Proceedings of PCaPAC 2005, Hayama, Japan

POWERPC-BASED CAMAC AND CAN-BUS CONTROLLERS IN VEPP-5 CONTROL SYSTEM

D. Bolkhovityanov*, E. Gousev, V. Mamkin,
BINP RAS, Novosibirsk, Russia

Abstract

In 2001 BINP began using Motorola MCF5200-based CAMAC controllers with integrated 100Mbit Ethernet and Linux onboard. Those proved to be very effective, but had a serious drawback: MMU-less CPU, which caused use of rather limited uClinux. In 2004 a new version with PowerPC CPU was born, which runs regular PPC-Linux. It is used in VEPP-5 control system since summer 2004 and seems to have many pros and much less cons. A similar device with CAN-bus interface instead of CAMAC is also begin being used. These devices are connected to control PCs via Ethernet. In this paper our experience of using these controllers together with Linux-based host PCs is discussed, as well as our architectural decisions.

HISTORICAL BACKGROUND

For decades most automation in BINP was made via CAMAC.

Rough Classification

CAMAC controllers can be separated into two classes (see Fig.1):

- “Dumb”, which can only execute a signle NAF operation and are connected to a PC (or another computer) via some special link; weird architectures, like chain-connected controllers, also fall into this class.
- “Intelligent” controllers, which contain their own general-purpose CPU, RAM, and a CAMAC interface, and are in fact computers in themselves. Such controllers are able to perform complicated tasks, including complete control of a facility, liberating the host computer from routine tasks.

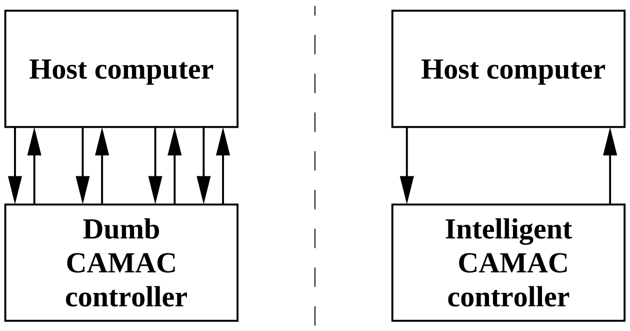

Figure 1: Data interchange between host computer and dumb and intelligent controllers for a typical “channel read” operation.

Historically dumb controllers were much cheaper than intellectual ones, which gave them dominance at least in BINP. But in early 1990s electronics became more and more cheap, and programmers’ time became more and more expensive. So, the main reason to use dumb ones disappeared.

BINP Intelligent Controllers’ History

1980s Since early 1980s BINP used inhouse-designed “Odrenok” controller[1], which in 1997 even adopted Ethernet interface[2]. This device has played key role in BINP automation, but became obsolete long ago. Its main disadvantage is ’70s ICL-1900 architecture, which has 24-bit words (consisting of 4 6-bit “bytes”). Unique programming language and a very specific OS finish the picture of Odrenok’s total incompatibility with modern systems.

1990s In mid-90s BINP made an attempt to base a controller on Inmos RISC-processors T414/T805[3]. This was semi-successful. Those processors are fast (25Mhz, 32bit), and their architecture is designed for multitasking. But very familiar disadvantages remained – everything is unique: unique language, unique communication “link”. And lack of OS forced use of monolithic program image, which in reality turned to be a nightmare – each modification of software requires reboot.

Inmos-based controllers, using modern electronic components, proved to be reliable devices – in a course of a decade, very few of them have died, and all of these because of human mistakes.

So, it became obvious that the most convenient would be a CAMAC controller, looking as close to a “real” computer as possible. I.e., with a “usual” CPU, running a “usual” OS, and employing Ethernet for communication with host computer.

2000s Such controller was designed by BINP electronics department in 2000 – CM5307[4, 5]. It was based on Motorola MCF5200 ColdFire-family processor and used uClinux[6]. It was a giant step in a right direction and was almost what we need. But – only “almost”. It has a great advantage – a Unix-like OS with C as a main language, thus making integration into existing inftrastructure rather easy. The key to its inconvenience was the MCF5200 processor,

* D.Yu.Bolkhovityanov@inp.nsk.su

which lacks an MMU.

