PC-Based IO Controllers from a VME Perspective

J. O. Hill, LANL, Los Alamos, USA

Abstract

The Experimental Physics and Industrial Control System (EPICS) has been widely adopted in the accelerator community. Although EPICS is available on many platforms, the majority of sites have deployed VME- or VXI-based input output controllers running the vxWorks real time operating system. Recently, a hybrid approach using vxWorks on both PC and traditional platforms is being implemented at LANL. To illustrate these developments we will compare our recent experience deploying PC-based EPICS input output controllers with experience deploying similar systems based on traditional EPICS platforms.

1 INTRODUCTION

About two years ago our site came to the awareness that Microsoft compatible components are becoming an increasingly attractive choice as building blocks for experimental physics control systems. Our initial attraction was certainly low cost, but primarily the wider selection of commercial components, and a potential "portability" of skills between desk top environments and control systems installations.

The Experimental Physics and Industrial Control System (EPICS) is a toolkit for deploying network distributed process control systems, a source code distribution, and an unprecedented collaboration of 100+ sites world-wide. These sites work together to produce shared software and also a pool of trained consultants which are frequently utilised to rapidly build momentum in embryonic projects. The system is traditionally staged from a UNIX development environment. The input output controllers in the system (IOCs) are typically running the VxWorks real time operating system on VME or VXI based hardware. These systems include a process control database consisting of function block records that are abstract software encapsulations of IO channels and algorithms that can be connected together in a modular fashion.

Recently, the EPICS system has been ported to the PC platform. In particular, under Microsoft Windows (WIN32) the EPICS client, server, and utility libraries are now available. An X Window System based operator console and synoptic editor has been ported to run native on WIN32 platforms when an X window system server has been installed. The EPICS application development environment and the associated portable source code build system based on GNU make and PERL have also been modified to be compatible with WIN32.

This paper will limit its scope to a comparison of PC-based EPICS IO controllers running the VxWorks real time operating system with similar systems based on VME.

2 POSITIVE ASPECTS

Since manufactures sell a larger number of PC-based than VME-based components, then they can recover their PC related development costs with a lower per-unit profit. Likewise we expect a larger variety of commercial I/O components to be available for the PC environment. At our site we also appreciate that there is only one software model for all PC motherboards, whereas there are typically many different software models for each of the many different VME-based single board computers (SBC). Finally, software developers appreciate the presence of hardware breakpoints in the Intel architecture which we commonly used on VAX processors, but have missed with the Motorola 68k and SPARC processors commonly used on VME platforms.

3 NEGATIVE ASPECTS

Mass-market PCs provide only a limited number of PCI and ISA slots. In contrast, VME allows for a much larger number of slots. Both the ISA and PCI busses employ card edge connectors which are expected to be unreliable compared to the pin and socket type connectors employed by compact-PCI and VME bus systems. Of course when there are compelling requirements on card-to-card noise immunity or requirements for specialised analogue supply voltages then a VXI-based system may be required. The reliability of mass-market PCs in an industrial environment is still (for us) unproven.

The primitive interrupt architecture of the ISA bus, having no concept of an interrupt vector, makes the integration of a large number of input output cards into a PC-based system difficult. Typically one Intel interrupt vector is assigned to each ISA interrupt level. Since many of the mass-market PC interrupt levels are already assigned to the various mother board peripherals then we are left with a marginal number of remaining interrupt vectors with which to interface with all of the input output cards. This frequently results in the chaining of multiple interrupt handler routines on each of the remaining interrupt vectors, and a resulting degradation of deterministic interrupt latency and efficiency. There are also differences in the assignment of PCI interrupt vectors on the PC motherboard. In figure 1 we see a best case interrupt routing scenario where each PCI interrupt source is provided to the programmable interrupt router as a separate input, and the inputs for ISA bus interrupts are completely segregated from the PCI interrupts. This

allows the programmable interrupt router to assign independent vectors to each PCI interrupt source[1]. In figure 2 we see a worst case scenario where all of the PCI interrupt requests are wired together and then routed to individual ISA interrupt lines attached to a functional stand-in for the legacy PT/AT cascaded 8259A programmable interrupt controller. Since this device is typically programmed in a PC environment for edge sensitive interrupts then care must be taken to generate a new edge on the interrupt request line before leaving the interrupt service routine if for any reason the interrupt is still asserted. If not, it is possible that an interrupt event will be lost.

