# 12. Control System

## 12.1 Executive Summary

SESAME control systems will utilize and use Experimental Physics and Industrial Control System (EPICS) toolkit for both Machine and beamlines; it is a distributed hardware and software control system (standard model) that requires no central device or software at any layer. For the development of graphical user applications, the Java Abeans package will be used, which is an application framework optimized for accelerator control systems.

When developing the control system (CS) of SESAME we try to consistently follow our vision: To produce a user friendly CS, where "user" stands for a physicist that operates the accelerator and not a computer expert. Therefore, the main characteristics must be that the CS is easy to maintain and that it allows non-experts to easily build powerful applications. To make such user-friendly behavior possible, a clean and consistent design is of paramount importance.

On the other hand, the budget for the whole machine must be kept tight and an optimum price/performance solution for the CS has to be found. For this reason one should incorporate as many standard and reliable commercial or open-source products as possible and not to unnecessary reinvent anything. We propose to use the same approach as for the SLS and ANKA control systems, which proved to be very successful both in technical performance, time-to market, simple installation and upgrade and management issues. It is possible, by adhering to the same design guidelines, to re-use the I/O boards of SLS, the core software EPICS and the applications of ANKA, which results in significant savings.

With the current state of technology, one should use PCs on the client side running either Windows or Linux operating systems. The software running on these PCs is best written in Java to allow platform independence. The input/output would be handled via VME boards, avoiding the need for an extra Fieldbus system. However, these statements will probably change in time and therefore should be revised at the implementation phase of the project. Be it any system and programming language, it is very important that the installation of the whole CS is very similar to an installation of a commercial product – using an automatic installation procedure for the software and plug-and-play for the hardware.

## 12.2 Introduction

In comparison with other light sources, SESAME can be considered a small to medium sized installation. Its control system can therefore reflect a reasonably simple and efficient design, as has been applied to similar light sources such as ANKA, BESSY, SLS, etc. Even though all those light sources use somewhat different control system technologies, they all share the commonly accepted three-layer architecture:

- Device Input/Output Layer, which is the only one that connects to the actual hardware
- Process Control Layer, where processes distribute controlled device data to/from many clients simultaneously
- Visualization Layer, with graphical user interfaces and similar applications

It is important to realize that the technology itself matters very little. The proof of this statement is the simple fact that all the above-mentioned light sources work very reliably and that their control system adds very little to the machine downtime. What is much more important is that detailed design and the implementation is made in a consistent way. This can only be achieved if all participants in the project, not just the control experts, adhere to well defined standard interfaces, both in terms of electrical connections and, probably even more importantly, in terms of software device interfaces.

The SESAME CS side must offer seamless crash recovery, flexibility in small variations of the design (e.g. new device types are added within minutes consistently across all layers) and above all it must enforce a consistent description of controlled items via standardized pattern interfaces. Such features have been implemented often by retrofitting existing set-ups and by highly non-portable solutions and spending a lot of resources, while a clean top-down design and the use of a powerful relational database could direct the implementation in the right direction with little added cost, just at the sake of increased discipline.

### 12.3 General Requirements

The SESAME control system should be reliable enough to provide users with the required beam for the scheduled operations with the minimum number of failures and a short time for repair. High performance is required to ensure that the response to the operator action is within a certain time (in our case 100 ms). Security should be taken into account very carefully in such a way that lets every control parameter to be monitored or changed only within the SESAME accelerator complex where at the same time has a controlled access from outside. Some information about the machine status could be placed at the SESAME home page on the World Wide Web. The control system should be easy to use for the machine operator; all the graphical user interfaces (GUI) should follow the same format and colors to monitor the signals and buttons. The GUIs should somehow let the operator to start/stop the machine very quickly or change the parameters easily. Alarms should be monitored and acknowledged in a clear and well-defined manner.

Hardware should be replaced without interruption to the machine operation whenever possible. Regarding software development packages and tools it should be well defined at the beginning in order not to end up with using and supporting so many different packages.

