CUSTOM HMI FOR COMMERCIAL SCADA

V. Aleinikov, G. Dreyzina, A. Nikiforov, JINR, Dubna, Russian Federation

Abstract

A new cyclotron complex for the production of radioactive nuclear beams (DRIBs – Dubna Radioactive Ion Beams) was developed and manufactured in Flerov's Laboratory of Nuclear Reactions. The control system for beam transportation system uses commercial SCADA FlexControl running under RTOS QNX4. Because of good interaction possibility it was able to develop self-made visualization server. It has highly improved flexibility and look and we use it instead of the regular one. Basic ideas for integrating custom HMI and commercial SCADA are described.

1 INTRODUCTION

Three years ago with the appearance of new projects DRIBs and CyLab (Slovakia) where Flerov's Laboratory of Nuclear Reactions (FLNR) contributes most to the development, it has been decided to renew software development environment. Upon market analysis we have selected commercial real-time OS QNX and SCADA (Supervisory Control and Data Acquisition) FlexCtrl [1].

QNX is UNIX-style scalable, multi-user, multi-tasking, real-time, network and POSIX-compliant operating system. QNX provides source code for most components with only the kernel and core portions remaining protected. After using QNX for 6 months we were certain that we could write a working device driver for this OS quickly and with high degree of robustness and reliability.

FlexCtrl is a process control system for the automation of technological processes. It is modular and extremely scalable. FlexCtrl is a pure software system, no specific or special hardware is needed for process control. The interface to the system is open and the user has the possibility, with the use of available hardware drivers, adding their own hardware to the system. After a week of training we were able to develop protocol driver for custom PLC and connect it to the working SCADA project with automation algorithm, graphical visualization and control.

The interfaces for FC are very flexible, and the user can develop the relevant drivers for the process connection for its own process hardware without difficulty.

The only trouble we had was poor flexibility and look of graphical images of the object visualization library. Because of good interaction possibility we were able to develop self-made visualization server and use it instead of the regular one.

2 DEVELOPMENT TOOLS

In order to develop different segments of the project we have to use several tool kits:

?? *FlexCtrl Project Development Tool* to build signal database and to describe the specific characteristics of the technological process

?? *Photon Application Builder* to design graphical user interface

?? *Watcom Development Tools* for writing visualization server and device drivers

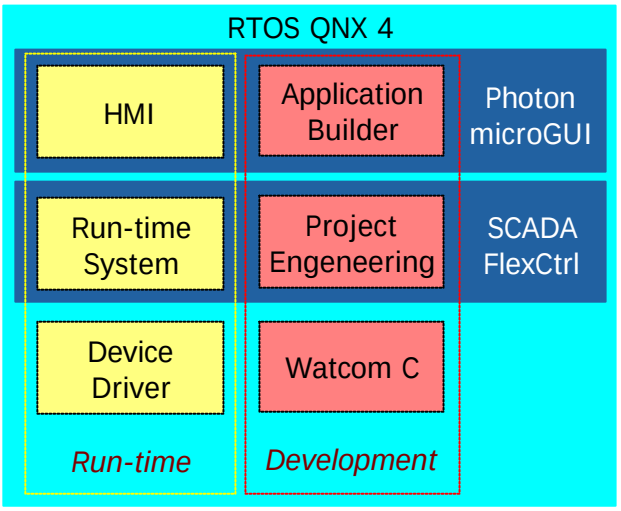

Fig. 1. Basic structure of the software development

2.1 FlexCtrl Project Development Tool

All parts of the FlexCtrl application can be managed with the project engineering system, that configures process model (process variables with all characteristics), creates process images, unites all of the project databases and files under a selected project name. The system includes Process Model Editor, Graphics Editor, Network Configurator, User Administrator, Driver, Visualisation and Run-time Compilers. Project and Run-time Installation tools complete the project development with generating files for the start of run-time components.

2.2 Photon Application Builder

Photon is a "second generation" GUI, with a tiny footprint but with an advanced design that gives it Windows NT look and feel, but with better performance and much cleaner programmer model. Photon's development toolkit consists of a complete collection of libraries, sample applications, self-contained widgets and

on-line documentation. The tool kit also offers a GUI development tool called the Photon Application Builder (PhAB), which allows an application programmer to create prototype GUIs, without writing a single line of code; build an entire GUI, by pointing and clicking; create applications with a consistent look and feel.

Since FlexCtrl Graphics Editor and graphical library does not meet our requirements we have decided to develop custom image library. For creating Human to Machine Interface (HMI) we used Photon Application Builder instead of the FlexCtrl Graphics Editor.

2.3 Watcom Development Tools

The QNX package includes the Watcom highly optimizing compiler (C/CPP) and tool set, including Watcom Debugger.

3 IPC AND DATA FLOW

QNX depends on the exchange of discrete packets of information – *messages* – to handle virtually all inter-process communication. Message passing lies at the heart of the operating system's microkernel architecture, giving the OS its modularity [2]. This paradigm applies to all levels of programming, from device drivers to file system and LAN. From the standpoint of a user's process there is no difference between a local call and a call from the network and hence all resources on all network nodes are transparently available everywhere on the network.

