How PC helps to develop new small control system for new small accelerator.

A.S.Chepurnov, A.S.Alimov, D.I. Ermakov, V.I. Shvedunov
Institute of Nuclear Physics, Moscow State University.
F.N.Nedeoglo, V.V.Garbuzov, D.V. Komissarov, N.S.Kochetkov, A.S.Lisjutin, R.E.Shugaley
Department of Physics, Moscow State University.

Abstract

A task to develop control system (cs) for newly designed compact electron linac which being planned to be industrial installation was solved. During the first stage of cs development, flexible modern DAQ boards were installed in conventional PC and were used to control subsystems of the accelerator. To normalize signals for acquisition and generate signals to control, existing analogue blocks were applied. All data acquisition and control algorithms were implemented under PC version of LabView 4.0 together with some simple operator interface to test and study subsystems of the accelerator. The PC, played a role of front-end level, was connected through Ethernet to another remote PC worked under Linux and supported operator interface and simple data-base. During the second, final stage of cs development "industrial type" control system was developed. Front-end PC with analogue electronics was replaced with few members of "Smart device" family -- intelligent front-end devices, while the same operator console under Linux was used. The approach allows parallel work of people involved in developing accelerator hardware, software and hardware of cs.

1 INTRODUCTION

A new cs was developed for new small cw linac. It was necessary to design and construct cs as quick as possible, to use as low financial resources as possible and to make cs as smart as possible. Additional difficulties lied in the fact that new accelerator planned to be constructed from newly designed components: new rf channel with new klystron, new accelerating section, new electron gun, new set of high voltage power supplies. So, cs should support providing of research experiments to study as separate components of accelerator as a whole accelerator. At the same time, the accelerator and it's cs should fit the requirements imposed upon installations for industrial applications. Development of cs was subdivided under two stages corresponding with the stages of accelerator constructing.

2 STRUCTURE OF CONTROL SYSTEM

The cs (Fig.1) consists from three levels -- non real-time top level, soft real-time middle level where relatively slow algorithms are implemented and front-end level with fast control algorithms, hardware locking, fast feedback loops and signal conditioning hardware.

2.1 Control system helps to investigate the accelerator.

The cs fulfilled data acquisition functions other than control functions during the stage of initial testing of different parts of the accelerator: klystron, e-gun, etc. It is important because main task on this stage is to study carefully newly created parts of accelerator as objects to be controlled.

Top level consists of PC-N1 running under Linux 2.1.x. It communicates with middle level through separate segment of Ethernet.

Middle level consists of PC-N2 together with ISA-compatible DAQ add-on boards running under Windows-NT. Boards of two types have the following overall I/O characteristics: 32 channels of ADC with 12 bits 2 μs/channel resolution; 2 isolated channels of DAC with 12 bits resolution and 32 digital I/O. Signal conditioning hardware and galvanic isolation were used between accelerator and DAQ boards. Interlocks and fast feedback loops were implemented in hardware.

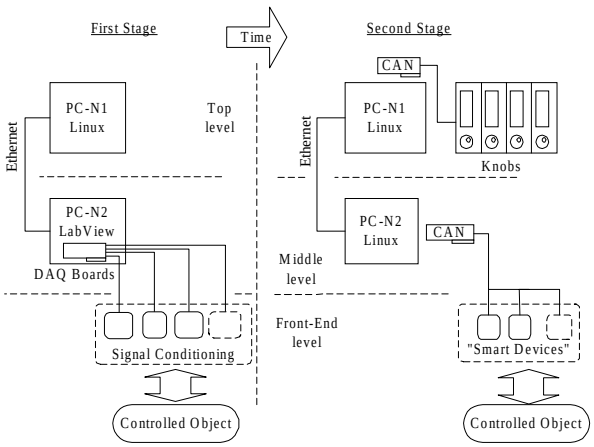

Figure 1. Stages of control system development.

2.2 Control system of industrial style.

The cs should be converted from research style system to industrial style finally.

CAN fieldbus is used on both levels of cs now. To support work of operator knobs-type devices will be used on the PC-N1. CAN adapter installed in PC-N2 controls embedded controllers belonged to family of "Smart Devices"--intelligent controllers which support functions

of real-time digital feedback control, data acquisition and processing [1].

3 CONTROL SYSTEM SOFTWARE

Application software of cs is based on architecture with Distributed Shared Memory (DSM) [1,2].

3.1 Application and system software on PC-N2.

While the software of top level is the same for both stages of cs development, the software of middle level differs from stage to stage.

