MICROIOC AND EPICS: THE COMPLETE CONTROL SYSTEM NODE

R. Sabjan, D. Golob, M. Plesko, A. Pucelj, M.Sekoranja, Cosylab, Slovenia

Abstract

The Experimental Physics and Industrial Control System (EPICS) [1] is a widely used particle accelerators control system. Being a product of a large scientific based collaboration, the required knowledge and experience to start working with EPICS represents a significant entry barrier. Furthermore, the traditional VME hardware is very costly for small, dedicated applications. By introducing a product which serves as a plug&play, PC-based Input Output Controller (IOC) and provides effortless integration in the control system, the microIOC together with its driver set and configuration software signals a significant step forward in making EPICS more user-friendly and accessible. microIOC, a package consisting of industrial hardware, popular control system (apart from EPICS, we can offer also variations with ACS or TINE) and Cosylab's significant software contribution, is ideal for integration of "problematic" protocol based devices (serial and GPIB) where there is important to isolate such devices from interfering with the rest of the control system. In addition, microIOC offers a wide range of communication and I/O modules and is currently installed at the Swiss Light Source and will also be used for the Australian Synchrotron Project.

INTRODUCTION

EPICS is a set of software tools, libraries and applications developed collaboratively and used worldwide to create distributed soft real-time control systems for scientific instruments such as a particle accelerators, telescopes and other large scientific experiments. Such distributed control systems typically comprise tens or even hundreds of computers, networked together to allow communication between them and to provide control and feedback of the various parts of the device from a central control room, or even remotely over the internet.

Although EPICS is the most popular among particle accelerator control systems it also has its shortcomings. The most obvious may well be its complexity and rather steep learning curve. Newcomers to EPICS are continually tormented by all the details needed to set up a working EPICS database. Even experienced developers spend significant amount of time preparing a development or testing environment.

Traditionally, EPICS is run on the VME-based platforms, such as the Motorola Power PC (PPC) [2], running vxWorks operating system [3]. These systems use VME boards to interface to controlled devices. VME boards are expensive, but offer very high performance and a large number of channels. This proved to be very good and price efficient for large system with a high density of channels. When it comes down to a small or extremely distributed control system, VME hardware can be also seen as a significant overkill in terms of performance and especially the price. Empty or half-empty VME crates significantly raise the cost of a control system.

There are additional clear cases that manifest a clear need for a small, standalone IOC in a control system:

- Close proximity to the controlled device is needed (e.g. due to limited cable lengths)
- Sensitive devices are controlled – influence of other systems needs to be minimized
- Problematic devices – minimize the influence on other systems (example: a GPIB driver may hang the whole VME crate)

Fortunately, recent developments in EPICS have enabled the developers to run the EPICS database also on non vxWorks platforms. One of the most popular is indeed Linux. Systems that do not need hard real-time control (such as serial and GPIB interface systems) can be better (cheaper) covered by PC based Linux system.

THE MICROIOC IDEA

In order to produce a good and reliable IOC and to achieve customer satisfaction, many factors need to be considered. We believe that commercial off-the-shelf (COTS) products ensure high variety, low price and a lasting supply of spare parts. It is very important to realize that a normal PC is not the best for the task. A much better solution is an embedded system, which is designed for use in industrial environments and longer lifetimes.

Such parameters are achieved only through a careful design and use of industrial grade components and avoiding all moving parts:

- Fan less CPU
- No hard disk – booting from a Compact Flash (CF) disk
- A higher quality power supply

To reduce network dependence of the system and reduce the overall network load during booting, locally stored software (operating system, EPICS, runtime application) should be employed. CF disks provide a cost effective and robust solution, providing that the software is modified in such a way that there is no continuous writing on the disk.

But an exceptional IOC is much more than just a piece of hardware and operating system. A rich and efficient software toolkit has to be provided to the end developer. The goal is to eliminate all interfacing and installation effort, but for providing the most important information, such as configuring the network parameters, I/O ports and sorting out naming conventions. Dedicated *wizards* will make use of pre-prepared templates and device support routines to deploy a working control system node. Additionally, example GUI screens, alarm handler and archiver configuration files would also be provided and made available using the web server installed on the IOC.

ARM BASED SERIAL DEVICES CONTROLLER

At Cosylab we already have experience in porting Linux and EPICS to various platforms. The first successful prototype was developed in 2003 in collaboration with the Swiss Light Source at the Paul Scherrer Institute [4].

