TESLA Test Facility Control System and its Current Status

O.Hensler, K.Rehlich, P.Shevtsov
DESY, Hamburg, Germany

Introduction

The TESLA Test Facility (TTF) linear accelerator for electrons is under construction at DESY [1]. It is based on the use of accelerating modules consisting of eight nine-cell superconducting cavities. About one year this linac (more than 100 meters in length) operated with one cryomodule and produced 125 MeV electrons. This period was intensively used by the control system developers to test, create and improve control software.

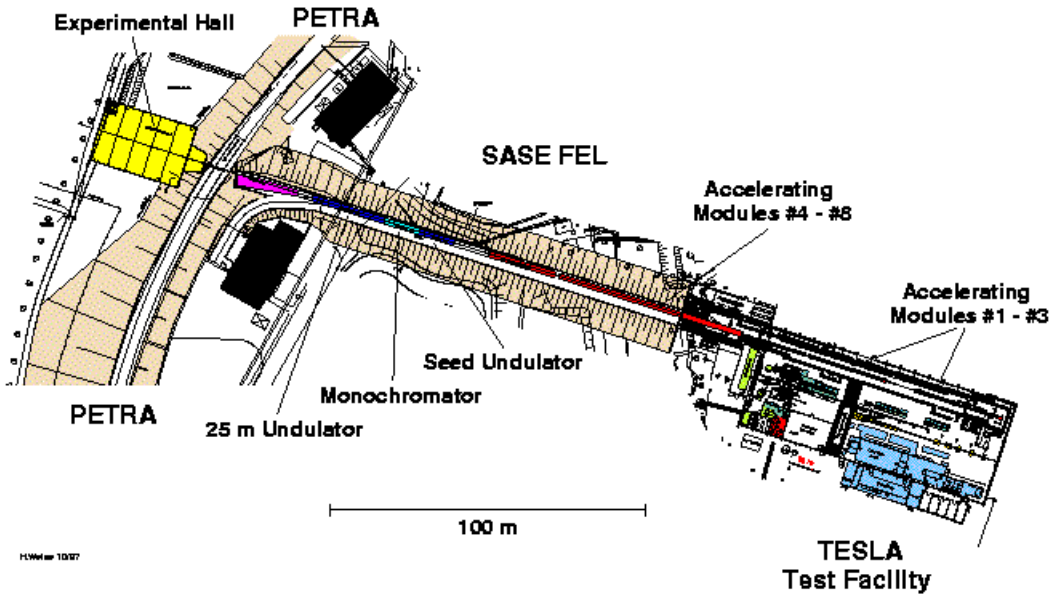

Figure 1: TTF infrastructure.

In August, 1998 the infrastructure of the TTF began to be extended by a new building. It will house the cryogenic unit cooling the 4.5 Kelvin cold helium from the HERA cooling system to 2 Kelvin, the working temperature of the superconducting cavities. In addition, the new building offers space for the cold water supply, high frequency transmitters, more control electronics and a test stand for complete TESLA accelerating modules (Fig.1). The present test facility is being extended by about 160 meters to incorporate a free electron laser (FEL) and a synchrotron radiation experiment hall. The final linac will accelerate electrons to the energy of more than 1 GeV.

1. The TTF control system overview

1.1. TTF subsystems and their integration

The TTF is an international project. Various hardware and control software components are supplied by collaborating Institutes from around the world. As a result one needs the integration of different software protocols into one, common control system. The integration is provided by

the Distributed Object Oriented Control System (DOOCS) [2] and is based on two methods: a multiprotocol Application Programming Interface (API) and different types of gateway servers. Both methods allow one to create a homogeneous control system.

