The Phelix Control System based on *CS* Framework 3.0

H. Brand, D. Beck, S. Götte, M. Kugler

Gesellschaft für Schwerionenforschung mbH, Planckstr. 1, 64291 Darmstadt, Germany

Motivation

The Petawatt High Energy Laser for Ion eXperiments will offer the unique combination of a high current, high energy (GeV/u) heavy-ion beam with a powerfull laser beam thus providing the opportunity to investigate a variety of fundamental science issues in the field of atomic physics, nuclear physics, and plasma physics.

The PHELIX Control System (*PCS*) is based on the *CS* framework. About 40 additional classes were developed for the *PCS* and ~250 objects are running on 13 PC's publishing ~10000 process variables.

The *PCS* has been upgraded to version 3.0 recently. In *CS* 3.0 the entire communication layer has been changed to DIM (Distributed Information Management), which is a light weight protocol for inter-process communication based on TCP/IP.

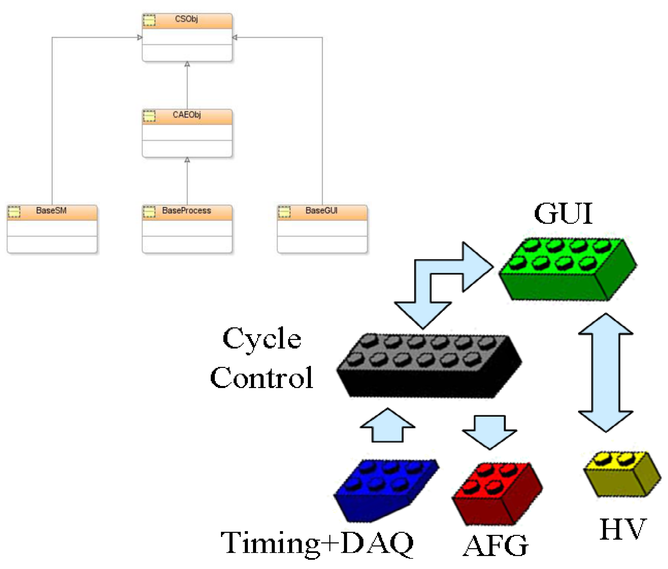
The *PCS* is designed to make use and profit from the concept of named services. Clients may receive information from a service (observer pattern) or may send a command to a server (command pattern). By these means the implementation of the *PCS* behaviour with hierarchical state machines was eased.

CS Framework 3.0

CS is a framework that can be used by many experiments. *CS* is a multi-threaded, event driven, object oriented and distributed framework with SCADA functionality based on LabVIEW and DIM. An experiment control system can be developed by combining the *CS* framework with experiment specific add-ons. *CS* is supported on MS Windows, Linux and the realtime OS Pharlab (LabVIEW RT).

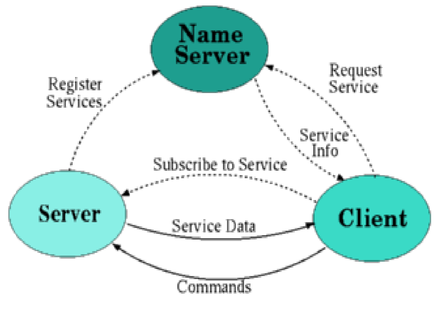
The *CS* framework provides an object oriented approach for standard LabVIEW by using *Template-VI*'s to create unique object references and threads and *Functional Global Variables* (uninitialized Shift-Registers) to maintain the object attribute values protected by *Semaphores*. DIM is used for the local and network communication layer. The *CS* framework provides base classes for GUI's, State Maschines or active processes that can react on events, e.g. BaseProcess can subscribe to DIM services and commands to ensure a common event data structure.

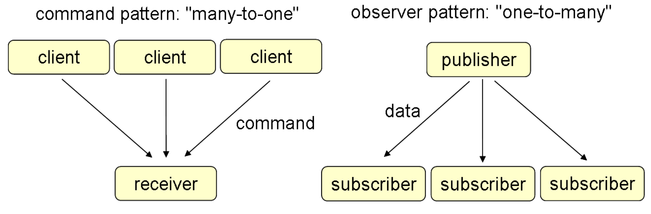
Each class has a well defined responsibility. Objects are instances of such classes which are created at runtime. Due to the common event mechnism any objects can be dynamically combined to a concret distributed control system like LEGO bricks. A very simpe system is shown in the picture below.



