

## PSOS 2.6.1. Evaluation Executive Summary

The following article is an executive summary of the evaluation report of pSOSystem/x86 2.2.6 from Integrated Systems, Inc.

### INTRODUCTION

During the summer of 1998, Real-Time Consult officially started an RTOS evaluation program. First, Windows NT and the real-time extensions to Windows NT were studied. The evaluation reports for the following products are currently available:

- RTX 4.2 from VenturCom, Inc.
- INtime 1.20 from Radisys Corporation Ltd.
- Hyperkernel 4.3 from Imagination Systems, Inc.
- VxWorks/x86 5.3.1 from WindRiver Systems Inc.
- pSOSystem/x86 2.2.6 from Integrated Systems Inc.
- QNX 4.25 from QNX Software Systems Ltd.

The evaluation reports, as well as comparison reports highlighting the decision critical information are available on our website (<http://shop.realinter.net/>).

This article presents an executive summary of the evaluation report of pSOSystem 2.2.6 from Integrated Systems, Inc.

### ARCHITECTURE

First a note about the nomenclature: pSOSystem is the complete operating system. pSOS+ is the kernel of the operating system. pRISM+ is the integrated development environment used for creating an application that is integrated with the operating system.

pSOS+ is a small kernel for embedded applications. It is a variant of client-server architecture but does not have a message based communication protocol. It uses a software bus to communicate between the different modules. There is a choice of modules that can be built into the system and these are chosen at compile time.

Various components can be built into the system during compile time. These include pHILE+ (file system), pROBE+ (stand-alone debugger), pNA+ (networking protocols), pRPC+ (remote procedure call library) and pREPC+ (ANSI C standard library). These components are shown in Figure 1. Calls to the different APIs are performed using software interrupts (SWIs).

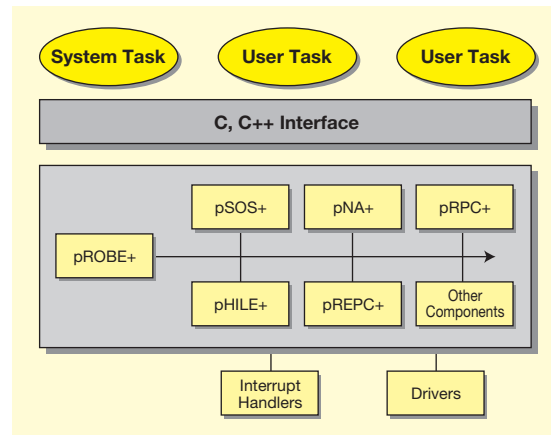


Figure 1. pSOSystem components

pSOS+m is a multiprocessor version of the pSOS+ kernel. It requires one node to be a master and the rest are slaves. The kernel adds features to system calls allowing them to operate across processor boundaries. The nodes can be connected in different ways; shared memory, message-based buses or custom links are a few examples. It is not fault tolerant as there is only one master node and a failure in this node will prevent the whole system from working. It is also only available for certain processors and the x86 is not one of them.

### Tasks

pSOS+ has no notion of processes; it only has what the documentation calls "tasks". These are what we call "threads" all executing within one process. All system objects are shared between all the threads. As all the threads share the same context, thread switching times should be quick as there is less context to change.

Two very serious bugs were found concerning thread handling when using priority-based semaphores and message queues for the x86 platform. The first problem was a bug that we found in their handler's comparison of priorities. Often a thread of lower priority

PSOSYSTEM 2.2.6	
Model	Threads only
Priority levels	256 (0 → 255)
Max. number of tasks	Limited by memory
Scheduling policies	<ul style="list-style-type: none"> <li>• Prioritized FIFO scheduling</li> <li>• Round-robin scheduling</li> </ul>
Number of documented states	4 (Created, Ready, Running, Blocked)

Table 1. pSOSystem 2.2.6 Task handling properties

would be released instead of the highest priority thread ready to execute, causing the time-critical tasks to be delayed indefinitely. We found that all threads of priority less than 128 (0x80) would be considered as if they had a higher priority than threads of priority 128 or more.

We also found that when many threads of the same priority were waiting, the last thread that joined the queue would be the first to be released. Normally this should be operated as a FIFO queue for threads of the same priority. This problem can cause indefinite postponement disrupting the real-time system. Both problems were reported to ISI and confirmed to be bugs.

For more details, the reader is referred to the full evaluation report.

## Memory

pSOSystem has a flat memory space. An MMU can be used for reserving memory but is not necessary. A memory management library is available which provides memory protection. Code, data and stack spaces can be protected by defining the memory protection maps for each task. It is the developer's responsibility, however, to define these maps and is not as simple as it should be.

PSOSYSTEM 2.2.6	
MMU	Optional but not required
Physical page size	4KB
Paging/Swapping	No/No
Virtual memory	No
Memory protection models	<ul style="list-style-type: none"> <li>• No protection</li> <li>• Protection of code, data and stack space with optional memory management library</li> </ul>

Table 2. pSOSystem 2.2.6 Memory Management properties

pSOSystem offers two sets of memory handling functions. The first set refers to Regions and the second to Partitions. Regions are non-fixed block size pools, whereas partitions are fixed-size block partitions allowing for fast allocation algorithms.

