VMEBUS-BASED DATA ACQUISITION WITH REAL-TIME UNICOMPATIBLE CISC AND RISC PROCESSORS — SOME PERFORMANCE MEASUREMENTS

J. Petersen, C. Dessonet, C. Parkman, Y. Perrin, L. Tremblet
CERN, Geneva, Switzerland

ABSTRACT

A real-time UNIX system (LynxOS) available for VMEbus CPU modules based on CISC (MC68030) as well as RISC (MIPS R3000) processors has been evaluated in the area of data acquisition. The use of a portable data acquisition system as a measurement tool has allowed a comparison to be made between the LynxOS and the OS-9 operating systems running on the same hardware platform (MC68030). In addition, the real-time performance of MIPS R3000 under LynxOS is compared with that of a MC68040 based OS-9 system. Some basic real-time features (e.g., interrupt latency, semaphores) of LynxOS and OS-9 are described in more detail.

INTRODUCTION

In high-energy physics (HEP), and at CERN in particular, the OS-9 real-time operating system is widely used for VMEbus-based data acquisition. Recently, another operating system (LynxOS) well adapted to real-time applications in VMEbus, has become available. Due to its main features: real time, UNIX compatibility, portability amongst many types of processors and POSIX compliance, LynxOS has received some attention in the VMEbus users community; it has, for example, been chosen by groups at CERN for applications in particle accelerator control.

In this report, we evaluate the performance of LynxOS in the area of real-time data acquisition on CISC and RISC-based VMEbus CPU modules and compare with the performance of OS-9. This evaluation has been made by employing a portable data acquisition system as a measurement tool. The portability was achieved by structuring the data acquisition system as a system-independent kernel with well-defined software interfaces to the environment, i.e. the underlying operating system, the physics readout system and the recording devices.

The measurements are presented in terms of "number of events per second", i.e. the number of triggers which can be handled per second by the system (event overhead). This number, which defines in a condensed form the data acquisition performance (DAQmark) of the system, is, of course, a function of a large number of hardware and software parameters, processor and bus speeds, memory wait states, compiler and real-time features of the operating system, etc. However, to the end user of a data acquisition system the global performance of the system is more relevant than the performance of its constituents.

Finally, it should be pointed out that our principal aim has not been to achieve the highest possible performance on any given system but rather to compare the performance of data acquisition systems implemented on different platforms, using "standard" methods. Therefore, under LynxOS as well as OS-9 the implementations are based on standard facilities offered by the operating system for task synchronization, I/O interfacing, etc.

Presented at the Open Bus Systems 1992, ETH, Zurich, Switzerland
13–15 October 1992
INFRASTRUCTURE – HARDWARE AND SOFTWARE

The timing measurements have been performed using the simple hardware set-up shown in fig. 1.

![Hardware configuration diagram]

Fig. 1 Hardware configuration

The physics input/output is represented by a CAMAC crate interfaced to VMEbus via the CBD8210 (CAMAC branch driver) [1]. Experiment triggers, in the form of LAM (look-at-me) interrupts, are generated by a BORER 1802 CAMAC dataway display module. Several combinations of CPU modules — based on CISC or RISC processors — have been employed for the measurements, as shown in table 1.

<table>
<thead>
<tr>
<th>VMEbus CPU</th>
<th>Operating system</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LynxOS</td>
</tr>
<tr>
<td>MVME147, MC68030, 25 MHz</td>
<td>✓</td>
</tr>
<tr>
<td>RAID8235, R3000, 25 MHz</td>
<td>✓</td>
</tr>
<tr>
<td>MVME167, MC68040, 25 MHz</td>
<td></td>
</tr>
</tbody>
</table>

The CPU boards are all commercially available, general-purpose VMEbus modules:

- MVME147 from Motorola [2] is used by several software companies as a reference VMEbus board for the implementation of their MC68030 based operating systems, e.g. OS-9 from Microware and LynxOS from Lynx Real Time System Inc.

- MVME167 from Motorola [3] is the MC68040 based successor to the MVME147 and is available with OS-9.

- RAID8235 from Creative Electronics Systems SA [4] is a VMEbus single-board computer based on the MIPS R3000/R3010 RISC chip set. It is available with the TC/IX operating system from CDC [5]. This variant of UNIX is based internally on the LynxOS real-time kernel and we shall refer to this system as LynxOS in the following.

OS-9 from Microware Systems Corp. [6] is a general-purpose, real-time, multi-tasking UNIX flavoured operating system. The most characteristic feature of OS-9 is its modularity: the kernel, drivers, file managers and application programs have the form of modules which (in most cases) can be added or removed from the system dynamically. This is in striking contrast with the monolithic structure of UNIX.

