SIRIUS CONTROL SYSTEM: DESIGN, IMPLEMENTATION STRATEGY AND MEASURED PERFORMANCE

J.P.S. Martins, M. Bacchetti, E.P. Coelho, R.F. Curcio, J.G.R.S. Franco, R.P. Lisboa, P.H. Nallin, A.R.D. Rodrigues, L.D.S. Sachinelli, M.E. Silva – LNLS, Campinas, Brazil

Abstract

Sirius is a 3 GeV synchrotron light source that is being built by the Brazilian Synchrotron Light Laboratory (LNLS) [1]. The Control System will connect and operate all the equipment along the whole machine. The main goal of the Sirius Control System is to be distributed and digitally connected in order to avoid analog signal cables. A three-layer topology will be used [2]. The equipment layer uses RS485 serial networks, running from 6 to 12 Mbps, with a light proprietary protocol and CPU boards with proprietary hardware stacked on it, in order to achieve good performance. The middle-layer, interconnecting these serial networks, is based on Beaglebone Black single board computer and commercial switches. Operation layer will be composed of PC's running EPICS client programs. Special topology will be used for Orbit Feedback with a dedicated commercial 10Gbps switch. This paper will discuss the details of the Control System components, the implementation strategy for hardware and software and show some results of the prototypes.

INTRODUCTION

The Sirius Control System is designed to be scalable, distributed and easy to maintain. Currently, we are developing a generic solution for the hardware of the Control System – analog and digital interfaces, and also serial communications interfaces – to provide control features for the most of Sirius systems: vacuum system, pulsed power supplies, magnets power supplies, RF system, etc. The main characteristic of this solution is to be compatible with commercial low cost CPU boards. We achieve this using the SPI standard interface. The very first prototype for the Sirius Control System hardware was a homemade ARM CPU board with analog and digital modules, called PUC (Universal Control Board). Several tests have been done with this board and modules, and some results will be shown in this paper.

Besides the basic input/output, the hardware supports special features for synchronous operations, like curves for energy ramping of the booster and cycling magnets, and a circular buffer for post mortem acquisition.

With this strategy, the Controls Group will provide a low budget generic solution for hardware and software, which is reliable and of good performance in order to meet the requirements of operation of the various systems from Sirius.

HARDWARE IMPLEMENTATIONS

The main goal of the hardware solution provided by the Controls Group is to be easily adaptable to the equipment being controlled, avoiding the traditional "crate mounting" for a distributed installation, and sometimes inside the own equipment, preventing analog signals to travel over long distances.

Hardware Platform - GESPICON

This platform is under development and aims to be independent of a custom CPU board (like the PUC Base Board). Thus, any commercial CPU board, even the Beaglebone Black, can be used as embedded controller for the interface modules; it just has to have an SPI peripheral and some GPIOs. Figure 1 shows the components of the platform, called GESPICON (Generic SPI Controller). The connection of the stacked boards with the controller is done by a special bus, called SPIxxCON bus. This bus has the traditional SPI pins for serial data interchange, and some GPIOs for configuration of the transactions. It has also the pins for power supply and the optional use of quad-SPI features. The protocol of this bus is capable of addressing 8 module boards with 8 SPI devices in each one. Thus, one CPU controller can manage up to 64 SPI devices attached to it.

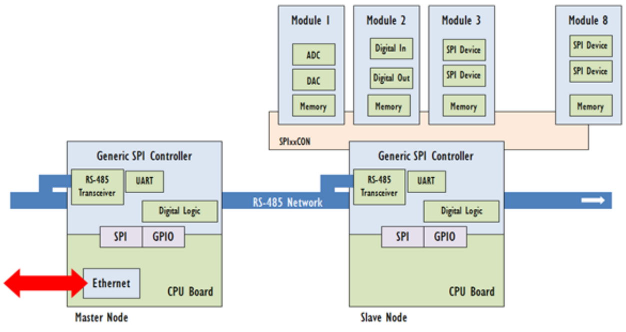

Figure 1: Schematic of the GESPICON hardware platform.

Another important feature of this implementation is the possibility to embed SPI flash memories on the interface modules. Thus, any interface can have static information used by the controller for auto-configuration. At boot time, the CPU module will search for each stacked module memories and download the data on it, which can be:

- Source code (Python or C) for the driver interface of the module;
- EPICS parameters for IOC configuration;
- Any parameter of the module devices;
- BSMP entities description (defined next page);

This system makes the CPU software independent of the interface modules drivers. Any module that would be developed in the future will be compatible with the CPU,

ISBN 978-3-95450-148-9

with no need of software upgrades. These features increase the modularity and the flexibility of the whole system.

