USAGE OF PC CANBUS CONTROLLERS IN VEPP-5 CONTROL SYSTEM

D. Bolkhovityanov*, O. Tokarev, BINP, Novosibirsk, Russia

Abstract

CANBUS is used in VEPP-5 control system for various types of slow controls, such as power supplies management. Finding a PC CANBUS controller was a complicated task, since most of these cards lack a properly supported Linux driver. So, for some time in-house-made CAMAC-based controllers were used. Currently these are being replaced with new PCI controllers. This greatly simplified the software and there are plans to use CANBUS in more areas. VEPP-5 experience with these hardware played a key role in other BINP labs' decisions to use it.

1 CURRENT USE OF CANBUS IN VEPP-5 CONTROL SYSTEM

In the late 90s we began looking at how CANBUS[1], which became very popular that time, could be used in the VEPP-5 control system.

CANBUS, initially designed for car industry, looked very attractive because of the following reasons:

- High reliability.
- Guaranteed delivery time for high-priority packets.
- Devices can be galvanically decoupled from communication media.
- A wide range of CAN devices is available on the market.
- Electronic components for CAN devices are very cheap.
- CAN specification and available hardware allow to control executive devices on long distances.
- In case of CAMAC significant part of device operation logic is performed by device driver, which usually runs in intellectual CAMAC controller. CAN devices, as more modern, are usually able to implement this logic themselves, so that "users" of these devices can be limited to sending simple commands.

While CAN bandwidth is rather limited by modern standards, intelligent executive devices significantly reduce traffic requirements.

In 1999 BINP electronics department began designing devices such as DACs and ADCs with CAN interface[2].

Currently CANBUS is used to control power supplies of VEPP-5 linac magnets[3]. Nearest future plans include using CANBUS for various slow controls:

- Moving thermostabilization system from CAMAC to CANBUS.
- Vacuum control.
- Beam position monitors.

VEPP-5 was one of the first CANBUS users in BINP, and our successful experience allowed other BINP labs to begin using CAN.

While various complicated high-level CAN protocols (like CANopen and DeviceNet) do exist, our tasks didn't require them, and implementing them in electronics would be a useless waste of resources. So, a very simple scheme is used: each CAN device has a unique "address", for which the CAN "identifier" field is used.

2 SEARCH FOR A PC-BASED CANBUS CONTROLLER

Main workhorses of VEPP-5 control system are Linux PCs. So it looked natural to employ some PC-based CANBUS adapters. And there was a strict condition: that adapter must have Linux support.

In 1999 the first CAN adapter card was tested. While it successfully co-operated with BINP-designed electronics, it was an ISA card, and had only Windows driver. And even in 1999 ISA slots were quickly disappearing from PCs.

So it was obvious that PC CAN adapter *must* have a PCI interface. That was the point when the problems began.

CAN wasn't very widely represented on russian market, so out of few available products in 2000 we have chosen IXXAT iPC-320/PCI, as more or less conforming to our requirements.

But... These cards didn't have a driver for Linux. Neither did them have documentation required to develop such a driver. And no reference driver source for *any* OS. And even more: most of the bundled Windows software didn't work.

Negotiations with IXXAT and with IXXAT's russian distributor (Datamicro) gave us nothing — responses to our mail became more and more rare, and it seemed that our e-mails weren't even read attentively. So, we finally gave up with that product.

3 CAMAC CANBUS CONTROLLER

Since CAN electronics for VEPP-5 was already designed and even produced, we had to find *any* working CAN controller. That time BINP electronics department had developed a CANBUS controller for CAMAC[4].

Most of VEPP-5 CAMAC crates employ "Odrenok" intellectual crate controller, which serves as a lower layer of the standard-model-compliant control system[7] (see Fig.2).