LynxOS from Lynx Real Time Systems Inc. [7] is the first real-time POSIX compliant UNIX available for VMEbus-based CPU modules. Its major features are:

- Portability between processors (MC68030, i80486, MIPS R3000, Sparc 2, etc.).

- UNIX compatible (System V and BSD 4.3).

- Real time: the kernel is fully pre-emptive to guarantee a deterministic interrupt response which is a qualitatively new feature compared to earlier “real-time” versions of UNIX.

- POSIX compliant.

The availability of the systems shown in table 1 have allowed us to make several comparative measurements:

- OS-9 versus LynxOS on the same hardware platform (MVME147).

- RISC (MIPS R3000) versus CISC (MC68030) under the same operating system (LynxOS).

- MC68030 versus MC68040 running the same operating system (OS-9).
OS-9 on a fast CISC (MC68040) versus LynxOS on a RISC MIPS R3000.

We have run the Dhrystone benchmark program on each of the systems in table 1 to evaluate their basic processing speeds. The program was version 2.0 and the "gcc" compiler was used in all cases. The results are shown in table 2.

Table 2 Dhrystone benchmarks

<table>
<thead>
<tr>
<th>Processor module</th>
<th>Operating system</th>
<th>Dhrystones</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVME147, MC68030, 25 MHz</td>
<td>OS-9</td>
<td>4760</td>
</tr>
<tr>
<td>MVME147, MC68030, 25 MHz</td>
<td>LynxOS</td>
<td>5560</td>
</tr>
<tr>
<td>MVME167, MC68040, 25 MHz</td>
<td>OS-9</td>
<td>16100</td>
</tr>
<tr>
<td>RAID8235, R3000, 25 MHz</td>
<td>LynxOS</td>
<td>23500</td>
</tr>
</tbody>
</table>

DATA ACQUISITION SYSTEM

SPIDER is a portable general-purpose, multitasking data acquisition system designed to be simple, fast and adaptable to various hardware and software environments; SPIDER is used at CERN in several experiments for simple tests and front-end acquisition.

The major elements of SPIDER are four tasks (some optional) organized around a number of buffers as depicted in fig. 2.

The producer task reacts to a trigger from the experiment and writes data into a buffer in the form of an "event", consisting of a header plus data read from the experiment. The trigger is then cleared and the task suspends itself, waiting for the next event trigger. Optionally, the recording task writes buffers to a magnetic tape and the analysis task allows for monitoring a user selectable minimum sample of events. The control task interacts with the user and controls the execution of the other three tasks.

The portability of SPIDER is obtained by using a Virtual Operating System (VOS) interface [8] to the underlying host operating system. VOS, inspired by early versions of POSIX, was developed at CERN originally to provide portability between VMS and UNIX; it has later been implemented on a number of UNIX-like systems as well as OS-9.

SPIDER relies heavily on semaphores to synchronize the various tasks and to control their access to global resources. An example of task synchronization is the need to suspend and resume the event producer as a function of available buffer space. Another example is the protection of critical sequences to guarantee indivisibility of certain instruction sequences or exclusive access to (global) variables.

SEMAPHORES

In this section, we describe the types of semaphores available under LynxOS and OS-9 and study their performance.

When mapping the VOS semaphore primitives (lock/unlock) into the system calls of LynxOS, there are several possibilities. In addition to UNIX System V semaphores, LynxOS provides its own lightweight counting semaphores as well as POSIX semaphores. In many applications the LynxOS semaphores — referred to below as “fast semaphores” — is a viable alternative to using UNIX System V semaphores.

Table 3 displays timing figures for the fast LynxOS semaphores, called directly or via the VOS interface, whilst table 4 shows some figures for the UNIX System V semaphores. The measurements were made by executing program loops on the semaphore wait and signal operations and with only one user process active in the system.

Table 3 LynxOS fast semaphores

<table>
<thead>
<tr>
<th>VMEbus CPU</th>
<th>Time per semaphore call (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVME147, MC68030, 25 MHz</td>
<td>47</td>
</tr>
<tr>
<td>RAID8235, R3000, 25 MHz</td>
<td>7.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VMEbus CPU</th>
<th>Time per semaphore call (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVME147, MC68030, 25 MHz</td>
<td>Direct 60</td>
</tr>
<tr>
<td>RAID8235, R3000, 25 MHz</td>
<td>Direct 10</td>
</tr>
</tbody>
</table>
Table 4 LynxOS UNIX System V semaphores

<table>
<thead>
<tr>
<th>VMEbus CPU</th>
<th>Time per semaphore call (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct</td>
<td>via VOS</td>
</tr>
<tr>
<td>MVME147, MC68030, 25 MHz</td>
<td>118</td>
</tr>
<tr>
<td>RAID8235, R3000, 25 MHz</td>
<td>28</td>
</tr>
</tbody>
</table>

It is observed that:

- UNIX System V semaphores are a factor three to four slower than the fast semaphores.
- The overhead due to VOS for the fast semaphores is about 25%.
- The semaphore calls on the MIPS R3000 are about six times faster than those of the MC68030, clocked at the same speed.

