PROTOTYPING THE BESSY II CONTROL SYSTEM

G. v. Egan-Krieger, R. Müller

Berliner Elektronenspeicherring-Gesellschaft
für Synchrotronstrahlung m.b.H. (BESSY)
Lentzeallee 100, 1000 Berlin 33, FRG

Abstract: *In the upgrading procedure for the old control system at BESSY only hard– and software is chosen that meet the standards nowadays known. Modern computer networks will be used: host computers running* UNIX *are linked by an ethernet, a* PROWAY-C *field bus network connects master microcomputers of interfacing parallel bus systems. The software that is developed now for the control of the given machine will supply a basic system to BESSY II. Any application program that has to be rewritten for the modernization of BESSY I will be transferable to BESSY II. Operating experiences at the running light source BESSY I will influence significantly new features that will be realized for BESSY II.*

The window oriented interface for the operator is easily reconfigured for BESSY II. A library of well tested system interface calls *gives a safe set of tools to the scientists for the programming of diagnostic and operating procedures. For the microcomputer software controlling the interfaces modules are designed, that allow a unified and hardware independant access to any equipment.*

Introduction

After the modernization of the BESSY control system has been completed successfully the basic components of the new system will be: X window servers, a UNIX workstation network based on the ethernet, a micro processor network based on a serial field bus conformal to the *manufacturing automation protocol* (MAP) standard and internet-field bus server stations [1].

In the progress of modernization the primary pilot system will be expanded smoothly to full scale. Only the hard– and software *interfaces* between the constituents of the different system levels have to remain fixed. They were chosen to follow known or emerging standards as closely as possible today. Thereby we are able to make use of the flexibility of an open system. Every decision about new components can profit from the experiences with the preceding subsystem. Thus, the system will be tuned to the demands of the machine people.

System Components

Presently we are in the 'introductory phase' [1]. Own operating experiences with the new system are based on the configuration of the pilot system.

Presentation Level

The operators interface will be based on 'powerful' high resolution color graphic displays with keyboard and mouse controlled by the X windowing system [2]. We will learn from the operators complaints which properties of the available hardware have to be emphasized. Speed in throughput from the application program to the presentation can be tuned on the X client side with processor power and RAM, on the X servers side by dedicated graphic boards. The complexity of many overlapping windows overloaded with information (fig. 1) could force the operator to spend most of the time on manipulating the windows and finding out how to perform a certain action. We prefer a dozen simple X display stations and distribute the different tasks in a prestructured way over these screens.

All console workstations must have NFS access to the database of the central host. Tasks that require real time performance or throughput to the field level eventually have to run on workstations equipped with an own field bus link. Today only the gateway and a spare workstation have access to the field bus. We feel that the ethernet will not be a bottleneck in the command transport line. In any case interactive tasks have to pass their instructions from the X server to the X client to the application on the ethernet.