FlexCtrl strongly uses this message passing technique as for internal run-time communication as for interfacing to device drivers. The components of FlexControl can be partitioned, installed and run on several computers of standardized PC network [3]. Every job of some significance in FC is handled through a single computational process (server). FlexControl comprises a long list of these tasks:

?? Process administration and process supervision
?? Real-time database (RTDB)
?? Local graphic server
?? Alarm server
?? Message server
?? Long-term processing
?? Calculation server (Soft SPC tasks)
?? Protocol drivers
?? … and other tasks

The tasks are able to communicate and synchronize with one another. The data from the transmitting process are exported via message passing over the network and imported directly into the address area of the receiver process. Deadlocks and blocks do not occur in any form in FlexCtrl.

The configured coupling parameters of process variables (PV) and drivers are retrieved by respective driver process when started up.

3.1 Monitoring

For process monitoring device driver cyclically reads device status and compares it with the old one saved in its memory. In case of any changes the driver writes new values to the real-time database (see Fig. 2). Control Server manages real-time database access and trigger off coupled server for every instance the PV is changed in the real-time database. Calculation Server processes these events, converts ADC codes (raw values) to real physical (operational) values and writes them to RTDB. Visualization Server gets information about changes of operational values as well. It appends these events of data change to the message queue (FIFO) driven by Message Queue Server. The Human to Machine Interface (HMI) cyclically checks for the messages in queue, reads them out and represent by graphical image or text on the screen.

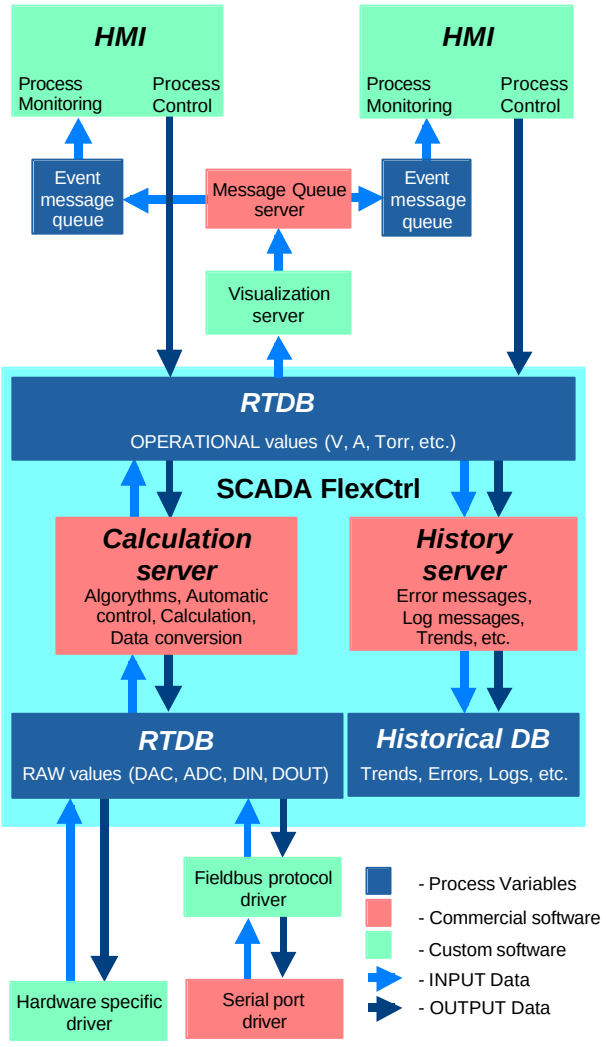

Fig. 2. Data flow

3.2 Control

Operator controls the process via HMI. He presses buttons, enters new values, etc. HMI writes new values directly to RTDB. The fact the operational value (Ampere, Volt, Torr, etc.) was changed causes Calculation Server to convert it to DAC code and write it to RTDB (raw values part). Next step Control Server sends message with the new DAC code to the coupled driver. Device driver receives new data from Control Server and issues write command to the device.

4 VIZUALIZATION SERVER AND HMI

Since FlexCtrl Graphics Editor have not meet our specific needs it was decided to create custom HMI development tool. It uses Photon Application Builder as graphical editor, object configurator and application compiler. We have developed a library of functions, which we build into application to control the screen and the keyboard, as well as any input from Visualization Server. In order to manage message queues for every HMI client in the network we have developed Visualization Server.

At the engineering step developer designs GUI by arranging active images (widgets) on the screen. Every widget represents process variable in text or graphical form. Every active image has property that contains the name of the coupled variable in RTDB. At final step Application Builder compiles GUI and custom library into the HMI application.

At start up HMI connects to the RTDB and reads coupling information from widgets to link them to the RTDB in its memory. Next step it requests Visualization Server to open message queue for it. This queue is a FIFO buffer of events of data changes managed by Visualization Server.

Widget picture can be imported from a file of BMP or GIF format. For better look and understanding of control process we have used 3D images created in Solid Edge package.

HMI allows analyze process data in real-time trend, store and retrieve a set of variables to repeat important system modes. Reports can be configured, printed and exported in text form.

5 CONCLUSION

Using commercial SCADA for a control system decreases total project development time, unifies data processing and allows to concentrate more on visualization and automation algorithms. It also increases reliability and endurance of the software since the core of the system is well optimized and tested.

We have had no problems yet with the stability of selected solution (QNX and FC). Because of good interaction possibility we were able to develop self-made visualization server (HMI) and use it instead of the regular one. The most important point for this kind of software is stability and we surely able to have it with this solution.

6 REFERENCES

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

[2] Rob Krten, “Getting started with QNX 4. A Guide for Realtime Programmers”. PARSE Software Devices, 1998.

[3] FlexControl - System Architecture Manual. BitCtrl GmbH. October 1998.