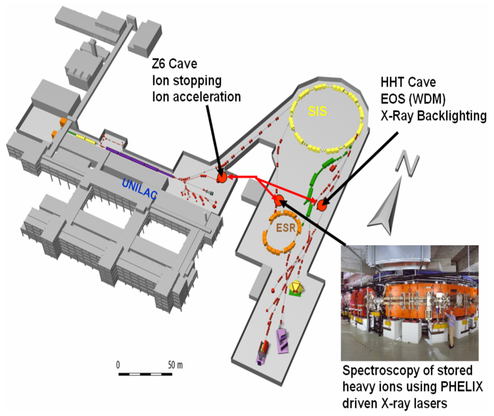
DIM

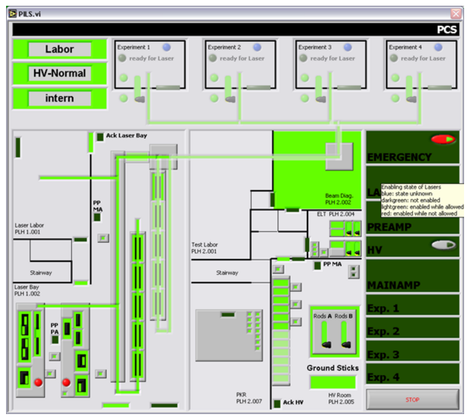
The Distributed Management System developed at CERN was adoped by the *CS* framework to implement the communication layer. DIM implements the Publisher-Subscriber and Command Patterns. A DIM Server registers named services and command at the DIM Name Server at startup. A DIM Client asks the DIM Name Server for service and command information and connects directly to the corresponding servers. A client will automatically reestablish a connection to a server in case of server restart or network interruptions.

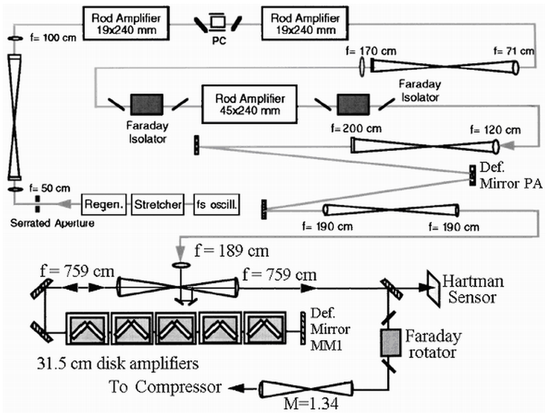




The PHELIX Facility





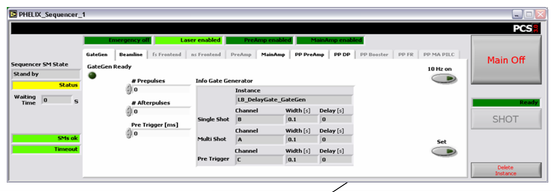


The PHELIX Control System Architecture

Operation Layer

The PHELIX Sequencer is an example for an object in the operation layer. It provides a graphical user interface (GUI) that enables the PHELIX operator to prepare and deliver a PHELIX laser shot to an experiment.
The sequencer requests the remote controls of objects in the application layer. Their local GUI becomes disabled to avoid concurrent modifications. The sequencer then monitors the status information and sends configuration commands or state change requests to objects in the application layer.

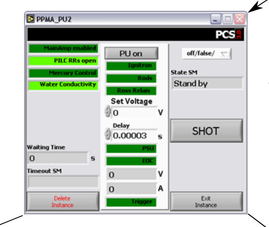
PHELIX Sequencer



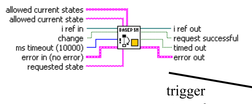
Application Layer

Application layer classes can inherit from BaseSM, the base class for state machines.
They are typically aggregations that control elementary devices or other application layer objects. Their behaviour is driven by a state machines as shown on the lefthand side. The implementation with LabVIEW is shown on the right hand side.
- Such an object requests the remote control of child objects and creates them, if not already existing, when it becomes instantiated.
- The local control GUI of a child object becomes disabled, so the object has exclusive control.
- An application layer object can be operated locally by using its GUI or remote controlled by higher level application layer objects or from the operation layer.
- Child objects are set to local control and will be released by the parent object on its destruction.