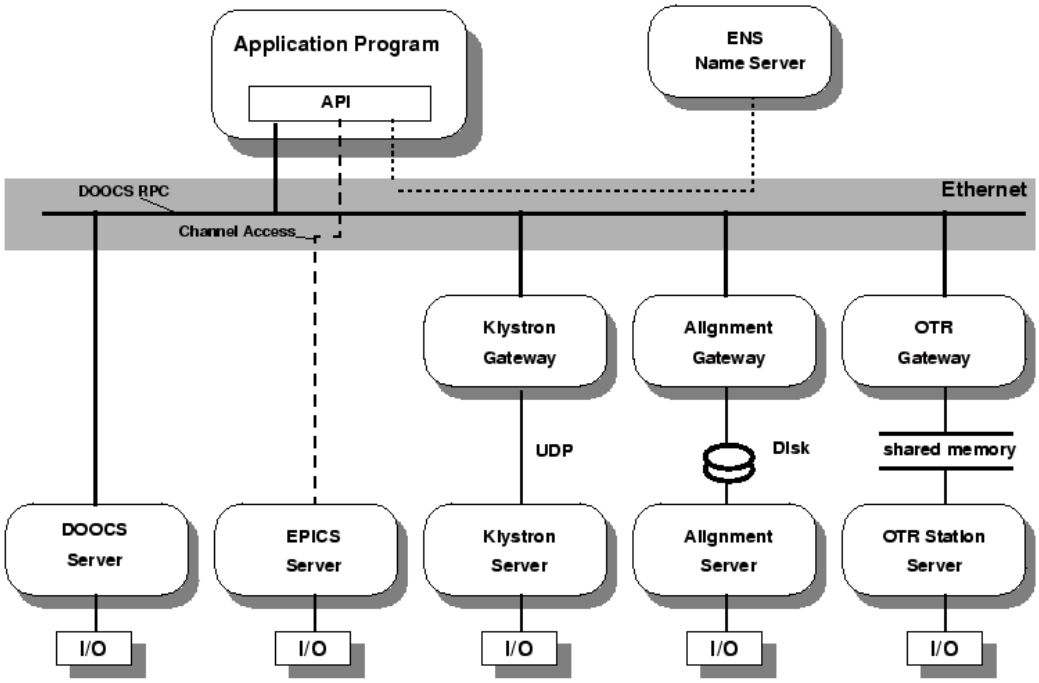

Figure 2: Integration of TTF control subsystems.

Some examples of the integrated subsystems are presented below (see also Fig.2).

– Many subsystems of the linac are completely controlled by DOOCS.

– The injector and cryogenics are controlled by EPICS [3] that is integrated through our multiprotocol API.

– The klystrons and some types of power supplies run a pSOS system and the so-called CLASSIC protocol [4]. A gateway server translates this service into the DOOCS environment and provides the data archiving.

– OTR screens are supplied with Macintosh (MAC) computers running LabVIEW. The communication is done through a shared memory in VME with two MACs and one SUN accessing the same data. Commands are transferred via mailboxes.

– The wire alignment system consists of OS-9 front-ends and a Linux data server. A DOOCS server provides the communication and data archiving with the use of the Network File System (NFS).

To make such an integration possible, a lot of new software components have been created during last few months: DOOCS devices servers, information services, new graphical user interface (GUI) and API modules.

1.2. DOOCS architecture

DOOCS is written in the programming language C++. It is an object-oriented design from the device server level up to the operator console. Class libraries were developed as building blocks for device servers, communication objects and display components. The communication is realized by a standard set of data and address objects which are transferred by Remote Procedure

Calls (ONC RPC) with the eXternal Data Representation (XDR) network data format. To incorporate C and LabVIEW applications, a C interface was designed on top of the C++ interface. A FORTRAN interface and a library for MATLAB/SIMULINK are also provided.