While looking innocent, lack of MMU leads to many bad consequences (they are analyzed in more detail in [5]):

- No interprocess protection (and the kernel isn't protected too).
- No shared libraries.
- No `fork()` syscall, only a lame `vfork()`, which instantly makes this system source-incompatible with regular unices.
- A very limited `malloc()` implementation.

Thus, the OS becomes in fact a "DOS with a Posix API".

Anyway, these controllers were (and are) used in significant numbers, and eclipsed old devices in all aspects.

CM5307/PPC FEATURES

From a programmer's point of view it was natural to use some x86-compatible embedded CPU – lots of these are available on the market. And such a CPU could enable binary compatibility with regular PCs. But controller's designer preferred a PowerPC chip. The reasons were:

1. PowerPC is electrically identical to MCF5200, so a simple CPU replacement was enough;
2. PowerPC requires much simpler circuitry than x86.

(Knowledge of x86 architecture weirdness makes anyone beleive the 2nd argument.)

The only serious drawback of PowerPC in an x86 environment is different byte order, but this problem is well-known and is easily solved.

So, in 2004 the CM5307/PPC was born. The CM5307/PPC microcontroller contains:

- 50MHz PowerPC 852 CPU.
- 16M or 32M SDRAM.
- 128K boot ROM.
- 4M flash-disk.
- Real time clock, one RS-232 port and one 100Mbit/s Ethernet controller.
- CAMAC bus controller based on Altera chip.

A potential drawback is a lack of FPU, which has to be emulated in software. But as for now, in VEPP-5 control system all floating-point arithmetic is performed in host computers.

The device can boot either an image from flash-disk, or use BOOTP+TFTP to boot from network.

Choice of operation systems includes RTEMS and Linux. RTEMS supposes a monolithic system image, and in fact has very little advantages over Linux.

Linux being used is a regular Linux-2.4.22 (2.6 kernel is also available, but we haven't tested it yet). As opposed to uClinux, it is 100%-compatible with Linux being used in "serious" computers, and provides multithreading abilities, which were missing in uClinux.

VEPP-5 CONTROL SYSTEM

VEPP-5 employs a 3-layer "standard-model" networked control system, named CX[7].

Control room computers are standard, office-class PCs (currently PIII-800) running Linux, and are connected via Ethernet. Most hardware is CAMAC and CAN-bus. Ethernet is also used as a fieldbus, connecting CAMAC controllers with host PCs. And even CCD-cameras, used for beam diagnostics, are connected via Ethernet.

TRANSITION FROM CM5307 TO CM5307/PPC IN VEPP-5 CONTROL SYSTEM

Building environment

Since the controller's architecture is almost identical, we made a decision to leave software architecture unchanged. So, transition from old CM5307/ColdFire to new CM5307/PPC seemed to be a simple task. Only a new cross-compiler environment had to be added to control system's building infrastructure.

For some reason, builders of cross-compilers somehow manage to produce tools which deviate from standards. That was the case with Lineo's compiler for uClinux/ColdFire, and that was the case with Yellow Dog Linux gcc, which also required some exorcism.

Finally a unified environment was developed, which produces drivers for both CM5307/ColdFire and CM5307/PPC from single source. From sysadmin's/operator's point of view, currently these controllers differ only with a "`uclinux/`" or "`ppc/`" prefix in configuration file.

And this is integrated into CX's tree of sources, thus simplifying the whole control system evolution and maintenance.

Performance

CM5307 hardware provides access to CAMAC controller via memory-mapped I/O. A CAMAC cycle can be performed by reading or writing 32-bit word at address

$$0xF0080000 + (N << 11)|(A << 7)|(F << 2),$$

and status of operation can be read at $0xF00C0004$.

But for ease of use the kernel includes a CAMAC driver, accessible via `/dev/camac`. While introducing some overhead, use of kernel driver is a good thing for many reasons. The most important is trouble-free concurrent CAMAC access by several processes. And kernel driver is the only way for application programs to receive signals upon occurence of LAMs.