PUC CPU Module

The PUC (Universal Control Board) was the very first prototype developed by Controls Group. The CPU module (called “base board”) is a homemade device embedded with an ARM Cortex M4 microcontroller, with 256 KB of RAM memory and 2 MB flash memory, running at 180 MHz. The base board also has flash and RAM memories (external from microcontroller), which are used to save waveforms and to hold fast acquisition data. The main communication interface is the RS485 serial, running at 6 Mbps baudrate. Another communication facility is the Ethernet interface. The MAC (Media Access Control) layer is embedded in the microcontroller and the LwIP (Lightweight IP Library) implements the TCP/IP stack.

Analog and Digital Modules

The digital board has 8 digital inputs and 8 digital outputs, TTL level, for digital commands and readings.

The analog board has one ADC (AD7634) and one DAC (AD5781) both with 18 bits resolution, for analog input and output. The working range of the converters is from -10V to +10V. This means that the theoretical resolution is 17 bits for 0 to +10V and 16 bits from 0 to +5V. This module was specially designed due to the fact that these kind of high precision interfaces are unusual to find commercially.

Three basic tests were done with these interfaces: long term stability, linearity and repeatability. The setup for all tests was three PUCs with one analog board module stacked. The analog output was connected to analog input of the same board. The measurements were done by a 2-channel Agilent 34420A 7½ digits multimeter and a HP 34401A 6½ digits multimeter. The multimeters were configured with an integration time of 100 NPLC (Number of power line cycles), and the ADCs with an integration of 20k samples over one second.

Long Term Test was performed setting the analog output once and measuring the multimeters and analog input during 24 hours. Some results are presented in Fig. 2 (analog output) and Fig. 3 (analog input).

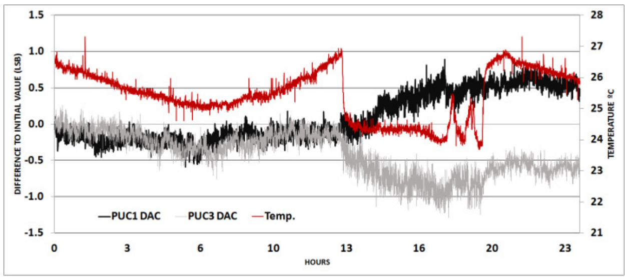

Figure 2: Long term of analog outputs from PUC1 and PUC3, fixed on -9V and +9V, respectively, along with temperature (right axis).

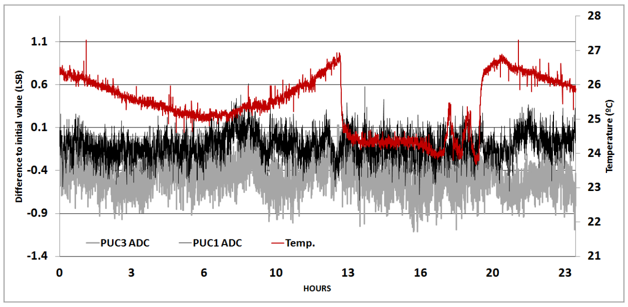

Figure 3: Long term of analog input from PUC1 and PUC3 (with temperature).

The result shows that stability is quite adequate. The drift of the voltage references of the converters increases as the output reaches the full-scale value (10V), but stays in a reasonable value.

Linearity Test was performed setting the analog output in 5 ranges of voltages (-9V, -5V, 0V, 5V and +9V), with 512 points around the center values. Table 1 and Table 2 below present some results of analog output module differential non-linearity and integral non-linearity tests, respectively.

Table 1: Analog Output Module Differential Non-linearity

DNL	PUC1		PUC2		PUC3	
Range	Min (LSB)	Max (LSB)	Min (LSB)	Max (LSB)	Min (LSB)	Max (LSB)
-9V	-0,235	0,232	-0,165	0,168	-0,396	0,402
-5V	-0,118	0,117	-0,090	0,098	-0,165	0,177
0V	-0,283	0,289	-0,275	0,273	-0,039	0,019
+5V	-0,138	0,138	-0,085	0,085	-0,161	0,170
+9V	-0,193	0,203	-0,156	0,184	-0,295	0,254

Table 2: Analog Output Module Integral Non-linearity

INL	PUC1		PUC2		PUC3	
Range	Min (LSB)	Max (LSB)	Min (LSB)	Max (LSB)	Min (LSB)	Max (LSB)
-9V	-0,096	0,762	-0,280	0,122	-0,717	0,104
-5V	-0,226	0,129	-0,248	0,059	-0,158	0,217
0V	-0,366	0,000	-0,409	0,000	-0,017	0,083
+5V	-0,082	0,257	-0,134	0,157	-0,133	0,263
+9V	-0,224	0,294	-0,304	0,096	-0,249	0,428

The results of INL and DNL of all boards are adequate, staying in ±0.5 LSB range for the analog outputs, and ±1.5 LSB range for the analog inputs.