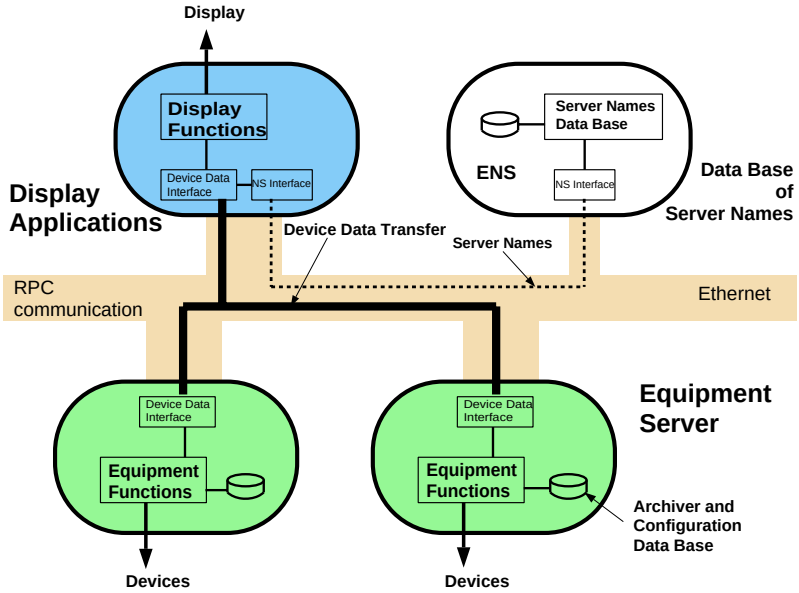

Figure 3: DOOCS communication interface.

DOOCS objects are concrete hardware devices. All properties of a particular device are handled by one device server. Every server can create as many as necessary device instances (locations) of its type. A DOOCS client is an independent program that requests the network on the data and then receives these data and works with them. A DOOCS device server is an independent program that completely controls a number of hardware devices, provides the network with the device data and receives messages from clients.

The DOOCS architecture consists of device servers with hardware connections on the lower level, middle layer servers to implement gateways, special calculations or sequencers, and client programs on the top level (Fig.4).

DOOCS is a distributed system because the device definition is done on the server side only and is transparent to the clients. Whenever a server is started, new device instances are created or new properties are added, all these changes are immediately available on the network. Client programs may ask an actual list of all devices and properties with a query request. Many different servers can run on the same or different CPUs. Even the same type of a device server with different locations is able to run on different computers.

The client part contains the communication, address and data classes. The data class has a lot of methods to access, change and convert data. With the address class the client program specifies the DOOCS names of control elements. The communication interface consists of only a few calls: to send data, to query the actual list of devices and properties as well as a synchronous (blocking) and a non-blocking (monitor) read calls. DOOCS clients run on SPARC stations with SunOS or Solaris2 operating systems as well as on Linux PCs.

Every instance of a device server defines a set of properties which are the access points for the communication and are implemented as data objects. The server library defines the basic class for device servers that provides the common communication objects like the name, error and status information. The device server inherits this information and adds the device specific code

and data objects to the standard objects. Data objects are defined for correction polynomials, float values, filters, archiver and spectra readings etc. By declaring a data object in a device server it is automatically inserted into the list of named device properties and becomes available on the network.

The device servers and middle layer servers are located on SPARC stations, VME-SPARC processors running Solaris 2 and PCs with the Linux operating system. Most server processes run on embedded SPARC CPUs in VME crates and talk to various VME cards via memory mapping or UNIX device drivers. As fieldbusses DOOCS supports the DESY SEDAC system and the industrial standards Profibus and CAN.

