UVX CONTROL SYSTEM: AN APPROACH WITH BEAGLEBONE BLACK

S. Lescano, INSA Lyon, Villeurbanne, France
A. R. D. Rodrigues, E. P. Coelho, G. C. Pinton, J. G. R. S. Franco*, P. H. Nallin,
Brazilian Synchrotron Light Laboratory (LNLS), Campinas, Brazil

Abstract

UVX is a 1.37 GeV synchrotron light source that has been in operation by the Brazilian Synchrotron Light Laboratory (LNLS) since 1997. Its control system, which was completely developed in-house, has received some upgrades lately in order to get around issues from aging, improve performance and reduce maintenance costs. In this way, a new crate controller was designed. It is based on BeagleBone Black single-board computer (SBC) [1], a cheap open hardware and community-supported embedded Linux platform that will be adopted for some control system applications in Sirius [2], the upcoming brazilian light source. In this paper, we describe an overview of the design and results obtained.

INTRODUCTION

As shown in [3], the control system of our machine is organized in three levels. The lowest one comprises 3U VME-like crates with I/O cards and a local controller module, responsible for managing these cards and communications to the upper level. We name this low-level control system based on crates with I/O cards as LOCO (from LOcal COntroller).

Since the start of UVX operation, I/O cards are the same. Most of them has only TTL digital pins and analog inputs and outputs with 12 or 16-bit resolution. There are also specific boards for reading Pt100 temperature sensors or counting pulses, for instance. However, LNLS Controls Group developed three generations of local controller boards, as shown in the next section.

UVX LOCAL CONTROLLERS TIMELINE

In a nutshell, the goal while developing a local controller for UVX control system is always to build a reliable, low-cost and general-purpose module to manage the crates. Since the beginning of control system implementation, another important feature is always present in our controller designs: units of a given model run the same program, a universal software that contains routines to read and write all types of I/O boards and performs all possible operations specified by high-level applications. Running local controllers with a unique software simplifies maintenance tasks and the embedded software development process. With an architecture like that, only high-level applications know the equipments controlled by each crate. Also, the application protocol used in communications to controller crates was specified by LNLS engineers and has never suffered drastic changes.

First Generation: Z80 and Serial Communication

First version of UVX local controller board was based on a Z80 microprocessor. Its software, named PSICO, was written in assembly language. Controller's communication interface was designed internally and is based on RS-485. Although these boards are in operation until today, they are treated as a legacy system, as we don't provide updates for the embedded software anymore and many of their electronic components are obsolete.

Second Generation: eZ80 and Ethernet

Aiming the adoption of a widely used communication interface standard (Ethernet), a new version of the local controller board was developed in the early 2000s [4]. Zilog eZ80F91 microcontroller was the hardware platform chosen. The embedded software was rewritten in C and renamed to PROSAC. Some of these boards are still in operation in UVX storage ring systems, despite the lack of updates for this design in the past years.

Third Generation: SBC with FreeBSD

Because second generation local controller was too much dependent on its hardware platform (an eZ80 microcontroller with built-in Ethernet MAC), the design of a new controller was started. At that time, Controls Group engineers wanted to experiment with the emerging world of embedded Linux platforms too.

An industrial-class single-board computer (Advantech PCM-4153F) was picked. Local controller software (PROSAC) was completely rewritten in C, taking into account libraries such as Pthread (POSIX threads) and the presence of an abundant secondary memory, used for storage of waveforms for special applications. Routines to read and write I/O cards were implemented with the mapping of SBC's PC/104 bus into LOCO crate backplane bus lines.

Although we have moved to an environment which is not hard real-time with this design, in practice all requirements for controller operation in UVX were satisfied. Today we have many of these local controllers under operation, notably those which interfaces to UVX storage ring high current power supplies.

Various flavours of Unix and Linux operating systems were tested for use with Advantech SBC. FreeBSD surpassed all the others because tests revealed that it was more responsive while performing synchronized operations over power supplies (beam energy ramp and magnets cycling).

This local controller based on Advantech single-board computer is the most failsafe we ever made. Machine operators always say that.

* guilherme.franco@lnls.br

ISBN 978-3-95450-189-2

Motivation for a New Design

We have been replacing local controllers of first and second generation by third generation ones for the past years, in order to achieve better reliability during operation or to substitute controllers that presented some serious fault.