Scope of the control system is from the supported electrical interfaces to the GUIs and all the hardware and software in between. These electrical Interfaces (signals) are supported:

- Analogue I/O (-10 to +10 Volts)
- Digital I/O (TTL, 0/24 Volts)
- Motors (DC and Stepper)
- Serial
- GPIB

### 12.4 Architecture

Expectations of the final user determine the design of the CS. On one end of the CS he requires the physical devices to be added easily and on the other he wants a powerful and easy-to-use user-interface. In addition it has to be taken into account that specifications are often modified during the course of development, usually by the addition of features. All this requirements demand a CS that is able to provide a great deal of flexibility.

Unlimited flexibility, however, results in unacceptably high cost. In order to achieve reasonable flexibility at low cost, the SESAME CS design needs a small number of fundamental building blocks across the whole control system, both in hardware and software, which are not allowed to be altered in any way. The design of the blocks is of course frozen only after careful investigation of the available requirements and has to foresee possible future needs.

In terms of hardware on the client level, our goal is a solid framework of components that can easily be assembled into powerful applications. This structure of blocks or, components, respectively, allows the programmers to develop the layers of the CS independently of each other. The only constraints to the programmer are the interfaces that must be carefully designed in the first step of the design process in order to minimize interdependencies between layers, both in code and data. These interfaces must define all the possible interactions between layers in a consistent way. The result is cleaner code and better equipment, since every participant only has a limited number

of things to worry about. If some components later have to be optimized, they can just be rebuilt from scratch without affecting the rest of the CS, because the interfaces stay the same. In addition, testing, debugging and error correction becomes much easier.

In the next few subsections we present the design from the technical aspect.

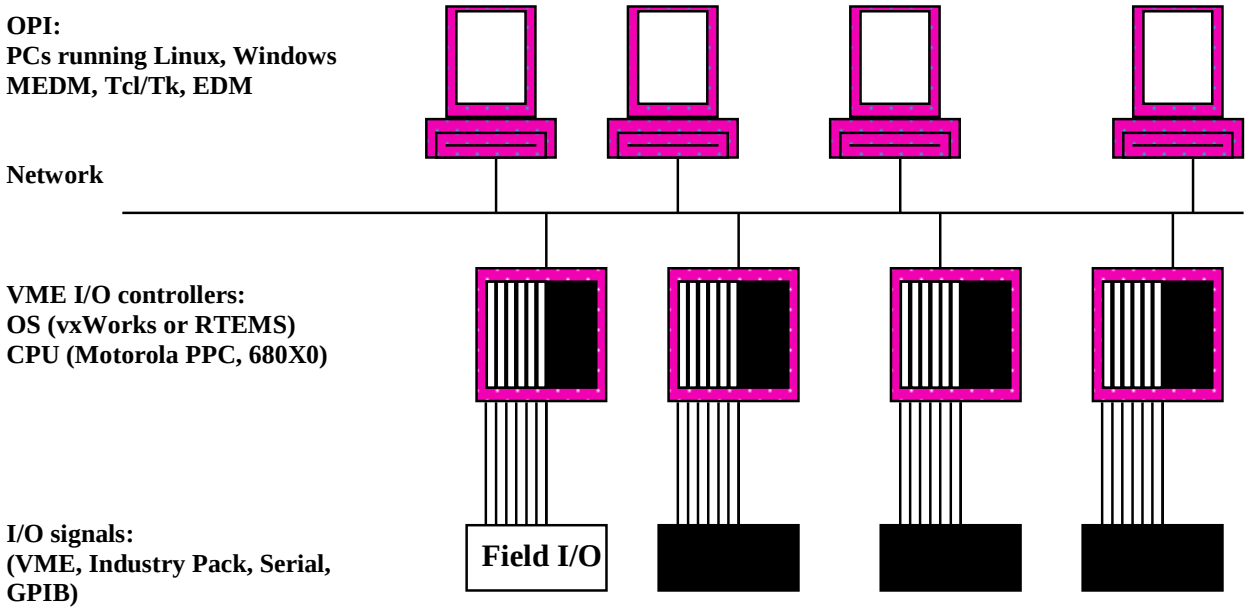


**Figure 12.1: EPICS architecture**

### 12.4.1 Hardware Layer

The Front-End (FE) layer—referred to as the Input/Output Controller (IOC)—is built from VME crates, CPU boards (Motorola 680X0 and PowerPC families), and I/O boards and runs a real time OS. I/O boards support many standard field buses and interfaces like IEEE-488 (GPIB), Bitbus, CANbus, RS-232/485, and Ethernet.