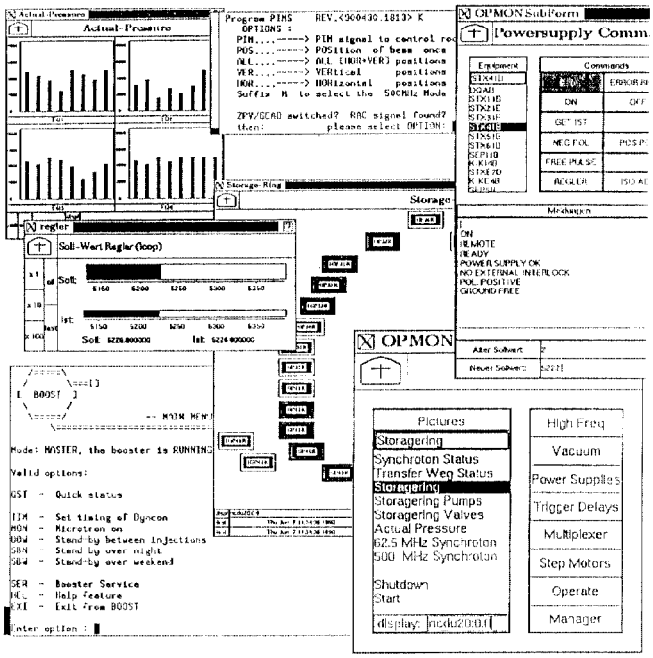

Fig. 1: *An operator's view of the machine*

Gateway

Presently we use a TRITON (SPECS GmbH, Berlin, FRG) as gateway. It is a RISC station based on Intergraphs CLIPPER processor with a typical performance of 5–7 MIPS and 1.5 MFLOPS at 33 MHz in a UNIX environment. This station is equipped with a VME bus coupler that allows the VME bus field bus interface to be tied to the high performance CLIPPER bus.

Network

For the communication between UNIX workstations today the internet protocol on an ethernet is obvious.

At the field level we decided to use eLAN [1], a realization of the PROWAY C standard (IEC 955). PROWAY C specifies the two bottom layers in the ISO/OSI model and is conformal with MAP. We can be sure that in the near future workstations commonly provide both ethernet and MAP boards. Then we need at most bridges to the real time PROWAY C segment instead of gateways with active protocol conversion.

Table 1: *Comparison of three real time field bus systems*

Field Bus Features	eLAN	MIL-STD-1553	PROFIBUS
bus organisation	multi master	master/ slave	multimaster master/ slave
max. number of nodes	128	31	32
Data Rate [kbit/s]	1000 (5000 max.)	1000	9.6 to 500
packet size (max.)	1024 bytes	33 * 20 bit	255 bytes
information length (max.)	1015 bytes	32 * 16 bit	246 bytes
data link service class	command/ response	command/ response	command/response polling
transmission medium	twisted pair fiber optic	twisted pair	twisted pair
line interface	transformer	transformer	RS-485
data encoding	Manchester II	Manchester II	UART ISO 1177/2022
error checking	16 bit CRC on packet	parity bit on 20 bits	2 or 4 bytes hamming distance
max. cable length [m]	2000	1000	1200
standard	Proway C IEC 955	MIL-STD-1553	DIN V 19245
VLSI chip	—	COM 1553B	—
functionality	—	LOW	—

The token passing bus eLAN is absolutly reliable even in a noisy environment like a booster synchrotron. At BESSY major subnets are ground insulated by fiber optics repeaters to provide additional stabilisation. In table 1 the features of the eLAN network are compared with other well known field buses.

Device Control

At this level compatibility with PROWAY C is less obvious. Already available are several parallel buses like ECB, VME or Multibus. The microcomputer boards working as bus master are based on Z 80, MC 680x0 or Intel processors (fig. 2). Predominantly we will use the ECB bus for I/O purposes. In most applications the stability of the inexpensive, 8 bit synchronous ECB bus is even superior to the 16 bit asynchronous VME bus. We will apply the even cheaper and less complex (MAP compatible) front end field bus systems (e.g. PROFIBUS) as soon as they become available.

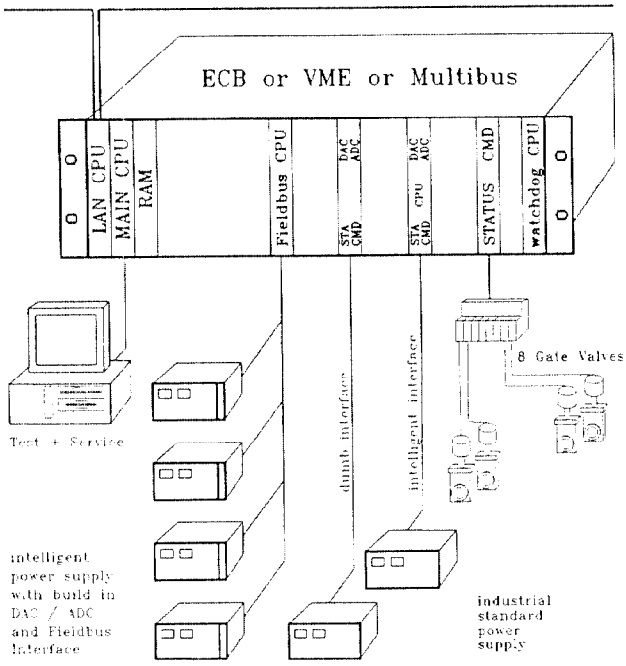

Fig. 2: *Sketch of a field bus node*

Software Structure

The basic structure of the software is introduced by rules that follow from the requirement to operate the old and new control systems in parallel [1]. Even if several UNIX workstations supply computing power it is in the very beginning a classical single master/multi slave model. Coordination of different and concurrent tasks running on different machines lies within the responsibility of the user.