So, a rather long line between the CAN "clients" and executive devices had appeared. The request must go from

* bolkhov@inp.nsk.su

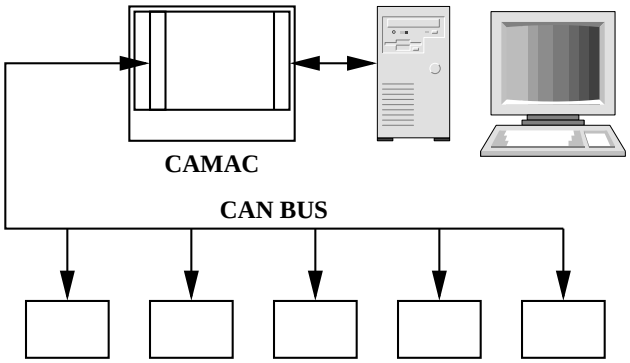

Figure 1: CAN network with CAMAC CANBUS controller

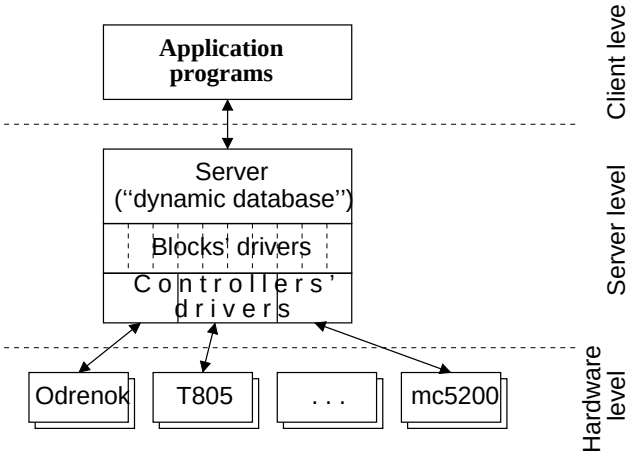

Figure 2: Basic layout of VEPP-5 control system software

the server to CAN adapter's driver in "Odrenok", and than to CAN device. I.e., we have got two steps between a control computer and CAN device instead of one.

This did work, but not very reliably — sometimes packets were lost, and it took some time to find out why. The problem turned out to be trivial buffer overrun — CAN packets arrived to CAMAC adapter faster than "Odrenok" was able to pass them to upper layer.

4 PCI CANBUS CONTROLLER

While CAMAC CAN adapter allowed us to use CANBUS, it was far from ideal. So we continued the search for a PCI CAN adapter. Surprisingly, it was found in Russia!

These adapters are produced by a small company "Marathon" in Moscow[5], and they cost about $300 per unit. Each card contains two Philips SJA1000 CAN controllers, so one card can be used for access to two independent CAN networks. And these cards do have drivers for Linux.

"Marathon" is located in the Institute of Nuclear Physics of the Moscow state university, so we are colleagues. While the company doesn't distribute Linux drivers in a source form, we concluded an agreement, stating that BINP receives a status of tester and gets access to documentation and sources.

We began using PCI CAN adapters on summer 2002, and it was a success. Communication problems, so frequent with CAMAC adapter, had disappeared. One adapter can serve all CAN devices currently used on VEPP-5, and those which are planned in the near future. Control system's drivers for CAN devices were easily rewritten for new adapter — this took a few days. Till now we hadn't experienced any serious problems.

CAN has a very nice feature, which is missing in case of CAMAC. When a new device is added to the control system, it usually needs some testing and debugging. If it is a CAMAC device, software developer must carefully select a time when debugging wouldn't interfere with normal system operation. But CAN, being a peer network, allows to use two host computers simultaneously (see Fig.3). So a new device is first tested from "debugging" host, and when it works fine the main control system's host takes over it. And this operation doesn't require any switching of hardware.

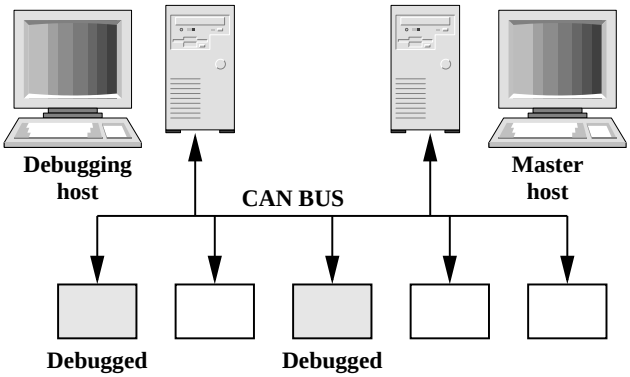

Figure 3: CAN network with two host computers

5 A NOTE ABOUT ADAPTER DRIVER'S I/O UNITS

In Unix most of the file I/O, and CANBUS is no exception, is performed with `read()` and `write()` syscalls. Their semantics is the following:

```
size_t read(int fd, void *buf, size_t count)
```

Here `fd` is the file descriptor which points to CANBUS device, `buf` points to a data buffer. Most interesting for us is `count`. It is a number of *bytes*, which should be read or written (`size_t` is usually just `unsigned int`). On success these syscalls return actual number of bytes processed. This is specified by POSIX standard.