The Back-End (BE) layer consists of PCs running Linux and Windows. These layers (FE and BE) are connected via a network layer, which is a media (Ethernet) and a set of repeaters and bridges supporting TCP/IP.

### 12.4.2 Software Layer

The software layer adopts ‘client-server’ model. The client layer usually runs on the Workstation/PC physical layer and represents the top software layer. The communication between server and client is done via channel access (CA), which is EPICS backbone that handles all communication via the network; it runs on both the client and the server. The CA server runs on the front end CPU and is a unique task on each network node. The IOC runs a real time OS, vxWorks is widely used but RTEMS is also highly considered.

Typical clients are:

- Operator interface panels, Display managers (MEDM, EDM), StripChart
- Archiver
- Alarm Handler
- Applications
- System Configuration Database

### 12.4.3 Applications with Graphical User Interface

The applications of the visualization layer provide the interaction with the operator of the accelerator. They accept commands and display controlled values, either one by one or in groups, through tables and charts. Another type of applications are applications that manage whole sets of devices at the same time, like generic device tables, ramping alarm table, snapshot, etc.

To represent the values of the accelerator variables (currents on magnets, status of the vacuum pumps, etc.), several graphical user interface (GUI) components will be used (see also Figure (12.1)):

- gauge to display the value of a property
- slider to manipulate the value of a property
- status LED display to display the status bits as on the physical panel of the device
- chart to display the same type of values of different devices along the ring, e.g. vacuum profile
- trend – a chart that displays one or several values as a function of time

Another component is the “selector”, which enables the user to search for all available devices of a given type dynamically at run time and chose one or a group of them. When the choice is made, the system automatically takes care of the initialization process and the gauge is immediately showing the correct value. An example of such an application is shown in Figure (12.2).

In order to make the automatic initialization happen, a common framework for the development of all applications is needed. It is often neglected that this framework must also provide solutions for handling device errors and communication problems that occur. The framework should provide an Application Programmer’s Interface (API) that completely hides the network. Ideally it should also be platform independent in order to be flexible for future hardware upgrades. Such platform independency can be achieved by using interpreted languages whose interpreter is implemented on several platforms, e.g. Java, tcl/tk, Python. However, one should not create a maintenance problem by allowing several of those languages and GUI-building environments to be used. The libraries of the framework map the remote devices and their properties (encapsulate all remote calls from the client to a device server of the process control layer), following the same naming rules as defined in the process control layer. Beside this, they provide the following functionality:

- open the connection and perform the function calls on remote objects
- report and manage all errors / exceptions / timeouts arising from network communication
- provide handles for asynchronous messages, queue / demultiplex responses