Mapper

Every application program that has to communicate with a X window display has to address the universal X client of that display, called *mapper*. Mapper and application exchange requests by the event driven *application form interface* (fig. 3). Only the objects and the context are specified, *not* the graphical representation.

Appearance and functionality of the form entries are described in a database of forms. On a request to display a form on the screen, the mapper reads the description from the database and interprets it with a *yacc* type parser. The form gives instructions to the mapper, which action to perform e.g. on a user request to update the setting of a specified device. This example could result in the simultaneous adjustment of a slider and the update of an editable 'string' field.

X11 output is currently produced with all the comfort of the object oriented 'InterViews' toolkit [3]. The source files of the forms are in ASCII and are composed of the elementary graphical objects interspersed with TEX like formatting commands. Creation and modification of forms with a text editor is easy and flexible.

System Calls

This library handles globally hardware access for application programs written in C++, C and F77. The basic C++ classes are full scale read and write operations to the field bus system with appropriate data structures. Member functions define different opcodes specific for the desired device control commands. Parameter exchange with the user program is simplified to the name of the addressed equipment and very specific information. The functions will be thoroughly tested and well documented to give a reliable platform for the development of stable application programs.

In addition a system call interpreter reads these system calls line by line from standard input. In connection with the control structures of a UNIX shell this interactive library interface allows one to rapidly write and execute small test or driving scripts. No programming knowledge is required.

Field Bus Daemon

The system calls are passed to the field bus *daemon* as remote procedure calls (RPC). The packet service UDP/IP is used in anticipation of the field bus packages. As long as we have to address the old control system the *daemon* has to fork according to his equipment address list. The RPC is either routed to the RPC server (boards running OS9) [1] or the call is converted to the BESSY message protocol and forwarded to the message server (boards based on Z 80 microprocessors *or* the old minicomputer system).

Incoming packages are received by the field bus reader function. Depending on the format they are routed to the client directly or after conversion to RPC packages. Unexpected messages are passed to an error handler.

System Control Monitor

On system startup the system control monitor schedules all essential server and background processes. An authentication widget (*getty++*) allows for login procedures. As a service to the operators the basic applications are started immediately after login. Whenever a process terminates abnormally the system control monitor tries to clear the situation.

