STANDARDIZATION OF THE DELTA CONTROL SYSTEM

D. Schirmer, E. Kasel, B. Keil and D. Zimoch
Institute for Accelerator Physics and Synchrotron Radiation,
University of Dortmund, Germany

Abstract

DELTA, the 1.5 GeV electron storage ring facility of the University of Dortmund, dedicated as a facility for accelerator physics as well as for synchrotron radiation based experiments, has started routine operation in summer 1998 [1].

The facility, consisting of a 80 MeV linac, a full energy booster and the main storage ring, is controlled by a typical 3 level soft- and hardware architecture with a high level operator interface (OPI), a real time process level and a low I/O device control level. Up to now the core of this system is non standard hand-made, which was sufficient during the commissioning phase.

Since January 1999 the small control group is working on the migration to the **E**xperimental **P**hysics and **I**ndustrial **C**ontrol **S**ystem (EPICS) [2]. This standard toolkit provides full support for all hard and software levels with a high degree of compatibility to other facilities.

Apart from the basic functionality of EPICS, additional adaptations to DELTA specific components and specialized tools are under development. Furthermore, in order to guarantee intelligent and more automatic ways of controlling, DSP based fuzzy controller, neuronal networks as well as expert systems are planned.

1 INTRODUCTION

Up to now DELTA is running with a control software, that was developed in house. This software uses the standard topology of modern accelerator control systems. Furthermore, the introduction of the CAN field bus system and object oriented programming techniques on the real time level, now more and more used in accelerator controls, are included in this system. It fulfils the requirements of basic machine operation, is stable and now provides a Tcl/Tk based graphical user interface (GUI).

The main caveat of the system is, that maintenance and further developments have to be done in house, completely, by a control group of 1 to 3 people. These personnel resources are insufficient to keep track with standards required for a developing facility. Therefore, the decision was made to change the system to EPICS, to introduce a standard and to participate on the progress of a large community around the world. Some basic parts of EPICS are already installed and some configurations are already under test. The system will be introduced within several steps, covering the superconducting asymmetric wiggler magnet (SAW) and the linac first.

2 THE MIGRATION PROCESS

2.1 The 'old' Control System

Even after intensive bug fixing, optimization and further consolidation of the present control system, there are still some problems which will come up when the demands increase and additional tasks are requested. The following main problems have been made out:

- no asynchronous message mechanism (only polling)
- inefficient network access due to data polling
- no automatic error handling
- no event synchronisation across the network
- no integrated modeling and simulation tools
- no database for static data management
- insufficient documentation

To overcome some of these essential problems, a long term changeover to the standard EPICS toolkit is in progress. This must be a step by step migration because parallel storage ring operation is required. Furthermore, to provide a good base for EPICS, some improvements concerning the hardware are advisable (see Table 1).

Table 1: Upgrade of the DELTA Control System

	'old' system	**'new'system**
Network	Backbone bus 10base5/2	Distributed star 100baseTX/FX
OPI	HP-UX	Linux-PC
Core System	Delta specific	EPICS R3.13
GUI	Motif & Tcl/Tk	Tcl/Tk, EPICS Extensions, ...
Process Level	VME 68k-CPU VxWorks 5.1/5.3	VME PPC-604 & VME 68k-CPU VxWorks 5.3.1
Field Bus	CAN, GPIB, FGs	CAN, GPIB, FGs

The 10base5 backbone-bus network will be exchanged by a distributed star ‘fast ethernet’ network topology, which guarantees sufficient bandwidth for higher network traffic. The operator interface (OPI) server as well as EPICS clients and high level control applications are running on Linux Pentium-PCs, an inexpensive alternative to classic UNIX workstations. On the process level, the 68K CPUs will be replaced by modern PowerPC CPUs providing 100Mbit high performance ethernet, modular PMC-Slots and expandable memory up to 128 MB DRAM. The device I/O-controls supported by CAN or GPIB field buses and the DELTA specific arbitrary function generator VME-Cards (FGs) will not be changed in the near future.

2.2 High Level Software

The high level controls based mainly on ‘Open Source’ [3] software products as Linux, KDE, Tcl/Tk, and the GNU tools. To be open for future developments further supplements like Perl/TK, Python, Qt or Java should be possible. A homogeneous interface for these extensions is the installation of a uniform middleware like CORBA [5]. CORBA provides a way to build up a rather flexible but nevertheless standardized distributed client/server model in the framework of accelerator control systems. The integration of accelerator physics simulation tools like TRACY-2 or Goemon [4] as a modeling server could be an useful example. The CORBA implementation MICO [6] is under test concerning such opportunities.