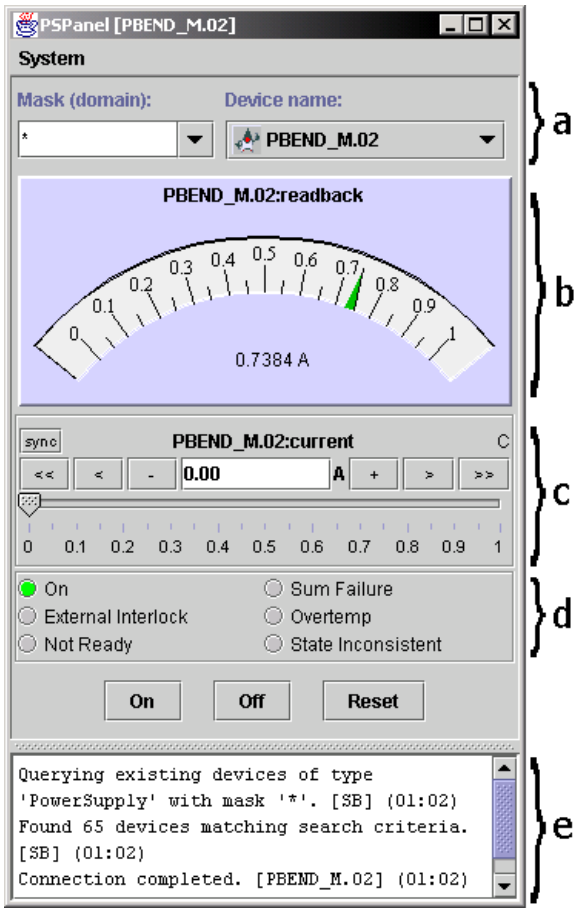


**Figure 12.2: An example of a power supply control panel, taken from the ANKA control system. This panel was created from Java accelerator framework beans (called Abeans) and GUI components in a Java visual builder environment without typing a single line of code. The following "ANKA CS standard" GUI components can be seen: a) selector, b) gauge, c) slider, d) status LED display, e) ATextPane - a pane with time-stamped messages for the user.**

We will use the Abeans libraries, which is the only application development framework that combines all the above-mentioned features.

### 12.4.4 The Databases

The control system involves three main databases:

- Configuration database that stores configuration parameters like channelo and device names, constants, calibration coefficients, attributes, alarm levels, I/O addresses, etc.
- Snapshot database that stores the state (i.e. all settings) of the machine
- Historic database that logs data over long periods of time

The main idea of the three-tier architecture is that clients don't access the database directly but through the EPICS and database server.

The static data used by the I/O boards are stored in the configuration database, too. They are downloaded to the EPICS server machines with a generic script procedure. Data that are used both by the client and EPICS server such as resolution, name, etc. are located only in one place of the database to ensure consistency.

To ensure a consistent configuration database, a version control of the database should be used. Ideally it should be the same versioning system as used by the software. The central version of the

configuration database is stored on one computer and then the data are copied to all the others in the control system using a series of scripts.

The snapshot database is used and managed through a dedicated application. The data are stored using a relational database or just plain text files.

Long-term history data are stored into a relational database and retrieved off-line through dedicated applications. A compromise between the sampling rate, storage time and available disk space must be made when the system is set up.

### 12.4.5 Machine Physics Applications

Machine physicists need a set of applications that deal with the physics of the accelerator. Some applications must exist already at the start of commissioning like machine function display (Figure 12.3) and orbit correction; others are developed if the need occurs: tune optimization, instability investigations, etc. As it cannot be predicted in advance, which applications are needed, most of them will be written by machine physicists and not the control group. Ideally, the machine physicists are given a software package for presenting the real-time state of accelerator transfer line in quantities relevant to machine physics. A unique design is the databush of Elettra, which has been revamped for ANKA and renamed DataBush. The DataBush connects to the accelerator devices, such as power supplies, beam position monitors, info server and then translates these objects to magnets and other to machine optics relevant devices and present them as software components. In this manner it acts as a bridge between machine physics and machine devices.