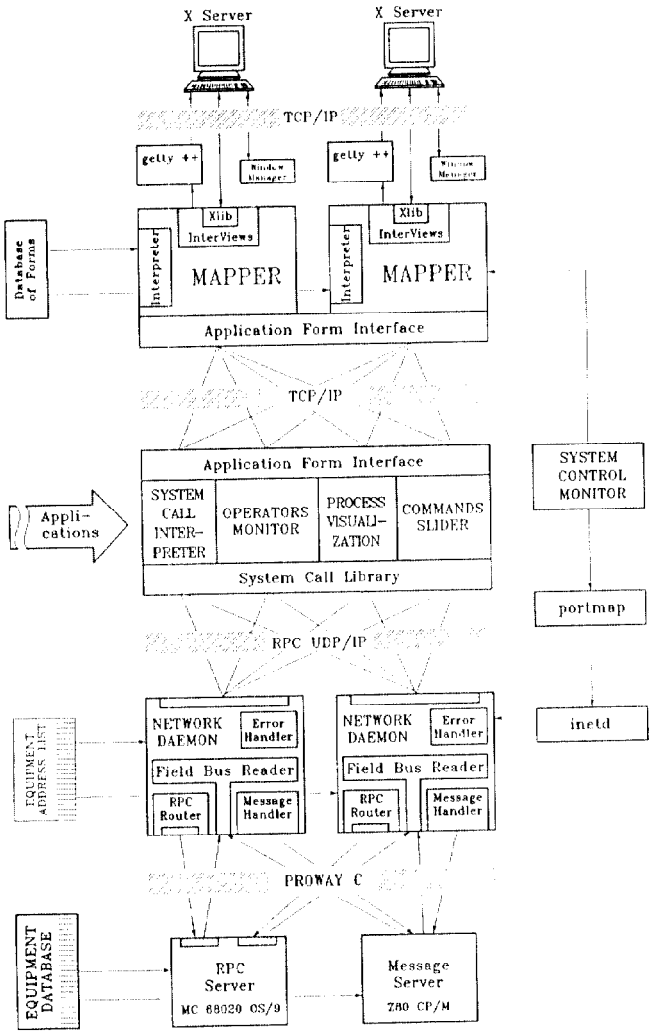

Fig. 3: *Software building blocks and communication paths*

Peripheral Interface Modules

All software for device control is written in a high level language (PASCAL, MODULA 2) and basically independant of the microcomputer board. Peripheral interface modules are structured in shells (fig. 4) and allow for easy adaption to hardware changes. A different parallel bus can be used as soon as the I/O driver capsule has been replaced. Different interface cards require modifications in the DATA SET OF EQUIPMENT. New basic functionality can be added to the OPCODE SERVICE. Autonomous tasks are located in the outermost layer (CONTROL/SUPERVISE).

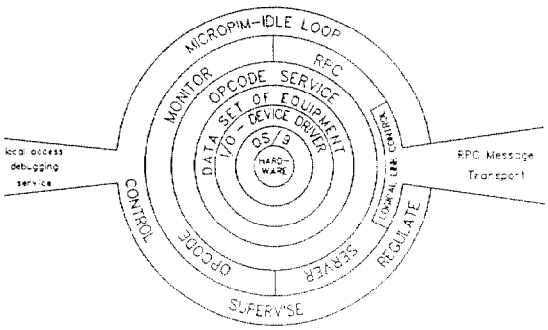

Fig. 4: *Microcomputer peripheral interface module*

Ways of Further Development

There will be no problem to implement modelling programs like MAD and supply the input data from the machine settings. Responses to planned actions could be calculated and displayed first, before a confirmation performs it in reality. Today this is done at many places.

With respect to the software interfaces (fig. 3) it should be no problem to install knowledge based supervisory packages which partly use tools typical for artificial intelligence applications [4].

On the field level many improvements are foreseen. A horizontal command transport layer will be introduced. Then complex instructions are partly executed at one node and the remaining tasks are passed to the other nodes. Status requests with an autoupdate flag will cause the microprocessor to constantly control the device and report status changes as long as the client lives. On the network repair mechanisms with respect to faulty stations can be sophisticated. The local database will contain pointers to expressive graphic presentations which are displayed on the console at specific device conditions. In case of failures the names of responsible personnel could be displayed.

Conclusion

Operating experiences with the control system now under development at BESSY will be a realistic test for specific hardware configurations and major software building blocks. In the end we will be in a position to scale the system to the needs of a more complex system like BESSY II.

Acknowledgement

It is a pleasure to thank I. J. Donasch (SPECS GmbH, Berlin, FRG), who contributed basic ideas to the software structure and also major parts of the realization. Cooperation with him and his colleagues is a fruitful stimulus for us.

References

[1] G. v. Egan-Krieger, R. Müller, *The Concept of Modernizing the BESSY Control System*, this conference

[2] R. W. Scheifler, J. Gettys, *The X window system*, ACM Transactions on Graphics, 5, 79 (1986)

[3] M. A. Linton, *Composing User Interfaces with* InterViews, IEEE Computer, 8 (1989)

[4] M. Lee, S. Kleban, *SLC Beam Line Error Analysis Using a Model-Based Expert System*, SLAC–PUB–4539 (1988)