For research stage of development, application software was developed under LabVIEW 4.0 and consists of three components: data acquisition and processing, software control algorithms (for example, slow PID feedback control), local man-machine interface and server program to support mirroring of DSM. Flexibility of combination composed of LabVIEW and DAQ boards allowed to provide different kind of experiments with accelerator hardware, develop and improve control algorithms.

On the second stage, PCs see as operator as control object through CAN. We have selected DeviceNet high level protocol for CAN -- one of the three most widely used protocols for industrial applications (CAL/CANopen, DeviceNet, SDS). Software supporting DeviceNet for Linux is under development now. Kernel driver for CAN-bus adapter has been developed for Linux. PC-N2 will be run under RT-Linux, so kernel real-time module for RT-Linux for CAN-bus adapter is under development.

3.2 Software of top level.

As was mentioned above, PC-N1 runs under non real-time OS Linux 2.1.x. Man-machine interface is based on X-Windows The structure of top level software is shown on Figure 2.

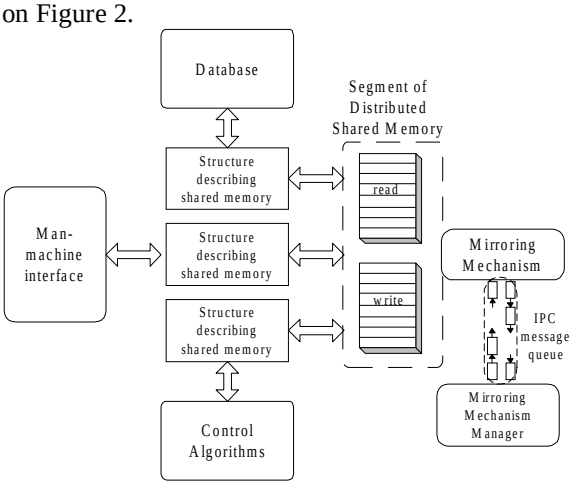

Figure. 2. Structure of top level software.

Modules of application software (man-machine interface, control algorithms, database, etc.) could watch and control an accelerator through segment of DSM. Every module of application software accesses the segment of DSM through individual describing structure. Data from particular blocks of DSM belonged to other components of cs (f.e. modules of front-end level) are reflected in the segment of DSM on top level (Figure.3).

3.3 Mirroring mechanism.

Segments of DSM together with mirroring mechanism are used to reflect data describing current state of an accelerator and necessary for operating of application software. Mirroring mechanism lies hidden from application software. It depends on type of used feildbus and could be implemented partially in hardware as it was done in one of the cs designed previously where MIL-1553B fieldbus was used [1].

Mirroring mechanism was completed in software with Ethernet interlayer communication in cs described in the article. It allows to use different hardware and software components at front-end level during first and second stages of cs development without any software changes on top level.

Software components accessing the segment of DSM, and not responsible for mirroring, could know nothing about inter-level communication construction and not to worry wherever data appeared. This approach ensures rather clear application program interface which simplify work of programmers. It allows develop independently as parts of application software as mirroring algorithm for different types of hardware.

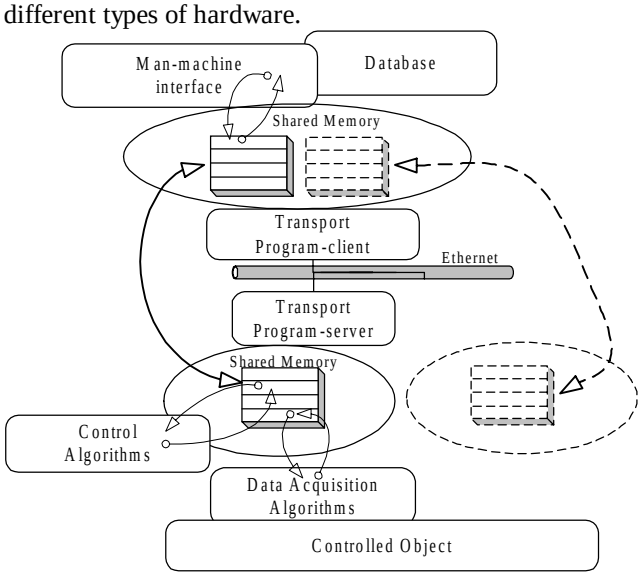

Figure 3. Structure of Distributed Shared Memory.