Since we want to compare OS-9 with real-time UNIX we have made the SPIDER measurements with versions of VOS based on fast semaphores.

Under OS-9, VOS semaphores are implemented using OS-9 counting semaphore (events). Table 5 shows some OS-9 timing figures for OS-9 system calls via the VOS interface. The overhead due to VOS is negligible.

Table 5 OS-9 semaphore (events)

<table>
<thead>
<tr>
<th>VMEbus CPU</th>
<th>Time per semaphore call (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct</td>
<td>via VOS</td>
</tr>
<tr>
<td>MVME147, MC68030, 25 MHz</td>
<td>135</td>
</tr>
<tr>
<td>MVME167, MC68040, 25 MHz</td>
<td>35</td>
</tr>
</tbody>
</table>

It is seen that:

- The OS-9 semaphores are about two times slower than the LynxOS fast semaphores on the same platform (MVME147).
- The semaphore calls on the MVME167 are about four times faster than on the MVME147 at the same clock rate.

CAMAC

Portability of the SPIDER producer is achieved by interfacing to CAMAC via the NIM/IEEE/ESONE standard CAMAC routines. This library can be implemented using different methods. Under OS-9 and LynxOS we have chosen a strategy whereby the CAMAC I/O system is considered to be a sharable system resource. Access to CAMAC and LAM interrupt handling is therefore made through drivers. Since CAMAC is memory-mapped into the VMEbus address space, another approach is to consider CAMAC as an extension to the processor memory space and to allow CAMAC access directly from user programs; this is more efficient but it does not provide the protection offered by the driver method.

Under LynxOS a UNIX-like CAMAC driver was developed with LAM interrupts handled by the interrupt service routine and CAMAC access via the “getstat” entry point of the driver. Under OS-9, interrupts go through a proper driver while CAMAC access is made in a system state trap handler. The OS-9 driver and trap handler are written in assembler. The implementation language has no significant impact on the timing.

Table 6 summarizes the timing of CAMAC on the different systems listed in table 1. The timing figures are obtained using a VMEbus backplane analyzer [9] and cross-checked by programmed timing loops, wherever possible.

Table 6 CAMAC timing

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Time (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>t1</td>
</tr>
<tr>
<td>MVME147, OS-9</td>
<td>150</td>
</tr>
<tr>
<td>MVME147, LynxOS</td>
<td>200</td>
</tr>
<tr>
<td>RAID8235, LynxOS</td>
<td>6</td>
</tr>
<tr>
<td>MVME167, OS-9</td>
<td>55</td>
</tr>
</tbody>
</table>

- t1: is the time from the occurrence of the interrupt (LAM) until entering the interrupt service routine in the driver (interrupt latency).
- t2: is the time from the occurrence of the interrupt (LAM) until the first statement in the user program after the wait call (cclwt).
- t3: is the typical execution time for a single action routine (e.g. cfsa).
It is observed that:

- The performance of LynxOS and OS-9 on the MVME147 were roughly the same (with LynxOS 30-50% slower than OS-9).

- The RAID8235/LynxOS ran about five times faster than both systems running on the MVME147. In particular, it was noted that the interrupt response was very fast; the fact that the MIPS R3000 has no vectorized interrupt hardware does not seem to significantly influence the interrupt latency.

- MVME167/OS-9 was about four times faster than MVME147/OS-9.

**DATA ACQUISITION PERFORMANCE**

In this section, we describe and discuss the data acquisition performance of the systems listed in table 1. On each we have implemented and run the SPIDER data acquisition system with various configurations of the producer, analyzer and controller tasks.

The producer task is triggered at a high rate (~1 MHz) from CAMAC which assures the presence of a hardware trigger as soon as the producer is ready for the next event (one consequence of the high trigger rate is, however, that the "trigger" kernel semaphore is signalled before the producer suspends itself, with the result that the producer "falls through" the suspend system call and no context switch occurs).

The analysis task was configured to have a sampling rate of either 100% or 0%. In the first case, all events are guaranteed to be analyzed and in the second only as many as allowed by the producer. The analysis task is minimal in the sense that it only contains code for copying an event from the event buffer into a local array but no real monitoring code for analyzing the event data.