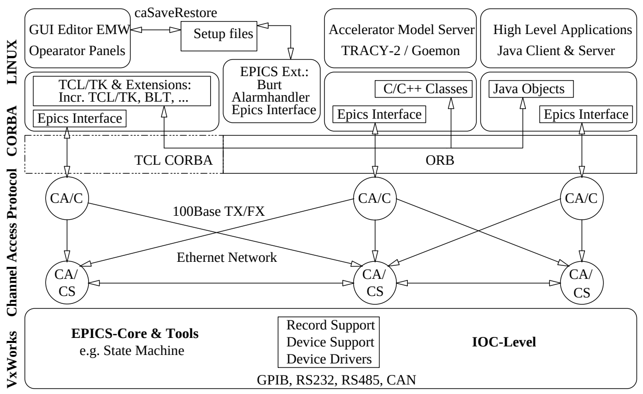


Figure 1: Layout of the Control Software Layers.

Fig. 1 shows the software layers from the high application level like Tcl/Tk operator panels, special modeling server or Java programs, communicating directly via the channel access protocol with the clients/servers running on the IOCs, where finally the device support handles the hardware access. The object request broker (ORB) makes horizontal interaction between object oriented programs possible and provides an optional gateway to the IOC level.

3 STREAM DEVICE SUPPORT

3.1 EPICS Device Concept

In EPICS, process variables are hold in so called ‘records’. These have a set of ‘fields’, depending on the record type. One of them, the VAL field, contains the value of the process variable. Other fields may contain values that are computed from VAL or compute to VAL. For output records, the device support reads VAL or one of the computed values and writes it to the hardware. For input records, the device support receives a value from the hardware and stores it into VAL or another field.

Device support code depends on the record type and on the hardware. While there is a very limited number of record types, which are almost the same to all EPICS users, there is plenty of different hardware. This requires a lot of separate device supports.

3.2 Field Bus Devices

Field busses allow various hardware to be connected to the controller. From the viewpoint of software development, one advantage is that the device support is only dealing with the bus, rather than the hardware device. Thus, no separate device supports are needed for every hardware type, but only for every bus type. If one can assume certain attributes to be common to all bus types, the separate parts can be made fairly small.

At DELTA, we have several devices that can be controlled with streams of bytes. Either transmitted over GPIB, RS232, RS485 or CAN. Since we use CAN to RS232 and CAN to RS485 converters, the number of separate bus depending parts is only two. We use the MicroSys IEC 03 GPIB card, based on a NEC µPD 7210 controller chip, and the esd CAN2 card, based on two Phillips PCA82C200 controller chips.

3.3 Stream Device Support

The Stream Device Support assumes, that a device can be controlled with a (bidirectional) stream of bytes, transmitted over a field bus with a number of logical channels. Data I/O is controlled by a simple configuration language, consisting only of the commands ‘in’, ‘out’ and ‘wait’ and of some parameter assignments. We call this a ‘protocol’. The protocol is the only part that depends on the accessed hardware device. Note that a new device only needs a new protocol file, but no C coding, compiling or any knowledge of field bus or EPICS internals. However, a new type of field bus (or even another bus controller card) certainly requires all this.

Arguments to ‘in’ and ‘out’ can be ASCII strings or a sequence of bytes (even zero bytes) or any mixture of these. Strings can contain format description, well known from the C functions ‘printf’ and ‘scanf’ and some additional formats like ‘%b’ (binary notation), ‘%r’ (raw bytes) or '%{enum0|enum1|enum2}' (enumerations). A format description refers to a record field, depending on the format and the record type. A formatted input or single input bytes can also be skipped.

The Stream Device ensures that the logical I/O channel is locked, thereby preventing mixing of I/O from different records to the same hardware. Other logical channels, i.e. other hardware on the same bus, are unaffected from this.

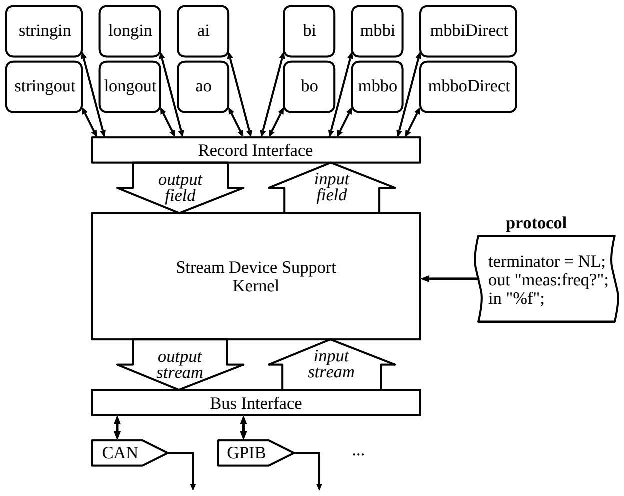


Figure 2: Interactions of the Stream Device Support [7].