In CM5307/ColdFire a single NAF operation via driver took 8μs. In CM5307/PPC this time have raised to 23μs, probably due to a “more serious” Linux, which has longer context switch time. This time is okay for slow devices, but is unsatisfactory for fast ones[1] and for cases when controller’s software has to perform realtime operation with feedback.

That made us look at direct I/O, which provides much better times – 1.5-3μs per NAF. But accurate use of this technique requires some interprocess semaphoring, which is very undesirable.

Finally a slightly modified CAMAC interface operation model was designed, which eliminates most problems. This will be implemented in a near future, thus providing good performance and reliability simultaneously.

RS232 Port

Some industrial devices, such as pyrometers, spectrometers, vacuum pumps, etc., can provide their measurements (and some control) to computer via RS232 interface. But often such devices are located far from any PC. Here CM5307’s builtin RS232 port comes to rescue.

Currently it is used in only one place – for Pfeiffer Vacuum QMS 200 Quadruple mass spectrometer control, but as the number of RS232-enabled intelligent devices at VEPP-5 increases, CM5307’s RS232 will see wider use. And with a simple converter it can drive RS485 devices, which are widespread too.

Present Status

Currently VEPP-5 CAMAC hardware is driven by a mixture of aforementioned controllers.

- There are about a dozen legacy Odrenok devices, which are in a “dying” state.
- 6 CM5307/ColdFire controllers, now working for 3 years without a single fail.
- And more than a dozen CM5307/PPC, gradually replacing old controllers.

All newly installed CAMAC crates for growing VEPP-5 are also equipped with CM5307/PPC.

For us, VEPP-5 staff, and even more – for the whole BINP, CM5307/PPC looks like “the ultimate CAMAC controller”. Because of Linux and Ethernet it fits well into modern computer infrastructure; it is very reliable, it performs well. And it seems to last as long as CAMAC will be used.

[1]CAMAC hardware cycle time is 1μs

CAN-BUS CONTROLLER

CAN-bus in BINP

In the last few years CAN-bus has become the second widely used interface standard in BINP. On some BINP facilities (for example, VEPP-2000) CAN-bus have almost completely replaced CAMAC. The only area where CAN can’t beat CAMAC – devices with huge amounts of data, such as digital oscilloscopes. The long tradition of inhouse development and production of CAMAC hardware has applied to CAN-bus devices too.

Currently BINP uses PCI CAN-bus interface cards[2] produced by Marathon[8]. This adapter provides 2 separate CAN-bus channels driven by Philips SJA1000 chips. While this solution works fine, its future is dubious – PCI bus is being replaced with PCI express, and in the near future PCs with PCI slots will become rare on the market.

CANGW – CAN-bus/PPC driver

A CAN-bus interface with the same core as CM5307/PPC was designed by BINP electronics department, and is called CANGW[9]. It has CAN-bus adaptor instead of CAMAC interface and, of course, a different packaging. It is a small 140$\times$110$\times$40mm box; power supply from a game console is used. From the programmer’s/admin’s point of view this is exactly the same device as CM5307/PPC, so that it requires no extra efforts. And, if such a requirement will ever appear, CAN-bus/PPC can be used with *any* host-machine, not only with PC+Linux.

And this device has one more useful feature – an integrated RS485 port. This was influenced by past experience of small facilities automation at BINP – stepper motor controllers with RS485 interface are in widespread use here.

CANGW was designed as a “CAN+RS485$\leftrightarrow$Ethernet gateway”, thus allowing easy network access to CAN devices from any platform via TCP/IP. However, it is a full-featured microcomputer, which can be used in the control system exactly like CM5307/PPC.

The pre-production model have sucessfully passed testing. Mass production starts now, and VEPP-5 will use these devices in parallel with PCI interfaces.

VARIANTS OF CONTROL SYSTEM’S ARCHITECTURE

Both of aforementioned PPC-based devices constitute a full-featured microcomputer, with virtual memory and a moderately fast CPU. So, as opposed to previous generation – CM5307/ColdFire, the controller itself can run the control system’s server process.

Currently, as a legacy of ColdFire variant, controllers work as “subordinate” devices. The core of control system – CX-server – runs in a host computer, with only “an-

[2]Due to CAN-bus architecture, the term “interface” is more adequate than “controller”.

cillary" drivers executing in controllers. Such sub-drivers (we call them "drivelets") know nothing about the control system – they only execute simple requests from their host.

