Network Variable Technical Overview*, http://www.ni.com/white-paper/5484/en/