Because of Advantech PCM-4153F elevated cost and also intending to evaluate BeagleBone Black in a real operation environment, we started a new local controller design. Main project policy was to run practically the same software as we do on Advantech SBCs, but using a computer which costs about a tenth. Powered with Texas Instruments AM335x processor, BeagleBone Black comes with a Debian Linux core and other two auxiliary real-time cores, the Programmable Real-time Units (PRUs).

HARDWARE DESIGN

We made a simple "carrier PCB" design for BeagleBone Black, just placing in a Eurocard-sized board digital buffers between BeagleBone Black GPIO pins and the LOCO bus signals on crate backplane. The board also has a counter, used during synchronized operations over power supplies (energy ramp and magnets cycling), a 7-segment display for status indication (just as it was since the first generation of local controller boards) and a reset monitoring circuit. Figure 1 shows the design, mounted with panel and handle.

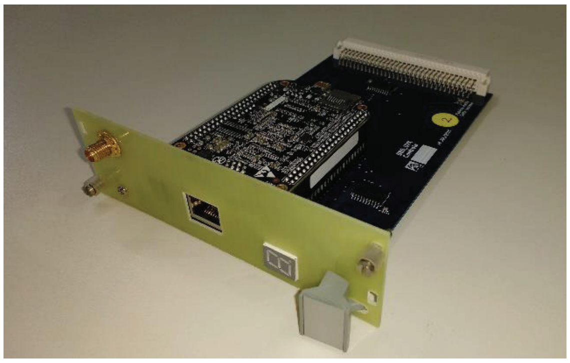

Figure 1: The new local controller prototype.

EMBEDDED SOFTWARE OVERVIEW

General-purpose embedded software for the local controller based on BeagleBone Black is an adaptation of Advantech SBC PROSAC. Porting the software to the new hardware platform involved some tasks.

First of all, we modified the low-level routines used to read and write data through the 8-bit LOCO bus. In the previous design, LOCO bus was accessed through SBC's PC/104 bus (ISA). With BeagleBone Black, bus crate is managed with its GPIO pins. Although we have moved to a quite different hardware approach, the corresponding software changes are minimal. Only a few lines of a C source code file had to be changed.

Moreover, we made minor changes in the code, adapting it from FreeBSD to Linux.

PROSAC Threads Structure

PROSAC runs four threads, described below.

- Main thread: launches the other threads and periodically updates a 7-segment display, which shows local controller status.
- Reader thread: reads continually all inputs from the I/O cards, storing these values in RAM memory.
- Networker thread: deal with client I/O operations through a TCP/IP socket. When the client only wants to read the inputs of the cards, the last values obtained by the reader thread (stored in RAM) are retrieved.
- Interrupter thread: this thread is active when the controller is performing synchronized operations over power supplies. Operation of this thread assumes that controller has in RAM memory a waveform, which is point-by-point traversed each time it receives a trigger pulse from UVX timing system.

Interrupter Thread Detailed

Since the third generation of UVX local controllers, interrupter thread operation is based on a hardware counter, which accumulates the number of pulses received from the timing system since the last actuation and can be read or reset by software. This implementation strategy was adopted because neither FreeBSD nor Linux are real-time systems.

For UVX power supplies synchronized operation, we consider that if the controller can actuate between two successive pulses of the timing system, then it works in a good way. So interrupter thread should always read a value equals to 0 or 1 from the hardware counter while polling its status. A reading greater than 1 means that the controller missed at least one in-time actuation. While designing a new controller, we always try to keep the number of lost pulses equals to zero or as small as possible during synchronized operations.

TESTS AND THREADS TUNING

With the BeagleBone Black "carrier board" prototype, we first tested our embedded software to make sure that main, reader and networker threads were operating correctly. During these first tests, board's 7-segment display worked properly and PROSAC could read all I/O cards currently used in machine operation. Also, tests performed with a standard test client showed that all commands defined in our communication protocol between operations room and embedded controllers worked perfectly.

First tests of synchronized operation of the controller exhibited many situations of loss of pulses. These tests were performed in workbench with pulse trains of most common frequencies used in UVX.

One could think about using BeagleBone Black PRUs to solve the issue. But this was promptly discarded, as it would require a complete restructuring of PROSAC code. In fact, we intend to use the PRUs only for new software developments. And here we only wanted to port an existing software to a new hardware platform.

ISBN 978-3-95450-189-2

In order to evaluate and improve interrupter thread performance under different trigger frequencies, we considered a series of tests exploring PROSAC configuration parameters and important aspects of the embedded operational system (Linux), such as its kernel configuration and scheduling policies and priorities.