Placing the whole control system's core into controller looks tempting – this will lower the complexity, and each controller will become self-containing. However, this approach also has some disadvantages:

- Currently the control room's network and controllers' network are separate, and are connected at *one* point by a dedicated host computer. Placing CX-server directly into devices will require either joining the networks, or setting up routing between them. Both variants are undesirable.
- The control system consists of *several* CX-servers, each one controlling its own subsystem (vacuum, magnetic correction, etc.) (Fig.2, a). But each subsystem contains a number of CAMAC crates, CAN-bus and other devices. Moving the "intellect" will split each subsystem between several servers (Fig.2, b), which isn't good too.

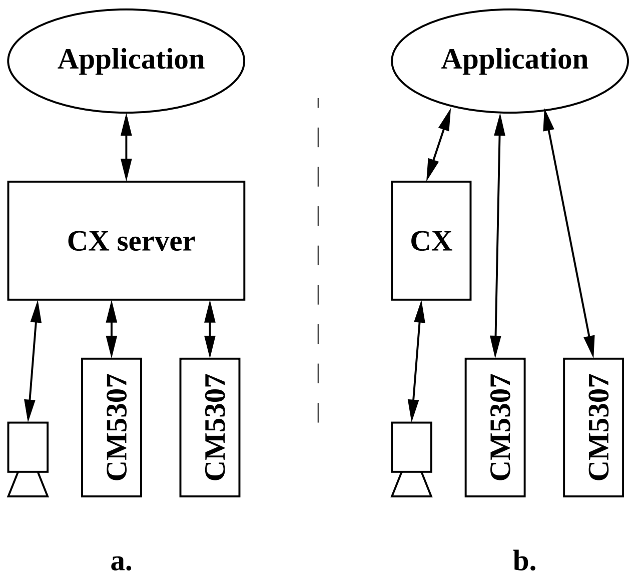

Figure 2: Data links between components of control system in two architectures

So, in VEPP-5 control system we plan to retain the old model. But the "server running in a controller" seems to be an adequate model for automation of small, self-sufficient facilities.

CONCLUSION

PowerPC-based CAMAC controller have proven to be a big success. It will replace all other VEPP-5's CAMAC controllers in the future. Use of Linux – the same OS as in control room computers – significantly simplifies programming and maintenance. This allows to build a highly unified and easy-to-support control system environment. Newly introduced CAN-bus interface with the same core fits excellently into this model.

REFERENCES

[1] G.Piskunov, "CAMAC-embedded 24-bit computer", Autometriya, N4 (1986) pp. 32-38 (in Russian).

[2] A.Aleshaev et al, "VEPP-4 Control System Upgrade", Proc. ICALEPCS'97, http://www.aps.anl.gov/icalepcs97/paper97/p057.pdf

[3] V.Shilo, "Intelligent CAMAC-controller Family Based on the 32-bits Inmos Transputers", Proc. 7th International School-Seminar on Automation and Computing in Science, Engineering and Industry, Yalta, Crimea, 1996.

[4] V.R.Mamkin, "CAMAC CM5307 Controller" (in Russian), http://www.inp.nsk.su/˜mamkin/camac to1.pdf

[5] D.Bolkhovityanov et al, "Experience of Using uClinux-based CAMAC Controllers in VEPP-5 Control System", Proc. PCaPAC'2002, http://www.lnf.infn.it/conference/pcapac2002/TALK/TU-P14/TU-P14.pdf

[6] "Embedded Linux/Microcontroller Project", http://www.uclinux.org/

[7] D.Bolkhovityanov et al, "Evolution and Present Status of VEPP-5 Control System", Proc. PCaPAC'2002, http://www.lnf.infn.it/conference/pcapac2002/TALK/MO-P15/MO-P15.pdf

[8] Marathon Inc., "CAN-bus-PCI" interface http://can.marathon.ru/devices/can-bus-pci.html

[9] V.R.Mamkin, "CAN/RS485 - Ethernet Gateway" (in Russian), http://www.inp.nsk.su/˜mamkin/cangw.pdf