control



BaseSM.requestStateChange.vi

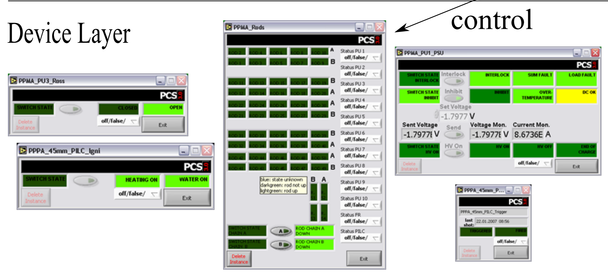


trigger state change

Device Layer

control

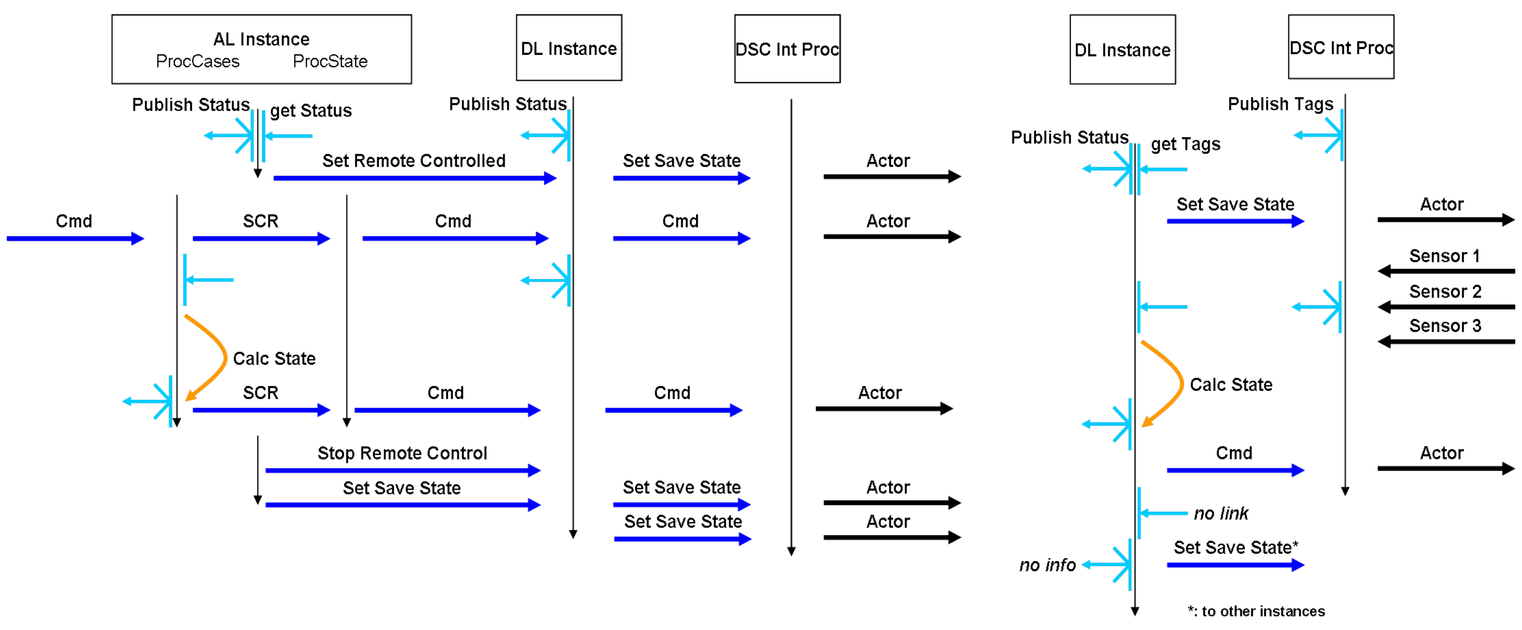
share



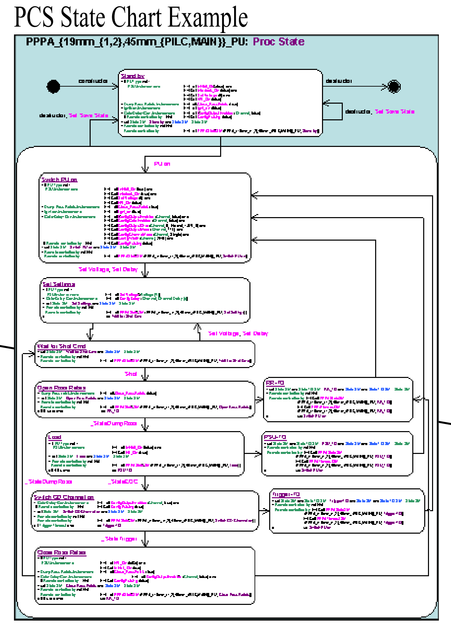
Each device type is represented by a class and each device by an object. Each device object has its own GUI for direct or standalone operation. Its state, e.g. controlled or local, is implemented as state chart.

Typical Scenarios

AL: Application Laye; DL Device Layer; DSC Int Proc: Interface to the Data Logging and Supervisory Control Module of National Instruments which also serves as OPC Gateway; Cmd: Command; SCR: State Change Request.