Repeatability Test keeps the same setup. The analog outputs were set repeatedly with the following values:

ISBN 978-3-95450-148-9

+9V, +5V, 0, -5V and -9V, during 12 hours, at each update, the multimeters and ADCs were read. No missing steps were found in all measurements. The results of the relative error at the 5 ranges are presented in Fig. 4 below.

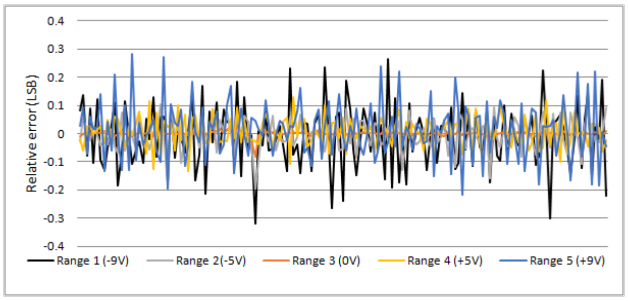

Figure 4: Repeatability test of PUC1 analog output (relative errors).

Beaglebone Black

The Beaglebone Black [3] is an open hardware single board computer based on ARM Cortex A8 running at 1 GHz. The rich set of features of this hardware makes it a good solution for an embedded platform:

- 512 MB DDR3 RAM, 4GB on-board eMMC flash storage;
- Linux operating system support;
- Flexibility and modularity;
- Fanless;
- 2X real-time 32 bits microcontrollers (PRU processors);

The Beaglebone Black fits perfectly to be used as a high level platform for communication with the Control System Hardware, and also as a CPU board for the custom analog and digital interfaces of the Control System.

To implement the fast RS485 interface, we designed a cape board for the Beaglebone Black using an external UART (MAX3107), which will reach up to 12 Mbps baudrate. The interface to this component is done by the PRU (Programmable Real-Time Unit), a real-time processor embedded in the ARM CPU of the Beaglebone Black (which has two PRU units). To optimize the software design, we created a C library to interact and manage the PRU firmware. This library has an API for sending and receiving bytes from the external UART, and also for configuration of the device. The result is a high-performance serial device with great processing power and all the facilities of an embedded Linux system.

SOFTWARE IMPLEMENTATIONS

The Sirius Control System will use EPICS middleware, one of the most used frameworks in Particle Accelerators Controls System. For the low level communication, we developed a light proprietary protocol, called BSMP (Basic Small Messages Protocol).

BSMP (Basic Small Messages Protocol)

The BSMP describes two layers of the network model: transport and application. The transport layer transmission unit is the *packet*. Each packet holds a *message*. The packets in a serial network have a well defined format: one byte for destination address, n bytes for the message and one byte for checksum. The addressing can be individual or in multicast groups. The protocol message in the application layer has the first byte for command, two bytes for size and finally n-bytes for the payload. Thus, the whole BSMP packet has only 5 bytes of overhead, for payloads up to 65535 bytes.

Devices using the BSMP protocol manipulate protocol entities, which can be up to four categories: variables, group of variables, curves and functions. To manage and implement the BSMP protocol, a C library (and a Python binding) was developed. The API of the library implements all the routines for data encapsulation, de-encapsulation, entities values attribution and protocol validation. To develop a new hardware and use BSMP, the developer only needs to configure the corresponding entities.

EPICS Device Support

In order to use the EPICS facilities, we developed a Device Support for the PUC prototype. An EPICS Soft IOC for the Beaglebone Black master was configured, because the low level hardware access is done via message based devices. The Device Support driver was created using asynDriver framework [4]. In order to embed the BSMP and PRU Serial libraries on the driver, we developed an Asyn Driver Support, managing the low level communication tasks. Above the driver support, there is an Asyn Port Driver in C++, implementing all the parameters that the hardware could have: analog I/O, digital I/O, waveform I/O and static configurations. The same strategy can be used to develop the EPICS Device Support of the new hardware over the SPIxxBUS. The strategy of message-based devices also permits the use of other software for low level interfaces and data structure, like Redis IO suite [5].

