Ein allgemeines Kontrollsystem für Experimenteinrichtungen der GSI

Dietrich Beck und Holger Brand,
GSI-Darmstadt, DVEE, Planckstr. 1, D-64291 Darmstadt

Kurzfassung

Die GSI [1] ist eine Großforschungseinrichtung für Schwerionenforschung. Hier werden mit einem Schwerionenbeschleuniger sowohl Grundlagen der Physik als auch angewandte Fragen benachbarter Gebiete untersucht. Die Kontrollsystemgruppe der Abteilung Datenverarbeitung und Experimentelektronik (DVEE) gibt den Experimentiergruppen Hilfestellung bei der Planung und Implementierung von Experimentkontrollsystemen.
In der Vergangenheit sind Kontrollsysteme oftmals nur für ein spezielles Experiment entwickelt worden und die Wiederverwertbarkeit für andere Experimente war entsprechend gering. Eine andere Möglichkeit ist die Entwicklung eines allgemeinen Kontrollsystems, das durch einige wenige experimentspezifische Erweiterungen an nahezu beliebige Experimente angepasst werden kann. Hier wird ein solches allgemeines Kontrollsystem vorgestellt, das in den letzten eineinhalb Jahren an der GSI entwickelt wurde und seit Herbst 2002 am SHIPTRAP Experiment [2] zum Einsatz kommt.

Abstract

GSI is a heavy ion research center. The laboratory performs basic and applied research in physics and related natural science disciplines using a heavy ion accelerator facility. The control system group of the department for Computing and Experiment Electronics (DVEE) helps the experiments in developing and implementing experiment control systems.
In the past, such control systems have in many cases been developed for a specific experiment. The reusability of such a system is limited. Another possibility is the development of a general control system framework. Such 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 at GSI. In autumn 2002, that framework has gone into production at the SHIPTRAP experiment.

Einleitung

Für Experimentkontrollsysteme an Beschleunigereinrichtungen gibt es ganz andere Rahmenbedingungen als für Kontrollsysteme in der Industrie. Auf der einen Seite werden Betrieb, Wartung und Weiterentwicklung üblicherweise von Doktoranden übernommen, die ein Kontrollsystem aber nur für einen Zeitraum von etwa zwei bis drei Jahren betreuen können. Eine Kontinuität auf der Entwicklerseite ist nur selten gewährleistet. Auf der anderen Seite werden die Experimente immer weiterentwickelt. Das betrifft nicht nur deren Ablauf, sondern bedingt auch eine ständige Zunahme der verwendeten Geräte und Gerätetypen. Der Umfang und die Funktionalität der Kontrollsysteme muss daher ständig erweitert werden.

Hier wird ein allgemeines Kontrollsystem auf LabVIEW-Basis vorgestellt. Dabei muss im Hinblick auf den geplanten Ausbau der GSI auch die wichtige Frage geklärt werden, in wieweit bei den geplanten Großexperimenten die Kontrollsysteme noch auf der Basis von LabVIEW realisiert werden können.

Anforderungen und Werkzeuge

Es darf nur ein Entwicklungswerkzeug geben, das einfach erlernbar sein muss. Gerätehardware und Treiber müssen kommerziell verfügbar sein. Die Anbindung an die Entwicklungsumgebung über Feldbussysteme wie CAN, CAN-OPEN, Profibus, Firewire oder GPIB sowie der Einsatz von Motion und Vision Produkten sind selbstverständlich.

LabVIEW von National Instruments erfüllt diese Anforderungen auf einer Windows Plattform.

Ein Kontrollsystem muss hohe Flexibilität besitzen. Die Betriebszustände müssen vom Bediener gewechselt werden können und auch während des Betriebs in gewissen Grenzen frei konfigurierbar sein. Das bedeutet, dass sowohl die Zahl der Geräte eines Gerätetyps als auch die verwendeten Gerätetypen erst zur Laufzeit feststehen. Für größere Projekte wie dieses empfiehlt sich aus Gründen der Übersichtlichkeit und Wartbarkeit ein objektorientierter Ansatz. Daher wird zusammen mit LabVIEW das Werkzeug ObjectVIEW der Firma Vogel Automatisierungstechnik eingesetzt.