## Interrupts

The Interrupt API is not very rich. The functions are provided by the board support package (BSP) and so may vary depending on the target platform. We refer to the PC/AT BSP here. There are system calls for installing a user interrupt handler so that the user routine is called from the ISR. An interrupt handler can be installed and enabled but functions do not exist to disable or remove it. The only way to dynamically change the handler is to change directly the interrupt vector table in assembler.

PSOSYSTEM 2.2.6	
Handling	Nested and Prioritized
Context	Interrupted task
Stack	Kernel stack or Interrupt stack (depending upon target)
Interrupt-to-task communication	Most communication and synchronization objects

Table 3. pSOSystem 2.2.6 Interrupt handling properties

## API RICHNESS

To assess the API richness, we created a list of features for the most common system calls and compared it with the available system calls in pSOSystem 2.2.6. Table 4 gives an overview of all the categories and the score (in percentage points) that was obtained. For a breakdown of the categories into individual features and system calls, the reader is referred to the evaluation report.

This table should not be misconstrued. pSOSystem 2.2.6 has system calls that are not in our list, and are therefore not represented in Table 4.

An average percentage of 44% was obtained. The average percentage does not include any weighting factors, it is simply the average of each category's score. Table 4 shows that the pSOSystem API is missing some basic objects like mutexes.

## PERFORMANCE TESTS

### Interrupt latencies

For this test, we measured two latencies:

- *Interrupt Latency (task to interrupt handler)*: The time elapsed between the execution of the last instruction of the interrupted thread and the first instruction in the interrupt handler.

# RTOS EVALUATIONS

MECHANISM	RICHNESS
Thread Management	75%
Clock	43%
Interval Timer	50%
Fixed block size memory partition	73%
Non-fixed block size memory pool	73%
Interrupt Handling	25%
Counting Semaphore	67%
Binary Semaphore	0%
Mutex	0%
Conditional Variable	0%
Event Flags	88%
POSIX Signals	63%
Message Queue	60%
Mailbox	0%
<b>AVERAGE PERCENTAGE</b>	<b>44%</b>

Table 4. API Richness

- **Interrupt Dispatch Latency (interrupt handler to task):** The time needed to go from the last instruction in the interrupt handler to the next task scheduled to run.

Figure 2 and Figure 3 display the minimum, maximum and average values for both interrupt latency and interrupt dispatch latency. The results for QNX 4.25 from QNX Software Systems, Ltd are included for the sake of comparison.

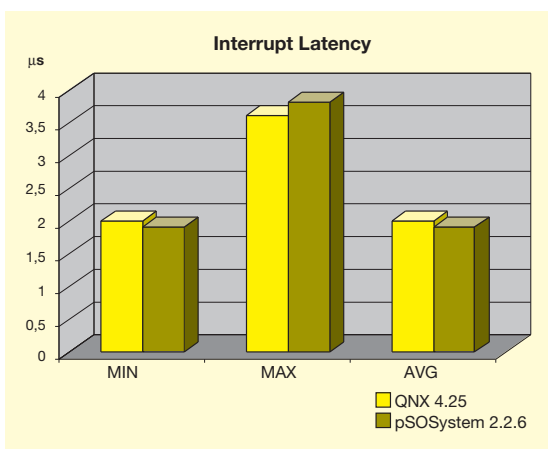


Figure 2. Interrupt Latency – pSOSystem 2.2.6

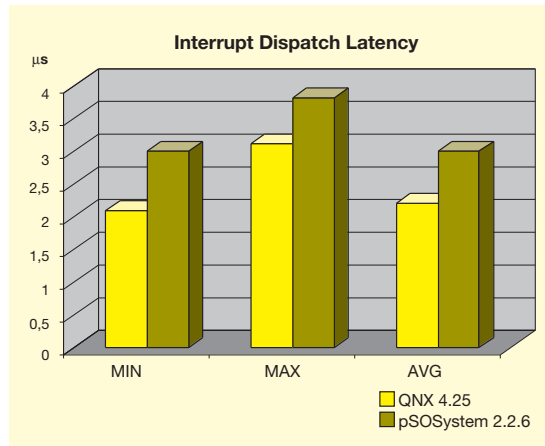


Figure 3. Interrupt Dispatch Latency – pSOSystem 2.2.6

## No Priority inheritance

PSOSystem 2.2.6 does not provide a mutex; there is no priority inheritance mechanism to avoid priority inversion. If a thread needs to manipulate a shared resource, there is no other option than to disable preemption. This mechanism is not as efficient as a mutex, since no other thread will be able to run as long as preemption is disabled, not even time-critical high priority threads.