The controller task does not normally interfere with the producer and consumer tasks, except when a run option is selected whereby the status of SPIDER is displayed on a screen every second. The resulting I/O activity may then have a non-negligible influence on the data acquisition rate.

Under both LynxOS and OS-9, the producer was given the highest priority, followed by the controller and analyzer, respectively. On all systems the SPIDER as well as the VOS software were compiled using the "gcc" compiler. The measurements were made with the SPIDER tasks being the only active user tasks in the system. However, networking software and basic system tasks were executing as normal.

The timing figures are obtained by counting the number of events flowing through the system in a certain time and cross-checked by examining the VMEbus activity on an event-by-event basis using a VMEbus backplane analyzer. The results are shown in table 7.

**Table 7 Data acquisition performance**

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Number of events/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVME147, OS-9</td>
<td>1410 760 715</td>
</tr>
<tr>
<td>MVME147, LynxOS</td>
<td>1470 870 740</td>
</tr>
<tr>
<td>RAID8235, LynxOS</td>
<td>5550 3125 2860</td>
</tr>
<tr>
<td>MVME167, OS-9</td>
<td>4760 2250 1220</td>
</tr>
</tbody>
</table>

n1: 0% analysis, no status display.  
n2: 100% analysis, no status display.  
n3: 100% analysis, status display.

The figures in the first column correspond to a configuration where the only active SPIDER task is the producer. The event rate is, therefore, mainly determined by the interrupt overhead and semaphore timing of the producer. The figures in the second column correspond to a configuration where the sampling rate of the analyzer is 100%. The producer task is suspended whenever no buffer space is available. There are now two tasks active and the average number of semaphore operations per event increases. The third column corresponds to a configuration where, in addition, the status display code of the controller task is activated with the consequence that the number of context switches is significantly increased due to the terminal activity.

The configuration in the second column, where events are flowing through the buffer, is probably more realistic than the configuration in the first column where events are continuously overwritten in the buffer.

We make the following observations:

- The data acquisition performance of LynxOS and OS-9 on the MVME147 is the same within 15%.

- Comparing the MIPS R3000/RAID8235 with the MC68030/MVME147, both running LynxOS, there is a factor of almost four in performance.

- Comparing the MVME167/OS-9 with the other systems we observe that the event rate in
column one is a factor 3.5 higher than that of the MVME147/OS-9 and 15% lower than for the RAID8235. However, with more tasks running — columns two and three — a certain degradation is observed. We believe this is caused by a higher context switch overhead for the MC68040 due to flush operations on the large data, instruction and translation caches, but we have not yet been able to verify this hypothesis.

**SUMMARY**

We have evaluated the performance of a real-time UNIX system (LynxOS), available for MC68030 and MIPS R3000 based VMEbus CPU modules, in the area of real-time data acquisition. The use of portable test and data acquisition programs has allowed a comparison to be made with the more "traditional" OS-9 operating system, running on CISC processors (MC68030 and MC68040) semaphores and interrupt latencies under LynxOS and OS-9 were studied in more detail. The main conclusions are:

- LynxOS and OS-9 have a similar real-time performance on the same hardware platform (MC68030).

- A RISC (MIPS R3000) based VMEbus processor running LynxOS is a performant real-time system. In comparison with LynxOS implemented on the MVME147 (MC68030) we find a factor of almost four in data acquisition performance, close to the ratio between the Dhriston figures. The R3000/LynxOS system is also more performant than the MC68040 based MVME167, running OS-9.

Due to the fact that the event overheads for the faster systems are only a few hundred microseconds, the total event processing time will, in many applications, be dominated by the time to read the event data. However, since the data collection efficiency of the system depends little on the real-time features of the operating system and the speed of the CPUs, but rather on independent data movers (DMAs), we have not discussed this aspect of data acquisition.

An interesting feature of LynxOS is its POSIX compliance, which we have not exploited. The architecture of SPIDER with several tasks sharing a common data area lends itself to an implementation based on POSIX threads. The use of these lightweight tasks strongly reduces the semaphore and context switch overheads. Preliminary measurements on the timing of the semaphores (mutexes) show an order of magnitude improvement in speed (e.g. 5 µs compared to 47 µs for a fast LynxOS semaphore on MVME147). Using POSIX threads and mutexes could therefore significantly improve the data acquisition performance of LynxOS (POSIX) based data acquisition systems.

**Acknowledgements**

We would like to thank our colleagues J. Blake and D. Kemp for their valuable help with LynxOS and OS-9 system management.

**REFERENCES**


