CS - A Control System Framework for Experiments at GSI

Dietrich Beck and Holger Brand, DVEE, GSI

1 Introduction

In many cases, control systems have been developed for specific experiments and the re-usability of such systems was limited. Another approach is the development of a general control system framework. A framework can be applied to a large variety of experiments by implementing a few experiment specific add-ons. During the past one and a half years, such a framework has been developed within DVEE at GSI. In the first stage, the framework aims at experiments with up to 10,000 process variables. In autumn 2002, that framework has gone into production at the SHIPTRAP[1] facility.

2 Requirements and Tools

In most cases, control systems are maintained and developed further by PhD students. Only one development tool that is easy to learn should be used. Hardware and drivers must be available commercially. The connectivity to field bus systems like CAN, CAN-OPEN, Profibus, Firewire or GPIB as well as the employment of motion and vision products must be easy. LabVIEW from National Instruments fulfills these requirements on a Windows platform.

The control system must be highly flexible. Changes between operational states may be required by the user during run-time. This implies that the number of used devices as well as the device types will be selected on the fly. An object oriented approach eases the implementation of such a system. Within LabVIEW this is made possible by the ObjectVIEW toolkit from Vogel Automatisierungstechnik[2]. However, inheritance of classes is not as comfortable as in C++. Thus, the number of levels of inheritance should be kept small which in return results in large classes with quite a few methods.

A real-time system is not required. If needed, this will be implemented in hardware. Nevertheless, the reaction time of the system must be short ($\approx 10\,\mathrm{ms}$), so that complex sequences may be realized within an acceptable time. This requires an event driven communication which is possible in LabVIEW.

Trending and alarm management of a few thousand process variables is realized by using the Datalogging and Supervisory Control (DSC) module of LabVIEW. Also, this module provides the connectivity to the OPC (OLE for Process Control) world.

Often, control systems must be distributed over a larger area. Event driven communication between different nodes is implemented based on TCP/IP.

3 Classes of the Framework

The "BaseProcess" class is the base class of the system. Almost all classes of the framework are direct children of the BaseProcess class. Its main features are the following. First, it has two threads. The first thread serves for handling incoming events. The second thread can be used to periodically perform actions. Second, status and errors for all threads are logged. Third, there exist methods for event driven communication. Fourth, there is the possibility to query the methods of a class together with input and output parameter types.

The "SuperProc" class serves for creating and deleting instances of classes during run-time. A database contains information on all instances of all classes together with instance specific parameters. These data are retrieved from the database prior to instantiation.

The "DSCIntProc" class serves as an interface to and encapsulates the DSC module of LabVIEW. All trending and alarming is done via the DSCIntProc class. Presently, only one instance of the DSCIntProc class is used, even if the control system is distributed over several nodes. This is possible due to the event driven communication between different nodes.

Next to the core classes described above, there exist about 10-20 other classes. Mostly, these are classes for specific instruments like arbitrary functions generators as well as timing, data acquisition and motion devices.

4 Status and Outlook

The control system framework is functional and seems to be robust. It is GPL licensed and available for download[3]. The time that is needed to send an event to an instance and to receive an answer from that instance via a second event is typically about 3 ms on a local node and 15 ms across nodes. With about 100 instances per node, the CPU load is less than 10 %. Quite some RAM is required per node, since each instance takes a few MBytes. This is due to the memory management of LabVIEW and the fact that there is almost no optimization when compiling code within LabVIEW. By increasing the number of computer nodes, the framework can easily be scaled.

Presently, the framework is in operation at SHIPTRAP [1] and in commissioning at four other experiments, ISOLTRAP[4], motion control in Cave A[5], LEBIT[6] and PHELIX[7]. It is expected that the number of classes available will increase dramatically during the year 2003. It is planned to extend the framework to a SCADA system. Although some SCADA features like trending and alarming are partly included already, others like user management, access control and security have to be added.

References

[1] J. Dilling et al., Hyp. Int. 127 (2000) 491.
[2] http://www.vat.de/.
[3] http://labview.gsi.de/CS/cs.htm.
[4] G. Bollen et al., Nucl. Instr. Meth. A368 (1996) 675.
[5] Atom. Phys. Contributions, GSI Scient. Rep. 2001.
[6] S. Schwarz et al, Eur. Phys. J. submitted 2001.
[7] R. Bock et al., Inertial Fusion Sciences and Applications 99, C. Labaune, W.J. Hogan, K.A. Tanaka eds. Elsevier Publishing (2000) 703.