Once traditional Linux kernel is not capable of meeting hard real-time requirements [5], our first idea was to apply PREEMPT_RT patch [6] to the kernel. Besides, scheduling policy of all PROSAC threads was set to the real-time option SCHED_FIFO, and their priorities were defined so that the interrupter thread had the biggest priority. Therefore, it could use the CPU without being preempted by any other thread until it explicitly yields the processor. Additionally, we set BeagleBone Black CPU frequency to 1 GHz (maximum allowed), modifying its default CPU frequency scaling configuration. Unfortunately, this proposal didn't lead to expected results, and controller continued to loss timing system pulses despite all these efforts. With the traditional Linux kernel, SCHED_FIFO scheduling policy wasn't able to decrease the number of lost pulses too.

Given the failure of PREEMPT_RT patch combined with a real-time scheduling policy approach, our second try was to keep the traditional Linux kernel and use its Completely Fair Scheduler (CFS), while running BeagleBone Black CPU at 1 GHz. In opposition to the scheduler of the first proposed solution, which allows a process to retain the processor as long as it wants, CFS tries to give a fair amount of CPU usage to each process or thread, taking into account a parameter called nice value.

Threads with smaller nice values are allowed by the scheduler to consume more processing time. For this reason, setting interrupter thread niceness to the minimum (-20) grants more CPU time for this thread only when it effectively needs, and does not starve the other threads out completely. Furthermore, we fixed main thread niceness to the maximum allowed (+19), since it requires no important processing at all (it just refreshes the 7-segment display), and reader thread niceness to +15, as it spends the majority of its processing time waiting for inputs.

Using this approach, we varied networker thread nice parameter and the number of I/O cards connected to the test crate in order to investigate the impact of periodically bus readings and client TCP/IP requests over system's overall performance. In general, results were very good. Number of lost pulses changed with the number of I/O boards in the crate. Variations on networker thread nice value didn't influence it. The only disadvantage of this configuration is that crate boards reading rate was degraded during synchronized operation. For instance, we obtained reading rates as small as 7 Hz in some tests.

In order to increase the number of input readings during synchronized operation, a new solution was proposed. We kept nice values of interrupter, main and reader threads as before, set niceness of networker thread to +10 and changed the way reader thread works. When under normal operation, reader thread competes for CPU time with all the others. Under synchronized operation, it is disabled, and a complete reading of all inputs is performed after every controller actuation, inside interrupter thread. As a result, we achieved a reading rate equal to the frequency of received pulses, without increasing the number of lost pulses. Table 1 summarizes obtained results for two trigger frequencies. Tests were performed with 2 I/O cards on crate and a TCP/IP client requesting 1000 readings per second.

Table 1: Interrupter Thread Performance

Frequency (Hz)	Total pulses	Lost pulses
512	5,376,065	8 (0.0001 %)
1000	6,050,225	18 (0.0003 %)

CONCLUSION

Tests showed that BeagleBone Black can serve as a hardware platform for UVX local controllers. The new design has an acceptable performance during synchronized operation, comparable to that of the previous one. We have already put in operation two local controllers based on BeagleBone Black. Their functioning is satisfactory. A batch of printed circuit boards was ordered for design replication.

We are also considering to use another SBC in UVX controller modules: BeagleBone Green, which is fully compatible and cheaper than BeagleBone Black. The main difference between these two boards is the lack of a video output interface on BeagleBone Green. Since a video interface is completely useless for our applications, this is not a problem. We have tested in workbench the new UVX local controller prototype with BeagleBone Green too, and results obtained are the same.

REFERENCES

[1] BeagleBoard.org, `http://beagleboard.org`

[2] J. P. S. Martins *et al.,* "Sirius control system: design, implementation strategy and measured performance", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 456-459.

[3] J. G. Franco *et al.*, "LNLS control system", in *Proc. ICALEPCS'99*, Trieste, Italy, Oct. 1999, pp. 651-653.

[4] J. G. R. S. Franco *et al.*, "Upgrading the LNLS control system from a proprietary to a commercial communications environment", in *Proc. EPAC'04*, Lucerne, Switzerland, Jul. 2004, pp. 530-532.

[5] R. Love, "Process scheduling", in *Linux Kernel Development*: Addison-Wesley, 2010, pp. 64-65.

[6] Real-Time Linux, `https://wiki.linuxfoundation.org/realtime/start`

ISBN 978-3-95450-189-2