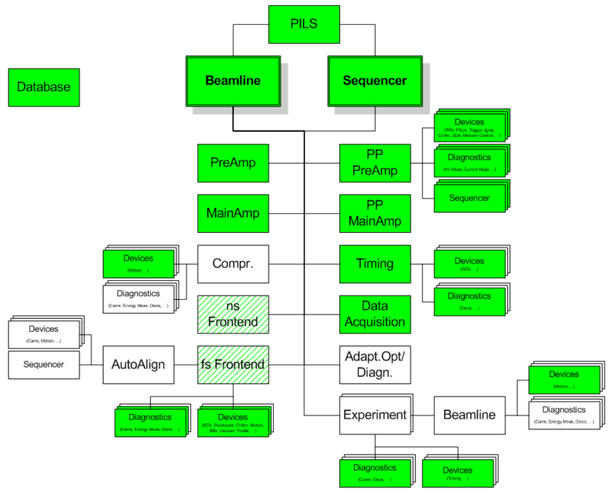


PCS State Chart Example



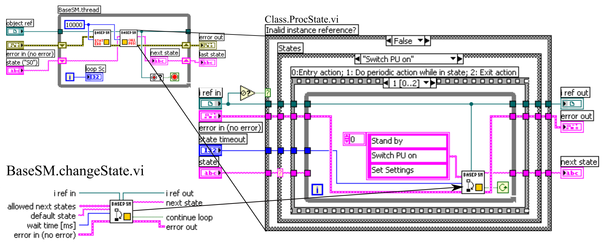
PCS Components

The PHELIX Control System is composed hierarchically of many subsystems and modules. The initial laser pulse is produced in the ns- or fs-Frontend laser systems. It will then be amplified in the Preamplifier (PreAmp) and Mainamplifier (MainAmp). There are two Pulsed Power systems (PP PreAmp and PP MainAmp) that provide the pump energy. Each subsystem controlls devices, arbitrary function generators, power supplies, motors etc., and diagnostics like oscilloscopes, cameras, power meters etc. Such subsystems are running on their own local control nodes.
Other subsystems like Timing or Data Acquisition are running on separate nodes since they have to control or readout several other subsystems.
PILS (PHELIX InterLock System), Beamline and Sequencer integrate all other subsystems to one global PHELIX Control system each focusing on different aspects.



CS State Chart Implementation

The BaseSM.thread.vi calls Class.ProcState.vi which implements the behaviour of the child class and publishes its state. Class.ProcState.vi contains a case for every state. A sequence structure ensures that actions on state entry/exit are performed before/after the periodic actions, if any, are executed while in state. A state change can be requested externally, from BaseSM.requestStateChange.vi, or internally by specifing the default state at BaseSM.changeState.vi in the periodic loop.



BaseSM.changeState.vi

BaseSM.changeState.vi changes the state of the state machine of THIS object. The requested state is obtained from the object attributes. The new state will be returned. If the waiting time is expired, the object changes to the default state if the default state is specified. If the waiting time is expired and the default state is not specified, the state is not changed. This method will check if the transition is allowed by comparing the requested state with the list of allowed next states.

Conclusion & Outlook

- The *CS* framework has been successfully redesigned
  - DIM is now used as communication layer
  - Object names must be unique
  - A Reservation mechanism has been implement in the low level object communcation
  - A Domain Management System allows distributed process management
  - *CS* Version 3.0 based on LabVIEW 7.1 has been released in December 2006
  - *CS* Version 3.1 based on LabVIEW 8.2 has been released in August 2007
  - *CS* will be extended for more device base classes to improve the standardazation of interfaces

- The PHELIX Control System has been successfully ported to *CS* Version 3.0
  - It is running stable; ~250 objects distributed on 13 PC's publishing ~10000 process variables.
  - A first Laser-Shot with design energy of 500J was performed in March 2007

- User requests for further *CS* developments
  - *CS* needs to be extendend for generic event builder, data archiving and inspection
  - Development of a GUI layout manager
  - Integrated installer
  - Demo example for quick startup and education purpose

References & Links

PHELIX:
- PHELIX Progress Report, GSI Report 2006 PLPY-PHELIX-15, 291, ISSN: 0174-0814
- http://www.gsi.de/forschung/phelix/index_e.html

CS Framework:
- D.Beck et al., Nucl. Instr. Meth. A 527 (2004) 567-579
- http://wiki.gsi.de/cgi-bin-view/CSframework/WebHome

DIM:
- http://www.cern.ch/dim