Das Kontrollsystem hat unmittelbar keine Echtzeitanforderungen. Diese werden in Hardware realisiert. Dennoch müssen die Reaktionszeiten des Systems so kurz wie möglich sein (≈ 10 ms), damit komplexe Abläufe ausgeführt werden können. Das bedeutet, dass die Komponenten des Kontrollsystems über Ereignisse (Events) gesteuert miteinander kommunizieren müssen. LabVIEW bietet hier mit den Message-Queues, den Notifications, den Semaphoren und den Rendezvous ein vorzügliches Spektrum an Werkzeugen.

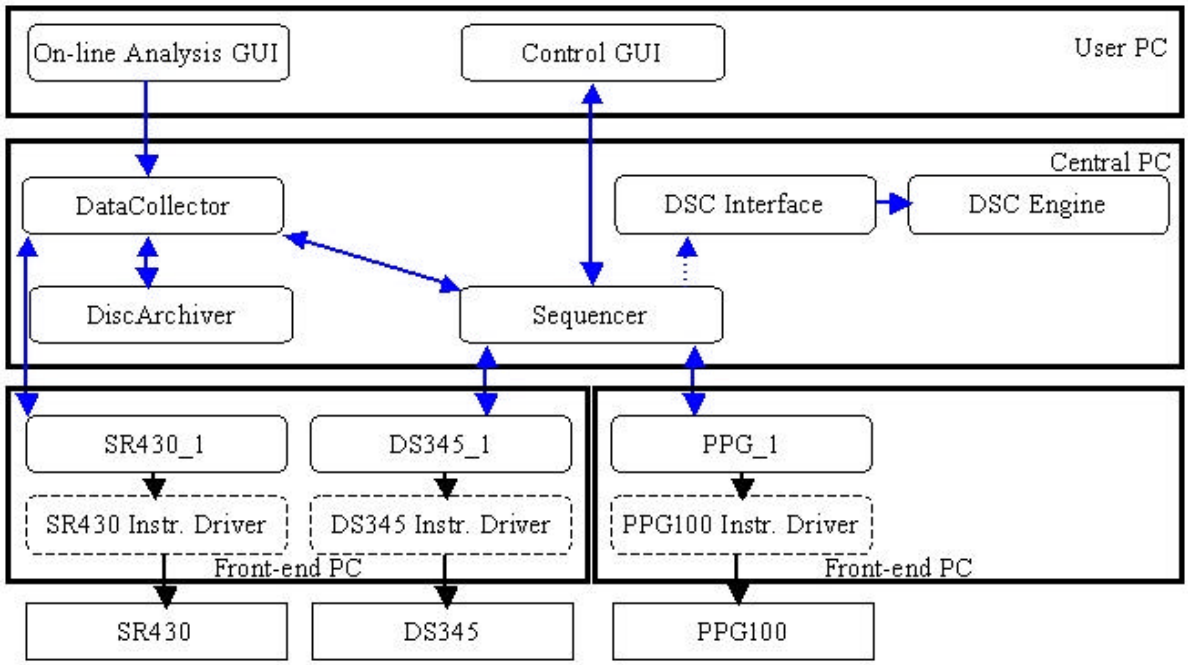

Bild 1: Beispiel für ein einfaches Kontrollsystem

Kleinere Experimente haben etwa einige tausend Prozessvariablen, deren Werte protokolliert und mittels Alarm-Management überwacht werden müssen. Dies wird durch den Einsatz des Datalogging and Supervisory Control (DSC) Moduls realisiert, das auch den Anschluss an die OPC-Technologie herstellt.

Experimente sind oft über große Bereiche verteilt. Die Experimentatoren können aus Sicherheitsgründen nicht immer direkt am Experiment arbeiten. Daher müssen Events über Rechnergrenzen hinweg verschickt werden können. In der gewünschten Form steht diese Funktionalität in LabVIEW 6.1 nicht zur Verfügung und wird über TCP/IP realisiert.

Beispiel für ein Kontrollsystem

Bild 1 zeigt ein einfaches Beispiel für die Struktur eines Kontrollsystems. Es gibt vier Rechner. Einen für die GUIs der Benutzer, einen zentralen Rechner und zwei Knoten mit den E/A-Geräten. Rechtecke mit runden Kanten und durchgezogenen Linien sind eigenständige