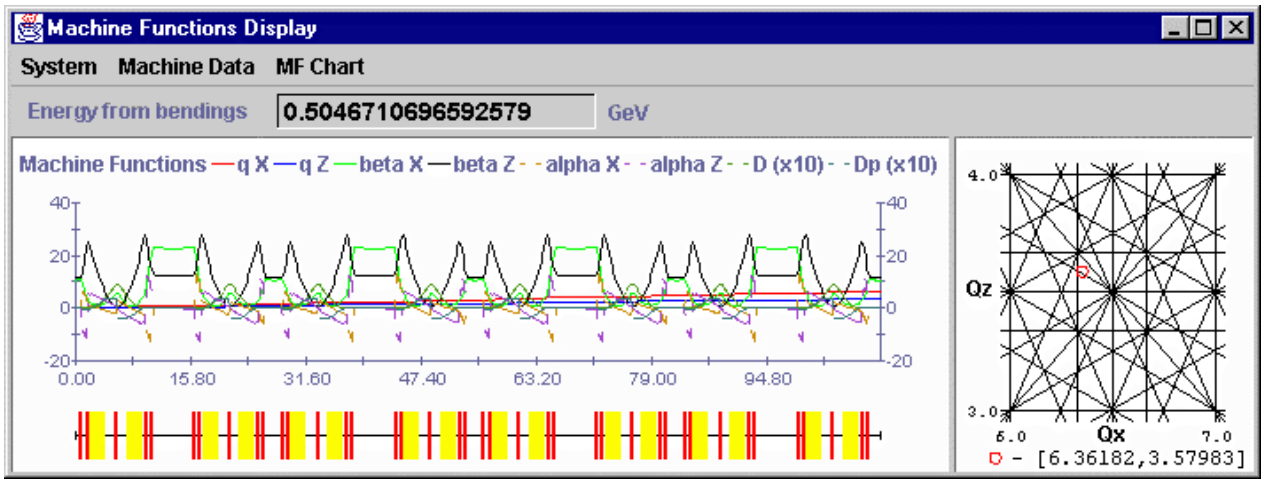


**Figure 12.3: The machine function display made with DataBush. It displays properties calculated in "real-time" by the DataBush, which read all relevant data from the control system.**

The DataBush is also able to simulate any device and work with it, as it would be a real machine device. This feature makes it easier to debug with DataBush and to use it similar to other accelerator optics calculating programs (MAD, etc.) but with the flexibility of a programming package.

Since the DataBush is Java package, it is platform independent. Plug-ins and models make it possible to fine tune DataBush to needs of particular implementation.

The DataBush is strongly integrated with CS. The accelerator specific implementation is hidden from the physics application into lower levels of the library. An application is communicating in this way with the accelerator through a general accelerator interface, not knowing/caring about how this communication is realized. Two major benefits come out of this architecture

**Portability**: The architecture of the CS allows Java clients to be portable by wrapping the machine specifics in a well-defined interface. Machine physics application can be used on other accelerators without changing single line of code, with appropriate modification of CS plugs of course. This enables sharing accelerator software among different accelerator, i.e. ANKA and SESAME in our case.

**More efficient programming**: A physicist can concentrate on machine physics problems. No special knowledge of the CS is needed, since all communication with CS is handled by DataBush
The following features make DataBush a flexible machine physics tool

- quasi real-time current or magnetic properties transformation
- quasi real-time linear optics calculation of machine function
- automation of mostly used machine physics routines
- representation of the machine with Java beans
- type and cast safe programming with DataBush beans
- customization of DataBush operations with pluggable mechanisms

### 12.5 Subsystems

Here we introduce a short description for the Timing and Diagnostics sub-systems, as they are critical and affect the overall performance of the machine and a brief to beamline controls.

#### 12.5.1 Timing System

The task of the SESAME timing system is to synchronize the operation of the whole accelerator complex and its subsystems together with Beamlines and experiments. The triggering of the Microtron, the beam injection to the booster, the ramping of the energy, injection to the storage ring and ramping of the energy in the storage ring happen according to the synchronization pulses from the timing system. The timing system also provides the capability to inject to specific buckets in the storage ring and control the filling pattern.

##### 12.5.1.1 Architecture

The SESAME timing system is based on the global distribution of timing signals using the so-called event system. This system is based on the design from the APS (Advanced Photon Source), which is redesigned for the SLS (Swiss Light Source).