But in case of a driver for Marathon's CANBUS interface `count` is treated as a *number of CAN packets*, not bytes!

This question was discussed with driver's author, and he replied that the driver operates with CAN packets, not

bytes, so it seemed natural for him to use PACKETS as I/O units.

On first sight this argument sounds reasonable: eliminating conversion of units simplifies the, driver thus reducing possibility of errors.

But deviation from standards never goes unpunished!

Most of the control system's drivers don't call `read()` and `write()` themselves, but instead use a dedicated server-wide library, which performs buffered binary I/O "behind the scenes" and calls driver's routines only when a packet is received. But the library doesn't know that in certain cases "`read(fd, buf, 1)`" means "`read(fd, buf, sizeof(can_packet)`"!

This is only one example of such undesired consequences, but others definitely exist and will gave us problems in the future.

Unfortunately, such deviation from standards isn't a rare case. For example, CAMAC driver for CM5307 microcontroller[6] suffers from exactly the same problem — it treats `count` parameter as a number of NAFs to execute. CAMAC drivers sometimes have another bug: devices ("N") are numbered from 0 instead of 1.

The root of the problem seems to be the fact that these drivers were written by hardware designers, not by programmers.

6 "ABSTRACT DRIVERS"

Since PCI-CAN is the second CAN adapter that we use for same hardware, and maybe not the last, we decided to develop "unified drivers", which are able to use any controller.

One solution was to develop an "abstract CAN library" specification, and several conformant libraries — one library for each CAN interface. The "ready-to-use" drivers are made by linking "unified drivers" with some of these libraries.

We have used another approach: each CAN device's driver is a .h-file, defining several functions — Init, Read and Write. And for each CAN adapter there is a special module, which a) knows how and when to call Init, Read and Write, and b) knows how to interact with driver's user (in fact — control system's server). To generate usable drivers those special modules are compiled several times (one time for each driver), #include'ing driver's .h-files.

We have already used the same approach in CAMAC drivers for CM5307 microcontroller[6, Section 6]. So, we call unified CAN drivers "abstract drivers", as in case of CAMAC.

Such solution allows to quickly switch from one CAN adapter to another, and even to use several standards in one server simultaneously.

7 CONCLUSION

CAN turned out to be a good choice of fieldbus for VEPP-5 control system. We have got even more than expected. Majority of the old CAMAC hardware can be replaced with CAN analogues, which are more intelligent. So the control system software would be simplified. Currently the only class of CAMAC hardware which don't fall into this category is digital oscilloscopes and similar devices: these produce large arrays of data, for which CAN is inappropriate.

8 REFERENCES

[1] CAN Specification Version 2.0, 1991, Robert Bosch GmbH, Postfach 50, D-7000, Stuttgard.

[2] V.R.Kozak, "The latest designs" http://www.inp.nsk.su/~kozak/designs/designse.htm

[3] E.Y.Ermolov et al., "Control System for Electromagnet Power Supplies", Proc. ICALEPCS'2001 http://www.slac.stanford.edu/econf/C011127/THAP053.pdf

[4] A.Fisenko, "Technical Description of CANBUS Driver" (in Russian) http://www.inp.nsk.su/ kozak/designs/cbusdrv.pdf

[5] CAN-bus-PCI interface (in russian) http://can.marathon.ru/devices/can-bus-pci.html

[6] D. Bolkhovityanov, R. Gromov, "Experience of Using uClinux-based CAMAC Controllers in VEPP-5 Control System", PCaPAC'2002 poster TU-P14.

[7] D. Bolkhovityanov, I. Pivovarov, O. Tokarev, "Evolution and Present Status of VEPP-5 Control System", PCaPAC'2002 poster MO-P15.