Objekte, die über Events (Pfeile) miteinander kommunizieren. Die Rechtecke mit gestrichelten Linien stellen die Gerätetreiber von LabVIEW dar. Geräte sind als normale Rechtecke gekennzeichnet. Lediglich drei Objekte, nämlich der „Sequencer“ für die Ablaufsteuerung und die beiden GUIs sind experimentspezifische Erweiterungen. Der Rest ist Bestandteil des allgemeinen Kontrollsystems. Das „ControlGUI“ sendet die zu verwendenden Parameter an den „Sequencer“. Dieser erzeugt die Objekte zu den einzelnen Geräten, z.B. „DS345_1“, parametriert diese und startet einen sogenannter Pattern Generator „PPG100_1“. Dieser steuert den Ablauf eines Experimentes durch Ausgabe von Bitmustern mit einer Genauigkeit von 100ns. Anschließend stößt der „Sequencer“ eine Instanz „DataCollector“ an. Diese liest die Daten von DAQ-Geräten und puffert diese. Ein sogenannter „DiscArchiver“ ist als Client mit dem „DataCollector“ verbunden und schreibt die Daten auf eine Festplatte. Ein weiterer Client des „DataCollector“ ist schließlich ein „On-line Analysis GUI“, wo ein Teil der Daten zur Kontrolle des Experiments dargestellt wird. Die DSC-Engine von LabVIEW wird durch ein „DSC Interface“ gekapselt. Alle Objekte, hier beispielhaft durch einen gebrochenen Pfeil für den Sequencer angedeutet, schreiben Fehler und Werte über den „DSC Interface“ in die Echtzeitdatenbank der DSC-Engine.

Wichtig ist die enorme Flexibilität. Da alle Botschaften und Daten über Events verschickt werden und die Objekte nicht als VIs fest miteinander verdrahtet sind, lässt sich z.B. leicht das Objekt „Sequencer“ austauschen und die komplette Ablaufsteuerung ändern. Ebenso kann der Benutzer zur Laufzeit die Zahl der verwendeten Geräte eines Typs ändern oder auch Gerätetypen austauschen. Beim Verschicken eines Events muss lediglich der Name des Empfängers und eventuell der Name des Events geändert werden. Sollen keine Messdaten archiviert werden, so könnte man auf den „DiscArchiver“ verzichten. Umgekehrt könnte man auch zwei „DiscArchiver“ verwenden. Der erste archiviert die Daten auf einer Netzwerkplatte, der zweite schreibt eine Sicherheitskopie auf die lokale Platte.

Die Basisklasse „BaseProcess“

Hat man einen bestimmten Gerätetyp, z.B. einen Funktionsgenerator DS345 der Firma Stanford Research Systems, so werden die Eigenschaften dieses Gerätetypes durch eine Klasse repräsentiert. Werden fünf dieser Geräte konkret eingesetzt, so werden zur Laufzeit fünf Instanzen der Klasse DS345 erzeugt. Jede Instanz ist einem bestimmten Gerät zugeordnet, während die Klasse die Methoden bereitstellt, mit denen der Gerätetyp DS345 charakterisiert wird. Neben den gerätespezifischen Methoden sollen die Klassen aller Gerätetypen auch über gemeinsame Basisfunktionalitäten verfügen, mit denen z.B. die Kommunikation der Instanzen untereinander ermöglicht wird. Solche grundlegenden Eigenschaften und Funktionen werden in der Basisklasse „BaseProcess“ zur Verfügung gestellt. Alle anderen Klassen des Systems sind Erweiterungen der BaseProcess Klasse und erhalten deren Eigenschaften durch Vererbung. Die wichtigsten Eigenschaften sind:

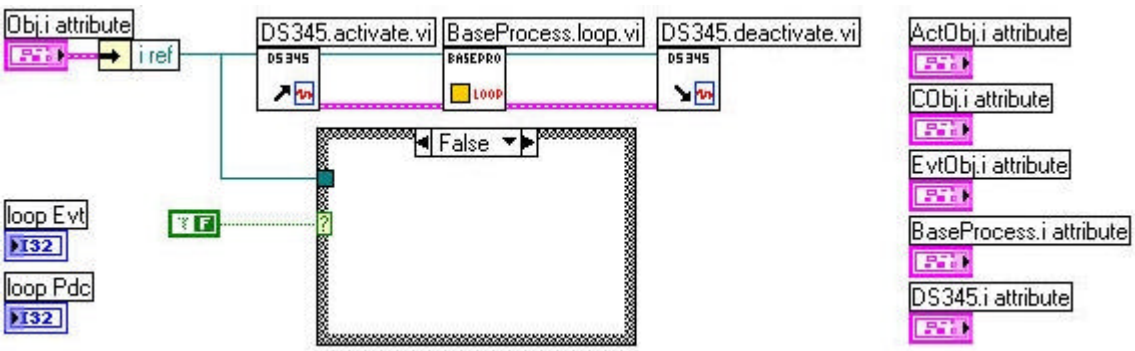

Bild 2: VI-Template der Klasse „DS345“

1. Zwei Threads. Der erste dient zur Handhabung von Events. Bei Eintreffen eines Events wird eine entsprechende Aktion ausgeführt. Im zweiten Thread werden Aktionen periodisch ausgeführt. Optional kann ein dritter Thread für eine flache Zustandsmaschine genutzt werden.
2. Protokollierung von Status und Fehlerzustände aller Threads
3. Methoden zur ereignisgesteuerten Kommunikation
4. Abfrage der vorhandenen Methoden und deren Ein- und Ausgabeparameter

Geräteklasse DS345

Die Geräteklasse DS345 soll exemplarisch für alle Klassen erklärt werden. Durch Vererbung der Klasse BaseProcess entsteht das in Bild 2 gezeigte VI-Template und eine Bibliothek mit gerätetypeigenen Methoden, die erstellt werden müssen. Das VI-Template besteht nur aus drei VIs. In den Methoden "DS345.activate" und "DS345.deactivate" können Aktionen zur Herstellung definierter Ausgangs- und Endzustände definiert werden. Die Funktionalität der Klasse „BaseProcess" ist in der Methode „BaseProcess.loop" gekapselt. Wie wird nun z.B. eine Methode „DS345.Reset" innerhalb Klasse DS345 angestoßen? Der Event-Thread innerhalb der Methode „BaseProcess.loop" wartet auf des Event „Reset". Innerhalb des Event-Threads wird dann die Methode „DS345.ProcCases angestoßen, die in Bild 3 dargestellt ist. Innerhalb dieser Methode wird im Fall „Reset" die Methode „DS345.Reset" aufgerufen. Diese Methode wiederum ruft lediglich das entsprechende VI des LabVIEW Gerätetreibers auf.

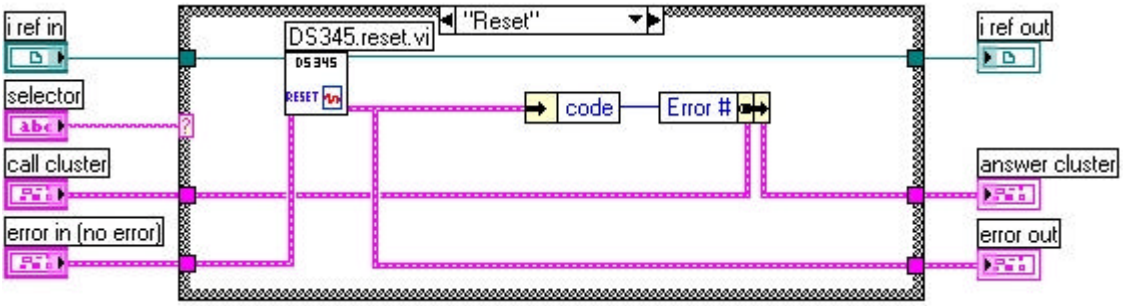

Bild 3: Aufruf von „DS345.Reset" aus der Methode „DS345.ProcCases"

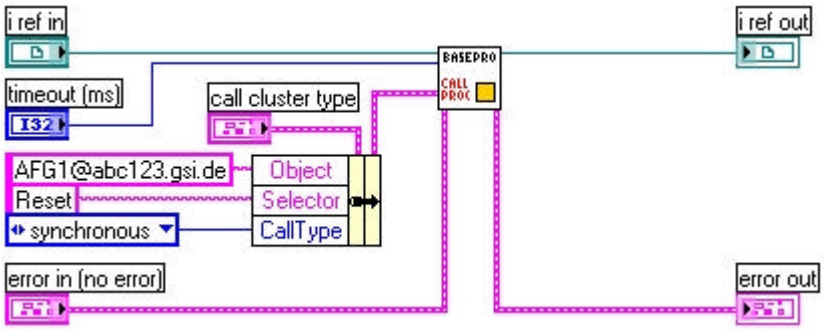

Bild 4: Aufruf der Methode „Reset" eines Objekts „AFG1"

Ereignisgesteuerte Kommunikation

Instanzen von Klassen kommunizieren über Events. Dazu stellt die Klasse BaseProcess die Methode „CallProcess" zur Verfügung. Bild 4 zeigt dies an einem Beispiel. Hier wird die Methode „Reset" des Objekts „AFG1" gerufen. „AFG1" könnte z.B. eine Instanz der Klasse „DS345" sein. In diesem Beispiel befindet sich das Objekt nicht auf dem lokalen Rechner, sondern auf dem Knoten „abc123.gsi.de", der durch den Zusatz „@abc123.gsi.de" spezifiziert wurde. Es gibt drei verschiedene Möglichkeiten, Objekte zu rufen.