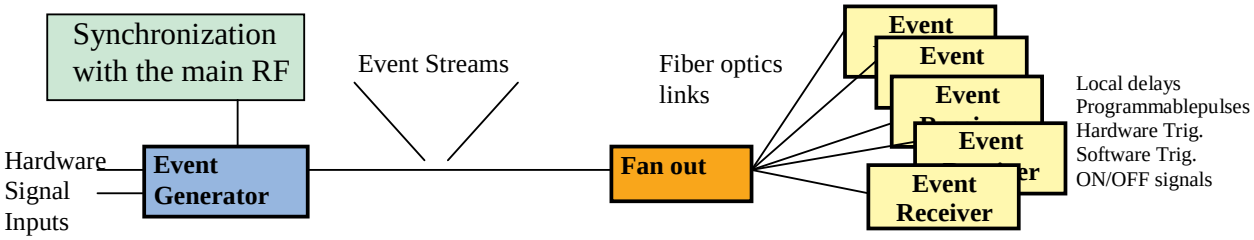


**Figure 12.4: Global Event Distribution**

Events are requests for various actions like triggering of the injector or extraction from the booster or even start of a software action. Codes are assigned to the events and are stored in the RAM of the event generator then it is scanned through at a specified programmable clock rate. The events are time multiplexed in a frame and distributed to all the event receivers at a rate of 50 MHz (20 ns resolution, 15 ps RMS jitter). The timing system has been integrated fully into the EPICS control system software so that timing parameters are available as EPICS channels. All the event receivers have a timestamp counter (40 ns precision), which is latched when an event occurs.

### 12.5.2 Diagnostics

Fine integration of the Diagnostic devices into the control system has great importance for operation and commissioning of the machine. Almost all these devices need to be synchronized with the beam operation, which is done by the proposed timing system.

Beam position monitors and current transformers (fast and slow) are the most important diagnostic tools and are monitored and controlled by number of ADC modules with 16 bit precision and also digital I/O modules.

Several screen and synchrotron light monitors are/is used to monitor the profile and presence of the beam in transfer line and ring, live on a dedicated monitor. All these cameras are synchronized with the beam injection and machine operation and connected to a video multiplexer, which will be placed in the control room and provides selection of each of the screen monitors at a time. Furthermore we will use a very cost effective PCI image grabber installed on a PC in control room, which provides us with digitized images and is available over the network for further analysis by physicists. One scraper is also used in the transfer line from booster to storage ring.

### 12.5.3 Beamline Controls

Beamline control will require many of the same facilities and features as the machine control system. It should therefore use the same system hardware and software, but with access to many of the features only available locally. Beam lines will require consoles that are tightly integrated with the scientific computing facility, as well as being part of the control system. The consoles also require good access to external institutes. As beam line configurations are likely to change, it is in this area even more necessary to allow easy configuration and reconfiguration of devices. One of the most common hardware devices in a beam line are stepper motors and perhaps also DC motors. These are used for the primary and secondary slits as well as monochromators, and the movement of the experiment. It should be possible to synchronize the movement of these motors, and operate them in a continuous as well as incremental manner. This is especially important during scans of energy at the beamlines, which is achieved by fine synchronization of the insertion devices with beamline components using timing events.

## 12.6 Control Room

A central control room will operate the entire machine, operator stations with the same facilities provided, in addition to some special purpose instruments like oscilloscope and spectrum analyzers.

## 12.7 Network

Network will support the control system, data acquisition and offices. It has to ensure a reliable and fast data transfer without bottlenecks or jams, based on fast Ethernet technology (100 Mbit/s). This will cover a wide range of supported VME interfaces, good performance and low cost.

A file server is needed where a central point of data collection and managing is provided, Development and Boot consoles in addition to port servers or serial cards (connect VMEs to hosts) are also required.

## References


[1] Bob Dalesio, "*Experimental Physics and Industrial Control System (EPICS) Overview*", 1999

[2] Stephen A. Lewis, "*Overview of the Experimental Physics and Industrial Control System: EPICS*", April 2000

[3] Steven Hunt, Swiss Light Source, "*Requirements of the Control system*", 1999

[4] Canadian Light Source, "*Control System Technical Specification*", Aug 2001

[5] ANKA CS Web pages:
http://kgb.ijs.si/KGB/Docs/Part%20I.%20User's%20Manual/1.%20Basic%20Concepts%20and%20Features%20of%20the%20Control%20System/Basic_Concepts_and_Features_of_the_Control_System_for_Anka.htm
[6] M. Plesko et al., "*The Object Oriented Approach to Machine Physics Calculations with Java Technology*", Proceedings of the ICALEPCS 2001 Conference, San Jose, 2001.