## TOOLS & DEVELOPMENT METHOD

Development of a pSOSystem application is done using pRISM+, a complete integrated development environment available for UNIX systems, Windows 95/98 and Windows NT. The Windows NT version of pRISM+ 1.2.3 was tested here. pRISM+ uses CORBA to integrate the different modules together which means that additional subsystems could be added to the development environment.

The editor is called SNIFF+ and has obviously been ported from a UNIX operating system to Windows. Many of the standard keyboard shortcuts perform a different action to that expected. It also requires some fiddling in order to edit two source files at the same time. It is usable but counter-intuitive for a Windows program.

A debugger called pROBE+ is available that can be built into the target application where one can set breakpoints and watch the state of system resources. This debugger runs externally from the kernel and therefore does not require the kernel to be running in order to debug. This allows the developer to debug startup code and ISRs for example. The system can also be debugged by an external debugger that is attached via a serial cable or a network connection.

Source code control systems are provided and are integrated into the development environment. Compilers are supplied for various targets from ISI's sister companies. An API simulator is available for Unix hosts. An instruction set simulator is available for Unix hosts that simulates target platforms such as PPC, 68K and ARM.

## DOCUMENTATION & SUPPORT

The main part of the documentation supplied with pSOSystem comes in Adobe Acrobat PDF format and HTML format on a CD-ROM. In general, however, there is not enough cross-referencing between the different sections of a manual or between the manuals making it more difficult to find related information.

Other sources for finding solutions to problems are ISI's website and the comp.os.psos newsgroup. There are very few current postings in the newsgroup however. ISI's website has an online technical support section, but at the time of the evaluation (March-April '99), this appeared to be a new development as there were less than 400 questions answered. After the evaluation was finished, ISI informed us of their plans to revamp the website in order to enhance support for developers.

## CONCLUSION

pSOSystem/x86 2.2.6 exhibited fast and predictable behavior during our tests.

Several tests revealed a few serious bugs however. Especially the scheduling related bugs discovered in the handling of priority based semaphores and message queues in pSOS+/x86 could have serious consequences. We also found a bug in the initialization procedure of the x86 PC/AT BSP supplied with pSOSystem that prevented it from booting on certain motherboards.

ISI confirmed these bugs and informed us that they would be fixed in the next version of pSOSystem (version 2.5). The reader should also bear in mind that only the x86 release of pSOSystem was evaluated, and that the obtained results do not apply to any other platform supported by ISI.

PSOSystem/x86 2.2.6 does not have a mutex. When a thread needs exclusive access to a shared resource, preemption needs to be disabled.

The pRISM+ development environment comes with an extensive set of tools and allows your favorite tool (e.g. from a third party vendor) to be plugged into the environment. The Windows version of pRISM+ 1.2.3 used for this evaluation however does not conform to Windows standards.

The documentation is not as clear as one would have hoped. The main way for us to obtain help during this evaluation was via direct answers from ISI. We had to send many emails in order to get a working system and each new test needed more communication.

## OTHER PUBLICATIONS AND SERVICES

As mentioned at the outset, evaluation reports for Windows NT 4.0, RTX 4.2, Hyperkernel 4.3 and INtime 1.20 have been for sale on our website since the beginning of 1999.

Real-Time Consult has also evaluated VxWorks/x86 5.3.1, QNX 4.25 and pSOSystem/x86 2.2.6. Evaluation reports for these products have been available since April 20, 1999.

The evaluation reports are intended for anybody who is in one way or another involved with dedicated systems technology. This obviously includes the system design engineers and application developers, who need to have a detailed understanding of how the product behaves in a real-time environment, but the audience also includes managers and project leaders who need to make strategic decisions like which RTOS to use, and how it will affect the overall execution of the project.

Finally, Real-Time Consult also performs feasibility studies and product validations on customer demand. Please contact our offices for additional information. ■

---

*Dr. Martin Timmerman has a degree in Telecommunications Engineering from the Royal Military Academy (RMA) Brussels and received a Doctorate in Applied Science from the Gent State University (1982) in Belgium. In 1983 he transferred to Computer Engineering and set up the System Development Centre (SDC) at RMA. He gives general courses on Computer Platforms and more specific courses on System Development Methodologies. He is a consultant to the Joint Staff of the Belgian Armed Forces in areas concerning Information System Methodologies and CASE tools and he is the Belgian representative in some NATO technical commissions. Outside the RMA, Martin is known for his audits, reviews and seminars, and for his two companies Real-Time Consult and R.T.U.S.I., where he makes use of his considerable knowledge of the Real-Time world. Real-Time Consult is the publishing house responsible for Real-Time Magazine, an International magazine about Real-Time system development. Real-Time User's Support International (R.T.U.S.I.) provides hardware and software support services and is involved in project engineering for real-time systems.*

*Bart Van Beneden has been with Real-Time Consult since 1998 where he is involved in the RTOS evaluation program of Real-Time Magazine as a project manager. He received his degree in computer science at the Free University of Brussels. Before joining Real-Time Consult, he designed multi-media applications with LaserMedia Inc.*