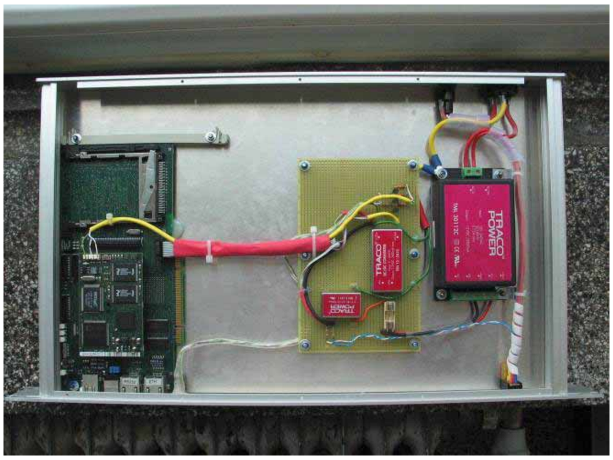

Figure 1: Arm based microIOC in 2003.

The first microIOC has featured a 200 MHz StrongArm processor, booted Linux from a Compact Flash disk and was able to control 1 device over an RS232 port. It was installed to control an electronic RF phase shifter. No problems were reported until it was replaced by a Generation 2 microIOC in October 2004.

A custom in-house developed Linux distribution was used; updates were being made by uploading new software over the serial port after being cross compiled on a normal PC Linux platform. Telnet server was installed for remote access and the Network Time Protocol (NTP) was put in place to synchronize the local time with a time server. A lot of effort was dedicated to make all software run with limited resources that were available.

THE MICROIOC OF TODAY

Several significant upgrades and important design decisions were made before upgrading to the microIOC of today, sometimes referred to also as the Generation 2 microIOC.

Hardware

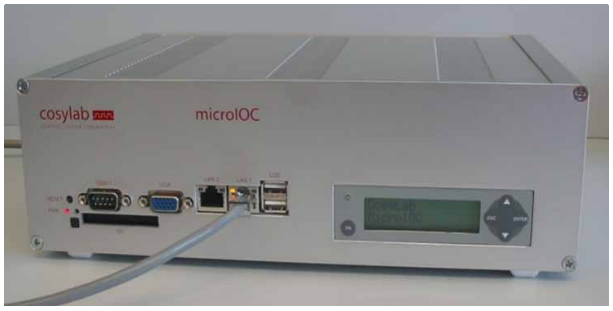

Figure 2: Generation 2 microIOC.

The most notable difference is the move to the PC (x86) platform. With this we have widened the spectrum of available models and suppliers and lowered the manufacturing costs. Software development was made easier as we were able to use a more common Linux distribution. Taking these facts into the account, we were able to dedicate more effort into the features that are important to the end customer.

The microIOC is based on an embedded Single Board Computer (SBC) with the following features:

- PC/104 bus for I/O cards
- Two 10/100 MBit Ethernet ports
- 2 USB ports
- VGA connector
- 2 on-board RS232 ports

The PC/104 standard features an ISA bus with stackable industrial grade connectors. A wide pool of suppliers provides various PC/104 extension cards.

A wide range of extra I/O and peripherals are routinely being used:

- Addition serial ports (RS232/422/485) – up to 16 ports
- GPIB port
- Analog I/O
- Digital I/O
- USB camera
- LCD Display

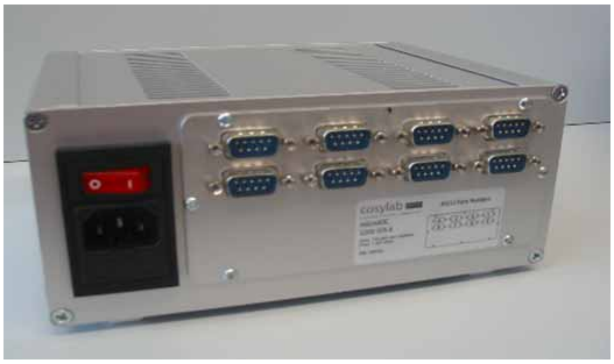

Figure 3: Serial ports on the backside.

The microIOC can be connected to a monitor and keyboard at any time. Additionally, the front side serial port also serves as a user console.

The microIOC comes in three different form-factors. Apart from the mid-sized version (Figure 2), a smaller or a 19” rack version may be ordered.

Software

Debian Linux distribution [6] is used for the operating system. Debian is very good at handling software versions, thus reducing maintenance costs. On the other hand, Debian also offers a very stable distribution with virtually no bugs.