Mirroring mechanism on top level could be modified dynamically by Mirroring Mechanism Manager -- special-purpose software process. The process controls software module responsible for transportation and replication of different segments of shared memory. Physical properties of measuring parameters and type of

communication are usually clear at the initial stage of cs development and define order and period of replication of different DSM blocks.

3.4 Why Linux?

Notice that during the process of development, Linux should finally replace any other OS running on PC. Linux is one of the dynamically developing modern, POSIX-compatible UNIX-like OS. It is freely available together with its source code and well documented. The idea of free distribution has unified around itself rapidly growing society of Linux users.

Linux OS is going well known and more widely used among teams working under development of cs all over the world. For example object oriented control system TACO supports Linux OS [3]. Client of EPICS could be run under Linux. There are few implementations of CORBA under Linux which is discussed as considerably promising technology for cs development [4,5]. CORBA is one of the object oriented environments for distributed systems which provides the ability to build distributed applications, running on heterogeneous systems, without knowledge of the underlying network.

Our four years experience shows that wide set of application software and tools, high level of reliability and supportability could be achieved with OS Linux in desktop, network and server applications. We use Linux successfully more than two years to support operator console of upgraded rather old control system of electron linac.

3.5 Linux in real-time.

As about application of Linux for operation in real-time, it is impossible to use it “as is” but there are some decisions and projects aimed to improve Linux to be applicable for real-time application [6]. One of the such project is Real-Time Linux, or RT-Linux [7].

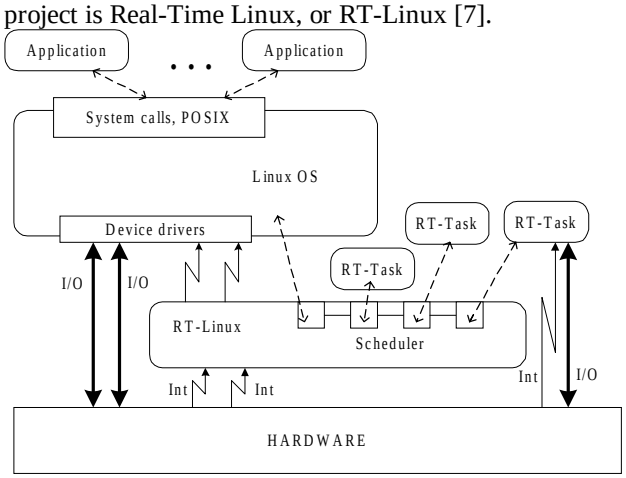

Figure 4. RT-Linux architecture [7].

RT-Linux is operating system in which a small real-time kernel coexists with Linux kernel (Fig.4.). RT-Linux is based on an approach when simple real-time executive runs a non-real-time kernel as it’s lowest priority task, using a virtual machine layer to make the standard kernel fully pre-emptable [8]. All interrupts are initially handled by the real-time kernel and are passed to the Linux task only when there are no real-time tasks to run. Thus, when Linux has disabled interrupts, the emulation software will queue interrupts that have been passed on by the real-time kernel. Real-time and Linux user tasks communicate through lock-free queues and shared memory in current version.

So, application of Linux on top level and RT-Linux on middle level will allow us to use unified software development technology and tools for both real-time and non real-time parts. It would allow to make good use of hardware resources of PCs.

4 SIMPLEX ARCHITECTURE

The Simplex Architecture, a real-time software technology developed at Carnegie Mellon University Software Engineering Institute, was designed to make the cs software changes in a safe and reliable fashion while the system is running [9]. The Simplex Architecture is built upon fundamental concept of analytic redundancy. It is achieved by implementing different types of control algorithms. Basic architecture of Simplex Architecture (Fig.5) contains User Interface module, a decision module, an I/O module and three control modules: a baseline controller, safety controller and an experimental controller.

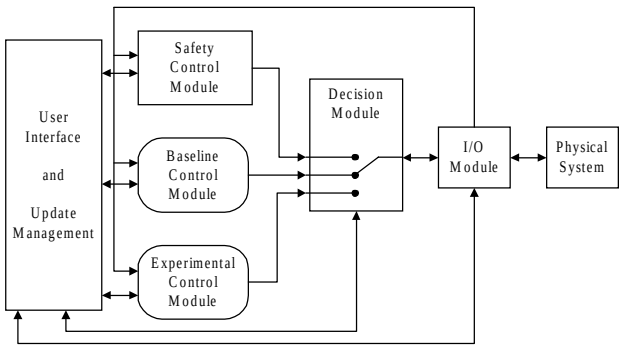

Figure 5. Basic structure of Simplex Architecture.