Figure 1: Improved PC Interrupt Routing

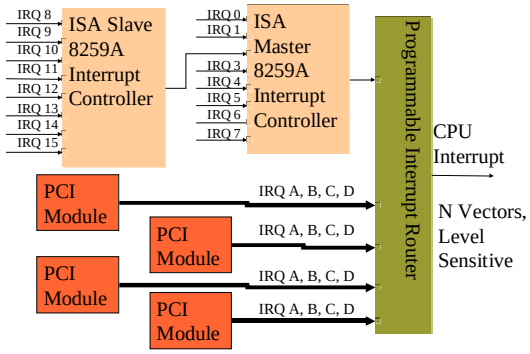
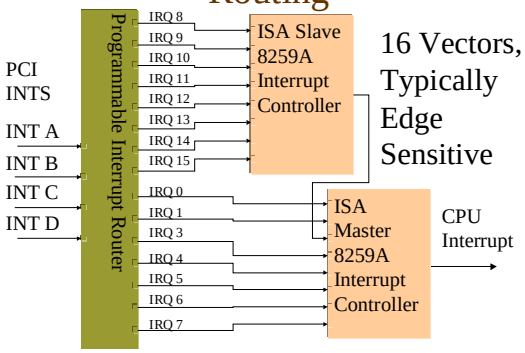


Figure 2: Legacy PC Interrupt Routing



With VME-based systems we find that remote network access via a terminal server console connection is convenient, and sometimes necessary in systems with a wide geographic distribution. Terminal server access allows us to interact with the system just before and also during the boot process when the network interface of the VME computer has not yet initialised. We find that this type of access is available to PC-based systems running VxWorks with the exception that we cannot configure a mass-market PC BIOS from a terminal server. This is because the BIOS typically will not communicate without a display adapter card, a monitor, and a keyboard physically present at the computer. Likewise, these components are also required when we must configure an input / output card from DOS or Windows.

It is necessary to repackage mass-market personal computers for use in industrial environments. Improved packaging allows for rack mounting, larger more reliable power supplies, improved cable paths to signal conditioning, and increased cooling.

Finally at our site a frequent determining factor in the decision between VME and PC platforms is the availability of an existing device driver in the EPICS distribution for the type of IO that will be used. We often find that the cost of writing a new device driver is more than the cost reduction associated with employing a PC-based IOC.

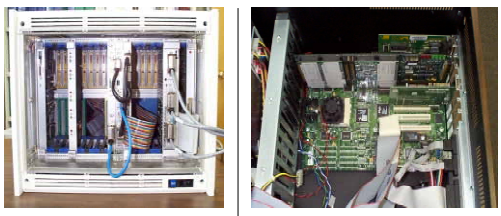
4 COST

In order to compare PC-based IOCs with VME-based IOCs we will compare the hardware costs in three hopefully representative situations. Table 1 shows the base cost of systems that include a single board computer (SBC), an IO back plane, a power supply, and an enclosure. The primary cause of higher costs for VXI systems is the increased cost of the VXI SBC. VXI SBCs, while functionally very close to VME SBCs, are manufactured in much smaller quantities than VME SBCs and perhaps this is responsible for their increased cost. Figure 3 shows a picture of the two system side by side.

Table 1: Typical Base System Costs

Type	US $
VXI	12k – 15k
VME	7k – 10k
PC	1k - 3k

Figure 3: PC-based IOC Versus VXI/VME I/O Controllers



Standard VME-based IOC
$7,000 - $10,000

New PC-based IOC
$1,000 - $3,500

The first representative system, dense replicated IO, will compare the cost for an IOC with a maximum number of analogue input channels. In a VME system we

can pack a maximum of 1280 analogue input channels while with a mass-market PC system with 4 ISA slots we can fit only 128 analogue input channels. In Table 2 we see that the per channel costs for these two systems are about the same, but with the mass-market PC system we don't have enough IO channels to keep a state of the art CPU busy.

Table 2: Dense Replicated IO Per Channel Cost

Type	Signal Count	US $
VME	1280	44
PC	128	54

The second representative system, diverse IO, where there is a wide variety of different IO types, but not a high packing density is the most common situation at our site. For comparison purposes I examined the cost of a system with one IO card for each of binary, analogue, GPIB controller, and stepper motor controller. The price for this system is shown in Table 3.

Table 3: Diverse IO System Costs

Type	US $
VME	10.4 – 13.4k
PC	4.4k – 6.4k

The third representative system, Multi-Drop IO, where all of the IO are physically present on a field bus is also common at our site. For comparison purposes I examined the cost of a system with one IO card which interfaced with the Allen Bradley Data Highway Plus. In this situation a low cost 3rd party Data Highway Plus Interface ISA card was available for the PC, but for the VME system only the more expensive original vendor product could be located. The price for this system is shown in Table 4.

Table 4: Multi-Drop IO system Costs

Type	US $
VME	8.5k – 11.5k
PC	1.7k – 3.7k

We must also include a brief comparison of software costs. The EPICS function block database, which is the heart of an EPICS IO controller, was run on the VxWorks real time operating system in the performance tests that follow. The VxWorks development system is expensive compared to development systems for Microsoft Windows, however run time licence costs for VxWorks and Microsoft Windows are comparable. We use a real time operating system instead of Microsoft Windows because we need low interrupt latency, fast context switching, resulting improved communication bandwidth with IO, simplified device driver development, and improved reliability.

5 PERFORMANCE

We compare the performance of several representative systems for EPICS function block processing, interrupt latency, and EPICS event latency. Table 5 shows the systems that were used in the performance tests that follow. Performance numbers for the mass-market PC in this table were obtained with the GNU 2.7.2 486 compiler which did not have a Pentium switch. The VME performance figures that follow, and our performance test procedures, were obtained from another paper[2].