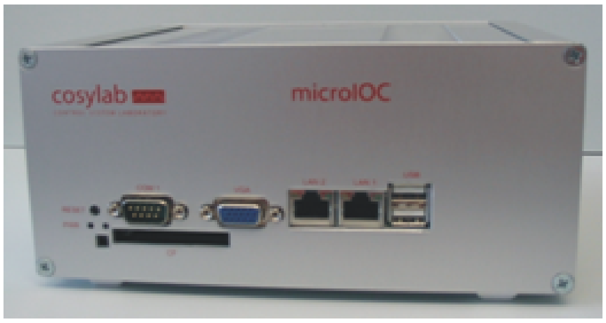

Figure 4: Small enclosure.

EPICS version 3.14.6 comes installed with the microIOC. Asyn driver [7] is the basis for all asynchronous device supports. In addition to all this, EPICS sequencer programs can also be used.

Every microIOC comes with a “system health” database, which enables monitoring of critical system information, such as IP, CPU usage, memory usage, boot time etc.

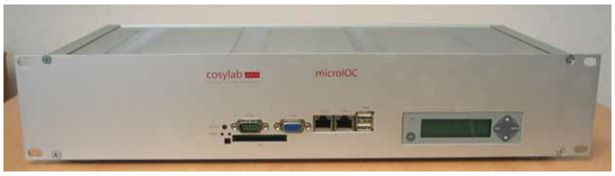

Figure 5: 19" Enclosure.

LCD display is also supported in EPICS and displays channels, their values and units. Displayed channels are easily configured in software and navigated by pressing the buttons on the LCD display. In case of alarms, the LCD flashes and the alarm can be acknowledged by pressing a button on the LCD.

A web server runs on each microIOC, displaying the system health data as well as the I/O ports configuration (baud rate, parity and similar for the serial ports). It is also possible to alter this configuration very simply with just a few mouse clicks.

The microIOC can mount NFS filesystems for data storage (e.g. for save/restore [8]). The correct date and time are acquired using NTP.

The development environment works on regular Linux workstations (tested on Debian and RedHat 9). Thus, during initial development, a microIOC is not even required. The development environment takes care of building device support and other custom code. Packaging of database and object files into installable Debian packages. Reproducible deployment of packages on microIOC provides additional added value to the customers and maintainers.

Installations

The microIOC platform is being used at SLS and several other labs. The single largest installation – the booster control system for the Australian Synchrotron Project – is in the installation and commissioning phase. As many as 20 microIOCs are being deployed to control devices over serial line, GPIB or Ethernet.

Figure 6: ASP booster development stand.

MICROIOC AND ACS

ACS (*Advanced Control System/ALMA Common Software*) is a modern, CORBA based, object-oriented infrastructure for control system deployment. ACS enabled microIOC has an *ACS Container* pre-installed and pre-configured to work with an *ACS Manager* of choice. When a microIOC is connected to the network, it reports to the *ACS Manager* and the components hosted by the microIOC are immediately available to the rest of the control system.

FUTURE PLANS

Some of main points on our to-do list are still making the microIOC more a “black-box” IOC. This will only become possible when a lot of different devices will be supported through drivers, device supports and database templates. Some wizards were already implemented (configuration of serial ports through a web page), but more need to follow.

We are also considering using RTEMS operating system for hard real-time applications.

ACKNOWLEDGEMENTS

We would like to express out gratitude to all people who contributed to ideas with respect to embedded IOCs.

We are indebted especially to SLS and the people of the controls group which have generously contributed ideas [10] and also funded the development of the first (StrongArm) prototype. Without their help, all this would not be possible.

Additionally, we would like to thank the whole EPICS community, especially to the core developers of EPICS base and Asyn driver. Marty Kraimer was most helpful when we encountered numerous problems.

Bob Dalesio provided much needed steering and deserved criticism. Without him, we would be offering a much worse product.

REFERENCES

[1] http://www.aps.anl.gov/epics
[2] http://www.motorola.com
[3] http://www.windriver.com
[4] http://www.psi.ch
[5] http://www.pc104.org
[6] http://www.debian.org
[7] http://www.aps.anl.gov/epics/modules/soft/asyn
[8] http://www.aps.anl.gov/aod/bcda/synApps/index.php
[9] http://www.rtems.org
[10] R. Sabjan et al., “ EPICS IN A BOX: THE MICROIOC”, PSI Scientific Report 2004