These software modules run simultaneously and execute control algorithms in real-time. The decision module is a central part of the architecture. Although all three controllers may be running and generating commands simultaneously, only one of the command will be chosen to be sent to the object to control. I/O module measures response of the physical system and distributes them over other processes. I/O module transmits commands from chosen controller to physical system. Each of controllers can control physical system to satisfy certain requirements. The experimental controller contains the system upgrade to be tested. It could be newly designed algorithm, more fast or more optimal but

with uncertain reliability. The safety controller is a high reliable controller that, in a known domain of the physical system state space, it is able to maintain the stability of physical system. But the safety controller may not be able to achieve the control objective of the system in a precise manner. The problem is solved by implementing yet another reliable controller, the baseline controller, which is designed to achieve the control objectives, but to operate within a smaller domain of the state space.

As a control system is upgraded, both the experimental controller and the safety controller are running simultaneously with the experimental controller actually controlling the physical system initially. The state of the physical system is monitored as the system runs. If the experimental controller drives the physical system to an undesirable state, which signifies possible errors in the controller, it will be disabled and the safety controller will take over. While the safety controller is controlling the physical system, the experimental controller can be checked and fixed, and than re-installed to take back the control. Such a process can be repeated dynamically while reliability of experimental controller achieves satisfactory level. So high reliability and high availability can be achieved during the system upgrade without shutting down the system.

During the development of cs for new accelerator one of the task was to stabilise amplitude of rf field in accelerating section. The rf system was newly designed and we did not know exactly dynamic parameters of the system in advance. We used simplified Simplex Architecture and found it very useful. Analogue PID feedback controller designed former time served instead of Safety Control Module. Reference level and closing of feedback loop of analogue controller is controllable digitally. Properties of controller were known well and distinctive level of safe operation with not enough tracking accuracy could be achieved just after starting up. Digital PID algorithm was implemented as Code Interface Node in LabVIEW and played a role of experimental controller. It allowed us to study object of control and successfully develop digital control algorithm for new rf system of new accelerator.

5 CONCLUSIONS

DSM approach has allowed to develop application software for cs more quickly and reliable due to simple API. Future upgrading of the system is simplified too. According our experience, Linux OS looks very attractive as basic OS for development an accelerator cs, but additional investigation and improvement of real-time compatibility should be done. Simplex Architecture could be proposed when safe dynamic upgrading and development of feedback control algorithms are necessary, which is often occur in development cs for accelerator.

REFERENCES

[1] A.S. Chepurnov, A.A. Dorochin, K.A. Gudkov, V.E.Mnuskin, A.V. Shumakov, Family of Smart Devices on the base of DSP for Accelerator Control,// Proceedings of Int. Conf. on Accelerator and Large Experimental Physics Control Systems, W2B-d (Chicago, Illinois USA,1995).

[2] A.S. Chepurnov, A.S. Alimov, A.A. Dorokhin, B.S. Ishkhanov, V.M. Lipgart, V.E. Mnuskin, S.A. Kosterin, V.I. Shvedunov, A.V. Shumakov, Gradual Upgrading of the Accelerator Control// Proceedings of the Int. Workshop on Controls for Small and Medium-Scale Accelerators, p.169 (KEK, Tsukuba, Japan Nov.1996).

[3] A.Götz, W-D.Klotz, J.Meyer, E.Taurel, P.Makijarvi TACO: An object oriented control system for PCs running Linux, Windows/NT,// CD-ROM Proceedings of PCaPAC, October, 1996, DESY, Hamburg, Germany

[4] S.Hunt, B.Jeram, M.Plesko,C.Watson The Implementation of an OO Control System API with CORBA,// Accelerators and Large Experimenta Physics Control Systems,Proceedings of ICALEPCS'97, Science Press, 1998, p.354].

[5] The MICO CORBA Compliant System http://www.icsi.berkeley.edu/~mico/

[6] J. Epplin Linux as an Embedded Operating System http://www.embedded.com/97/fe39710.htm

[7] Real-Time Linux http://rtlinux.cs.nmt.edu/~rtlinux

[8] RT-Linux Manual Project http://rtlinux.cs.nmt.edu/~rtlinux/whitepaper

[9] D.Seto, B.H.Krogh, L.Sha, A.Chutinan Dynamic Control System Upgrade Using the Simplex Architecture, // IEEE Control Systems, Aug.1998., Vol.18, N4, p.72.