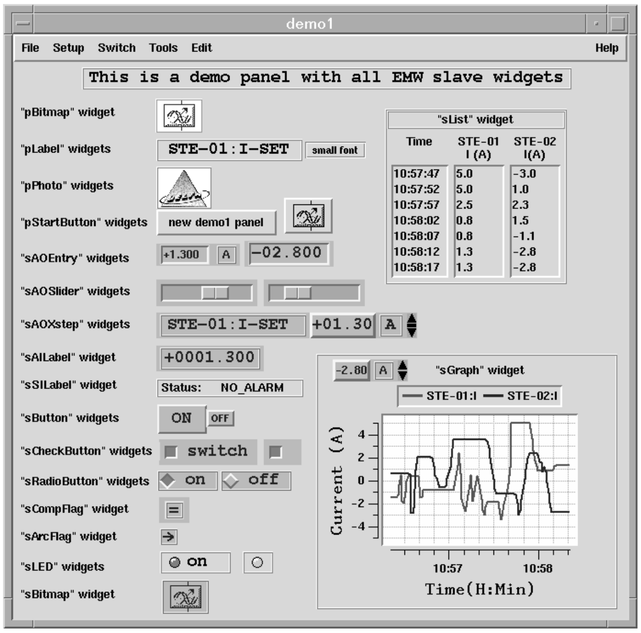
4 EMW OPERATOR PANEL EDITOR

EMW (Epics Mega Widgets) is an EPICS GUI builder. It can be used to develop operator panels for EPICS control systems without programming. EMW operator panels are created interactively by dragging and dropping so called EMW slave widgets on a background widget, called the master widget. Slave widget properties (I/O channel names, size, colors, ...) can be edited interactively and saved in a configuration file that completely describes a panel. Any EMW master widget can perform generic operations on sets of slave widgets placed on it, e.g. switching buttons, scaling analog output widgets, backup/restore etc (see Fig. 3).

EMW was developed using the object-oriented Tcl/Tk extension [Itcl]/[Itk]. It can be maintained easily due to the clean class structure of the software, comparable to C++, while still providing the rapid development speed of Tcl/Tk (e.g. compared to C++/Motif). EMW also uses the extensions BLT2.4 and Tcl-ET (EPICS interface).

Although using Tcl, the speed of EMW e.g. on Linux PCs is sufficient even for large panels. A panel for 30 power supplies (120 slave widgets (=800 Tk widgets), several hundred I/O accesses) requires less that 10 seconds startup time on a 350 MHz Linux PC. Scaling of 30 analog output widgets can be done with several Hz, including display update.

Since no GUI builder will satisfy all users, EMW can easily be extended without loosing compatibility to the base version. New master or slave widgets can be developed very fast, since they inherit methods and variables from the respective base classes provided by EMW. The class structure guarantees the same look and feel (e.g. alarm colors or I/O error handling) to all EMW operator panels of a control system, and it also defines a clean interface between the EMW base distribution and new user-specific extensions of EMW.



Time	STE-01 I (A)	STE-02 I(A)
10:57:47	5.0	-3.0
10:57:52	5.0	1.0
10:57:57	2.5	2.3
10:58:02	0.8	1.5
10:58:07	0.8	-1.1
10:58:12	1.3	-2.8
10:58:17	1.3	-2.8

Figure 3: EMW demo panel with all available widgets.

EMW is public domain (see [7]). It was developed by a German company for the SLS linear accelerator, but is also used at the **D**ortmunder **EL**ectron **T**est **A**ccelerator DELTA. At the moment, novel linac controls based on EMW, EPICS and the Stream Device Support are already under test.

5 ACKNOWLEDGMENTS

The authors would like to thank the EPICS community, particularly the control groups of BESSY and the SLS for permanent help, inspiring ideas and fruitful discussions.

REFERENCES

[1] D. Nölle et. al. EPAC Stockholm (1998) pp. 611-613
[2] EPICS: NIM A 352, pp. 179-184 (1994)
[3] Open Source: http://www.opensource.org
[4] H. Nishimura, NIM A352, pp. 379-382 (1994)
[5] CORBA: http://www.omg.org
[6] MICO:http://www.vsb.informatik.uni-frankfurt.de/mico
[7] http://delta.uni-dortmund.de/controls/pub/doc