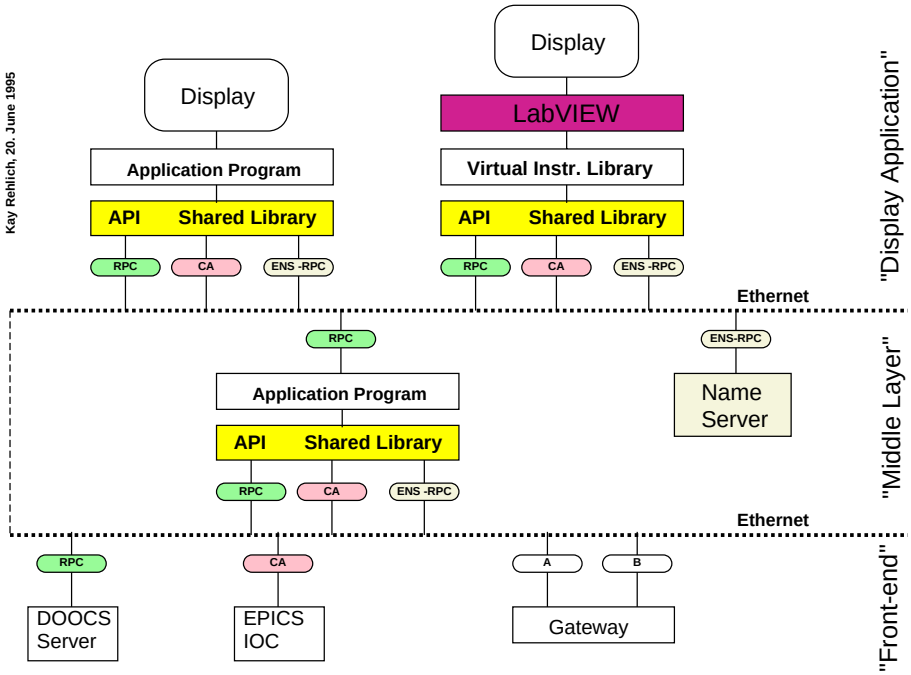

Figure 4: DOOCS software layers.

1.3. DOOCS device servers

A lot of efforts has been performed in DOOCS to make device servers independent from network problems. Every server station is equipped with a harddisk. This allows one to boot and run servers as well as to archive device data locally. Every device server keeps and updates a local database of the actual configuration and state in order to restore the previous system state after a power failure.

In addition, DOOCS has a service that looks after the current state of device servers and makes it possible to restart those that do not work at all or do not work properly at the moment ("hung", for example). This service was created in the frames of the DOOCS server library. It runs on control computers, periodically polls all local registered DOOCS device servers (with so-called "null" RPC calls) and collect various statistical information about their state (the CPU time and memory usage, time of the last start, the resident size, network statistics etc.). This information helps a lot to analyze hardware and software problems in the control system and to keep all server processes running.

1.4. Access to control devices

The TTF control system is highly distributed. The access to the parameters of the numerous control devices is supported the next way. The unique DOOCS name in the form "facility/device/location/property" is assigned to every element (parameter) of the control system. We already mentioned the last three parts of such a name. The "facility" acts as an additional degree of freedom and can be used, for example, to test the same server on different computers. The complete set of valid DOOCS names together with the information about the control elements corresponding to these names are stored by a special DOOCS RPC server - the Equipment Name Server (ENS). A user is provided with the Application Programming Interface library which has a set of standard calls to access the control system. The software is designed so that the ENS communicates exclusively with the API. Hidden from a user, this communication effectively resolves DOOCS names contained in client requests and directs these requests to the proper devices encoded by such names.

Every control system must provide protection against unauthorized access to its sensitive devices. Authorized control or modification of a particular parameter depends on many factors: who is making the request, what is the type of the request (e.g. get or set), from where is it (e.g. the control room, the office building, off site) etc. For the TTF control computer network the protection is organized with the use of the Equipment Name Server. The list of all users who have the permission to access control devices is prepared by the control personnel and is stored by the ENS in the form of the Authentication Table. With the use of this table the pair "API-ENS" decides if a client is authorized to perform the requested actions. If so, then such requests are executed. Otherwise, the client is informed that these actions are forbidden for him and then may only ask for a list of the authorized users to have a possibility to consult with them on control problems.