Table 5: Performance Test Platforms

Type	Processor	Clock Rate	Vendor
VME SBC	MC68040	25 MHz	Force SYS68k/CPU40
VME SBC	PPC 604e	200 MHz	Force Power Core 6604
Mass-Market PC	Pentium I	166 MHz	Micron Millennia

EPICS function block record processing performance was measured by examining the CPU load presented by 300 EPICS function blocks processed at 1, 2, 5, and 10 Hz processing rates. One operator interface client (MEDM) was attached, and the CPU load was measured with the VxWorks "spy" utility. The results are in Table 6.

Table 6: Function Block Performance - % CPU load

Rate	MC68040	PPC 604e	Pentium I
1 Hz	5.8 %	0.6 %	1.0 %
2 Hz	12 %	1.2 %	2.5 %
5 Hz	27 %	3.0 %	4.3 %
10 Hz	56 %	6.1 %	8.0 %

In real time systems another important performance metric is interrupt latency. We performed a classic interrupt latency measurement where a binary input was generating an interrupt, and this caused an interrupt service routine to change the state of a binary output. The interrupt latency was determined by viewing this binary output on an oscilloscope that was triggered by the interrupt generating binary input. Table 7 shows the measured interrupt latency for the systems in Table 5.

Table 7: Interrupt Latency

System	Latency in µS
VME Bus — MC68040	5.4
VME Bus — PowerPC 604e	6.0
ISA Bus — Pentium I	5

In the EPICS system an important performance metric is latency from when an EPICS event is posted until when a function block record is processed. A binary input caused an interrupt, and this caused an interrupt service routine to post event zero with the EPICS "post_event()" subroutine. Two EPICS function block records were processed whenever event zero was posted. The first record set a binary output bit to one, and the second record set it to zero. The event latency was determined by viewing this binary output on an oscilloscope that was triggered by the interrupt generating binary input. Table 8 shows the measured EPICS event latency for the systems in Table 5. We don't

have a good explanation for the large difference between the Power PC and Pentium results in this table.

Table 8: EPICS Event Latency

System	First Record Latency µS	Second Record Latency µS
VME Bus — MC68040	73	122
VME Bus — PowerPC 604e	12.5	19.5
ISA Bus — Pentium I	50	90

6 RELIABILITY

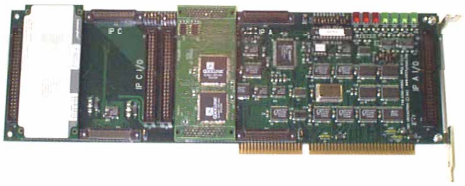
There have been no documented hardware failures at our site over the past year when controlling high power RF PLCs with mass-market PCs. Reliability will be highly visible in the next few months as our LINAC commissioning progresses.

7 EFFICIENT USE OF BOTH PLATFORMS

In our control system we have a mixture of PC- and VME-based IO Controllers. In order to reduce development and maintenance costs we maximize shared software and hardware components between the two systems.

Examples of portable hardware interfaces used in both system types include serial IO, Allen Bradley Data Highway, and Industry Pack IO. Industry Pack IO was a good fit for systems with a wide diversity of low density IO. We have found that a expansive range of cost competitive modules are available in this format, and that the simplicity of the computer bus structure facilitates rapid development of custom hardware modules. All Industry Pack modules use the same standardised plant side IO connectors, and this simplifies the wiring to and from signal conditioning. Figure 4 shows a picture of an ISA Industry Pack Carrier board. Two of four available industry pack slots are occupied on this board. The left most slot is occupied by a commercial input / output board, and the 3rd slot from the left is occupied by an in house designed timing board.

Figure 4: IPAC Carrier and Modules



Examples of portable software interfaces utilised at our site include the C STDIO library for portable serial device drivers, and the UKIRT / Gemini abstract Industry Pack IO carrier library for VME / PCI / VXI / PCI portable Industry Pack module drivers. To improve device driver portability we have implemented an abstract application programmers interface which provides logical to physical address mapping, interrupt vector linkage, and interrupt enable functionality. Device drivers that use this library avoid compromising their portability with details from the local operating system.

8 CONCLUSIONS

In our experience PC-based IO controllers are a reasonable choice compared to VME-based IO controllers in well-selected situations. PC-based IO controllers appear to be a good choice for systems employing diverse low density IO, and systems interfacing with multi-drop IO. Appropriately repackaged mass-market PC-based IO controllers have been reliable at our site for one year. Nevertheless, compared to VME-based IO controllers, mass-market PC-based IO controllers, have limited slot counts, primitive edge connectors, and an interrupt architecture which complicates software. We have found that a judicious selection of hardware interfaces, and a careful organisation of software layers, has allowed the efficient coexistence of PC- and VME-based systems at our site.

REFERENCES

[1] T. Shanley, Don Anderson, “PCI System Architecture” , Addison-Wesley, 1995

[2] J.Odagiri, A.Akiyama, N.Yamamoto, T.Katoh, “Performance Evaluation of EPICS on PowerPC”, Proc. ICALEPCS, Bejing, China, 1997, pp 602-611.