SYNCHRONOUS OPERATIONS

Sirius Control System must support synchronous procedures for many of the machine systems. As a main example, there is the booster energy ramp, performed by the magnet power supplies and RF systems. The Control System hardware supports synchronous operations broadcasting serial messages over the RS485 network, in order to reduce the number of signals and cables from the Timing System. Thus, the triggers only need to reach the single board computers, which will send the broadcast high-priority messages over the serial network, as fast as possible.

To validate this strategy, several tests were performed. The object of measurement is the latency between the reception of the trigger on a single board computer and a flag assertion on a slave node of the RS485 network. The first test uses a Beaglebone Black with our homemade RS485 cape as the master and three PUCs as slave nodes. The trigger signal was a 3.3Vpp pulse, 5% duty cycle, 5 kHz, produced by a HP 3314A Function generator. As soon as the trigger is detected at the Beaglebone Black GPIO (PRU Unit), the master sends a high priority serial packet over the RS485 network at 6 Mbps baudrate. The flag used to measure the reception of the node is the trigger for the DA converter of an analog interface module. This setup emulates the booster energy ramp. We run the experiment 4 times, acquiring 2 minutes of data in each run. Table 3 shows the average of the obtained values from the 4 runs.

Table 3: Latency Measurement Using Beaglebone Black as Master and PUC as Slave

Latency	Min (us)	Max (us)	Avg (us)	Std. dev (ns)
PUC 1	13.91	14.00	13.94	17.13
PUC 2	13.94	14.00	13.97	14.54
PUC 3	13.95	14.04	13.98	18.45

Other tests were done to evaluate the Beaglebone Black as master and as slave on the RS485 network. In this setup, the trigger signal is generated by the own Beaglebone Black master node, using the PWM peripheral. It's a 3Vpp pulse with 2% duty cycle and 100 Hz. This signal is in a loop back to a GPIO of the PRU processor, which sends the broadcast serial message over the network (at 6 Mbps baudrate). The slave Beaglebone Black PRU detects and decodes the broadcast message, raising a flag when the packet is decoded. The results of the measurements of the reception latency are presented on Table 4.

We also tested the Beaglebone as master and slave but running the software from the ARM core, under Linux environment. The latency increased to hundreds of microseconds and the jitter increased to tens of microseconds, which invalidates this topology.

Table 4: Latency Measurement Using Beaglebone Black as Master and Slave, Running Firmware on PRU Processor

	Run #	Min (us)	Max (us)	Avg (us)	Std. dev (ns)
PRU (master) x PRU (slave)	1	23.89	23.99	23.94	18.91
	2	23.89	23.99	23.94	18.73
	3	23.89	23.99	23.94	19.36
	4	23.89	23.99	23.94	18.97

The repetition rate of the Sirius Booster for top-up operation is 2 Hz. The specification of the ramping curves is about 1000 steps each curve. Thus, for 2 Hz operation, the period of each step is 500 us. The results of the first two tests prove that the system can achieve the necessary performance for booster operation. Even if the repetition rate increases to 5 Hz, (with 200 us each step) the latency and jitter of the serial broadcast triggers match the requirements. The results can be improved when using 12 Mbps baudrate on the serial network.

Another feature of this topology is the selective interrupt configuration, when the slave nodes respond to specific broadcast messages, allowing more flexibility for the profile of ramping curves. This can be useful for machine operation.

CONCLUSION

The design of the main components of Sirius Control System was presented. The implementation strategy is to provide a robust, modular, flexible and distributed hardware platform for machine operation, in addition to EPICS compatibility on the software side. The analog modules characterization tests are quite reasonable. The synchronous operations support proved to match the specification of the Sirius booster energy ramp. The next designs under development by the Controls Group will increase the performance and flexibility of the whole system.

REFERENCES

[1] L. Liu et. al, "Update on Sirius, the new Brazilian Light Source", MOPRO048, Proc. IPAC2014, http://jacow.org
[2] J.G.R.S. Franco et. al, "Sirius Control System: Conceptual Design", MOMIB01, Proc. ICALEPCS2013, http://jacow.org
[3] http://www.beagleboard.org/black
[4] http://www.aps.anl.gov/epics/modules/soft/asyn
[5] http://redis.io