- „Einfach“. Der Sender schickt eine Nachricht an den Empfänger und bekommt keine Antwort.
- „Asynchron“: Im Unterschied zu „Einfach“ verschickt der Empfänger eine Antwort an ein drittes Objekt, das bereits vom Sender spezifiziert wurde.
- „Synchron“: Der Sender wartet auf Antwort vom Empfänger, der eine Antwort an den Sender zurückschickt.

Der Vorteil von „Synchron“ liegt darin, dass der Empfänger eine direkte Rückmeldung über den Erfolg der beim Empfänger angestoßenen Aktion bekommt. Der Nachteil ist, dass der entsprechende Thread des Senders während des Wartens blockiert ist. Selbstverständlich können Nachrichten über Knotengrenzen hinweg an andere Rechner verschickt werden. Dabei erfolgt eine Umsetzung auf TCP/IP „peer to peer“. Darum muss sich der Programmierer jedoch nicht kümmern, da diese Funktionalität vom allgemeinen Teil des Kontrollsystems gekapselt zur Verfügung gestellt wird.

Status

Die derzeitige Version (Stand November 2002) ist 1.00d. Die Software ist GPL lizenziert. Die Webseite des Rahmenwerks ist http://labview.gsi.de/CS/cs.htm, wo die Software auch zum Download bereitsteht.
Wird ein Objekt synchron gerufen, so dauert dies lokal weniger als 3ms. Dies beinhaltet bereits die gesamte Ereignis-Protokollierung und das Alarm-Management. Geschieht dies über Rechnergrenzen hinweg, so steigt die Zeit auf etwa 10-15ms. Die Daten beziehen sich auf PC mit 700MHz, PIII und Windows NT. Sind 100 Instanzen von Klassen erzeugt, so ist die CPU Last immer noch kleiner als 10%. Die Rechner sind jedoch großzügig mit Speicher auszustatten. Eine Instanz benötigt einige Mbyte RAM. Dies liegt an der Notwendigkeit, an vielen Stellen sogenannte „reentrant“ VIs zu benutzen. Da LabVIEW 6.1 für jeden Aufruf auf ein „reentrant“ VI intern ein Kopie desselben erzeugt, ist der Verbrauch an Ressourcen entsprechend hoch. Ein entsprechender Änderungswunsch wurde an National Instruments gemeldet. Der Kern des Systems läuft stabil. Die Kommunikation ist auch über Rechnergrenzen hinweg robust.

Zusammenfassung und Ausblick

An der GSI wurde ein allgemeines, ereignisgesteuertes, verteiltes und objektorientiertes Kontrollsystem entwickelt. Diese kann durch wenige applikationsspezifische Erweiterungen für beliebige Experimente mit bis zu einigen tausend Prozessvariablen eingesetzt werden. Wartung und Weiterentwicklung bei den Experimenten werden deutlich vereinfacht. Seit Oktober 2002 ist das hier vorgestellte allgemeine Kontrollsystem SHIPTRAP [2] im Einsatz. Bei drei weiteren Experimenten wird an der Inbetriebnahme des Systems gearbeitet: ISOLTRAP [3] am CERN (Genf/Schweiz), PHELIX [4] an der GSI (Darmstadt/Deutschland) und LEBIT [5] an MSU (East Lansing/USA). Die beiden letztgenannten Experimente befinden sich noch im Aufbau und sollen im Jahr 2003 die ersten Daten liefern.

Literatur

[1] GSI-Darmstadt Planckstr. 1, D-64291 Darmstadt, http://www.gsi.de/
[2] J. Dilling et al., Hyperfine Interactions 127 (2000) 491-496
[3] G. Bollen et al., Nucl. Instr. Meth. A368 (1996) 675
[4] B. Becker-de Mos et al., Photonics Science News 5 (1999)
[5] S. Schwarz et al., Proc. EMIS-14, Victoria, BC, 2002, Nucl. Instr. Meth., im Druck