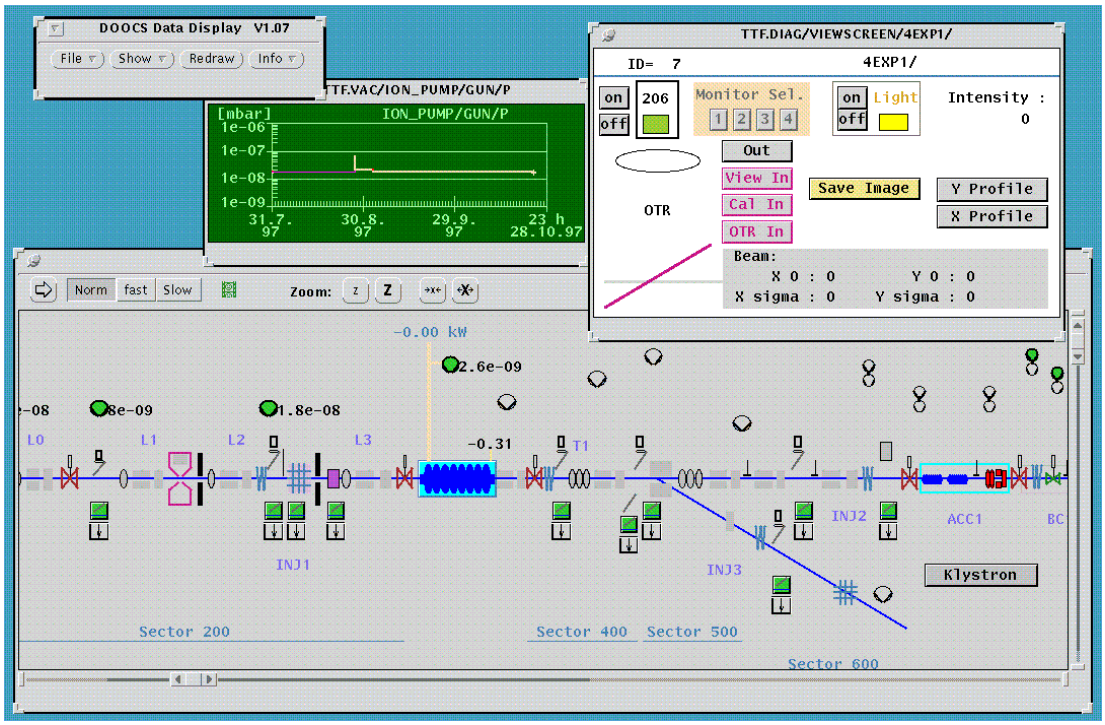

Figure 5: Zoomed display of the TTF Linac.The visible part can be shifted by sliders. A click on an object activates a window with detailed information (e.g. a viewscreen station) or a plot (e.g. history of the pressure of an ion pump).

2. Operator interface development tool (ddd program)

In order to provide easy graphical operator interfaces to the TTF control system, DOOCS offers an operator interface development tool: DOOCS Data Display (ddd). The ddd is a program to draw components for the control system with an editor and to run the created drawing with real data [5]. It is written in C++ with the use of the DOOCS API and X Window graphics libraries. The TTF control system has a lot of similar devices with the same types of properties. Only the device addresses are different. The ddd enables one to create reusable components. A component may be used in several separate interface windows.

All synoptical pictures are created with the assistance of the ddd editor. One can create simple graphical forms like text or lines and complex elements to display and control data (Fig.5). Parts of a drawing and any group of graphics forms may depend on the on-line data. The components or whole windows are stored in a library (in ASCII format) and can be used in other drawings.

The control display is animated by actual device data and may contain sensistive fields to activate commands to the devices or to start new subwindows. The required information is passed to the subwindow. As a result a single prototype of a subwindow has to be created once for all devices of the same type. History charts or digital scope plots are also integral parts of the ddd program.

3. First results and trends

Although the DOOCS was newly developed with very limited manpower, the TTF linac has been commissioned with a complete set of software tools in a very short time. It was necessary only three days to deliver the electron beam from the injector up to the end of the linac with full designed energy. First experience with the TTF control system has demonstrated its high reliability. It served a lot of user applications in a transparent way and could be rapidly extended to include new types of control devices. The decision to separate the actual information part of the control system into special DOOCS server tasks has significantly reduced the information load on API and made control applications very compact.

Current TTF control activities are connected with the completion of this project and are directed to the future electron-positron collider TESLA with its needs. This requires the further development of the DOOCS software libraries, the implementation of new protocols, client "standard" calls etc.

Conclusion

An important step to provide the TTF accelerator with a reliable and flexible control environment has been accomplished. The TTF common control system is open to any evolution and hardware modification. In the near future, it has to approve its high performance and to meet the operational conditions for the acceleration of high intensity electron beams and the FEL experiments.

References

[1] D.A.Edwards et al. "TESLA Test Facility Linac - Design Report", DESY-Print, March 1995.

[2] S.Goloborodko et al. "DOOCS: an Object Oriented Control System as an Integrating Part for the TTF Linac", Proceedings of ICALEPCS 97, Beijing.

[3] L.Dalesio et al. "The experimental physics and industrial control system architecture: past, present, and future", NIM, A 352 (1994), p. 179-184.

[4] R.Goodwin et al. Nuclear Instr. and Meth. A 293 (1990), 125.

[5] K.Rehlich "An Object-Oriented Data Display for the TESLA Test Facility", ICALEPCS 97, Beijing 1997.