

**RadiSys**<sup>®</sup>  
THE POWER OF WE

# WHITEPAPER

## PROFITS ARE MADE IN REAL TIME

VERSION 1.0 | OCTOBER 2006

### CONTENTS

➤ EXECUTIVE SUMMARY .....	2
➤ PROFITS ARE MADE IN REAL TIME.....	2
➤ SO MANY CHOICES, SO LITTLE TIME .....	3
➤ COMMERCIAL RTOS ON THE RISE.....	3
➤ LINUX FEVER ABATING.....	3
➤ IMPORTANT CRITERIA .....	4
➤ PROCESSOR SUPPORT .....	4
➤ SOFTWARE TOOLS.....	4
➤ OVERALL COST .....	4
➤ MEMORY REQUIREMENTS .....	5
➤ SIMPLICITY .....	5
➤ MIDDLEWARE .....	5
➤ CONCLUSION .....	6

## EXECUTIVE SUMMARY

➔ Shout the phrase “operating system” in a crowded theater and most people will think of Microsoft or Windows. In reality, the famous Windows software appears on barely 2% of all the computers in the world, even though most people would guess a far bigger number. Even Linux and MacOS command just small segments of the overall market.

Around 9 billion new microprocessor chips are sold every year, and almost all (98%) of them go into embedded systems, not PCs. Those billions of chips obviously need software to run, and often an operating system as well. For many years (decades, in fact), it was customary for programmers to create all of their own software, including a proprietary, in-house operating system or custom real-time kernel. But that age is passing.

Statistically, more than half (51%) of embedded projects with an operating system use a commercial one, compared to just 21% using a custom, proprietary, or internally developed one. That’s up from last year, when 44% used a commercial operating system. In fact, the commercial market share has increased every year as more and more designers (and their managers) choose a commercial operating system instead of “rolling their own.” Only 17% of developers expect to still be using an in-house operating system next year. The trend here is clear and unambiguous.

Why the sea change in embedded operating systems? It’s because writing – and maintaining – an operating system is simply not most companies’ core competence. They’d rather be designing and shipping products than developing and debugging the underlying software. That’s a task better left to experts, while the in-house talent concentrates on adding value. Just as engineers use commercial chips, programmers now use commercial software components.

A commercial RTOS can be inexpensive, small, fast, and easily modified for the specific task at hand. The result is a better product, developed quicker and more reliably. Just don’t go shouting about the benefits in a crowded theater.

## PROFITS ARE MADE IN REAL TIME

Embedded systems aren’t just gears and grease anymore. From the early days of industrial controllers running machinery, today’s embedded systems are in the home, office, cars, toys, and transportation. The average new car includes more than a dozen microprocessors; the Mercedes Benz S-class has more than 100. The typical middle-class American home includes 40–50 microprocessor-based devices, not counting computers in the den or cars in the garage. Nearly anything that takes batteries or plugs into the wall includes at least one microprocessor-based system.

A whopping 9 billion new microprocessor chips go out the door each year, and the rate has been growing at double digits for decades. That’s more than one new chip for every man, woman, and child on the planet – just for this year. Next year another 9–10 billion chips will find their way into new systems, with more billions the year after that and the year after that... Even low-cost microprocessors used in toys have more processing horsepower than a NASA moon lander.

Where there are processors there’s software, and lots of it. Simple 8-bit thermostats may not require a multitasking operating system but just about everything else does. A recent survey showed that only 28% of new embedded systems don’t use an operating system at all, a fraction that’s been dwindling steadily for years. It’s no wonder; new systems have to deal with a combination of complex user interfaces, streaming media, network interfaces, real-time constraints, encryption, decoding, peripheral I/O, power management, inter-task communication, and a dozen other complicated issues.

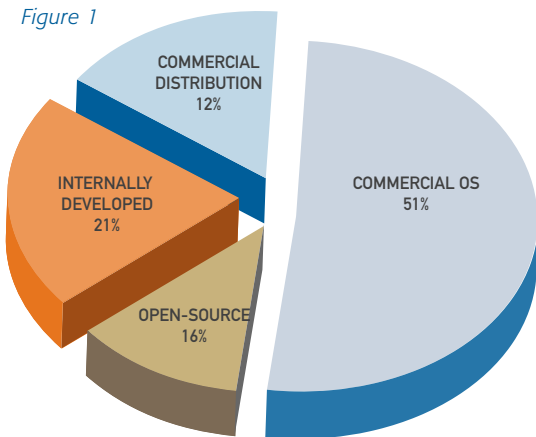
Most programmers (and their managers) simply aren’t experienced in this kind of work. Their expertise lies in some other market- or application-specific niche. They’re experts at media decoding, or robot kinematics, or machine vision, or satellite downlinks, or network protocols. They’re not experts in juggling all those things in one box. Nor should they be. Just as no engineer would dream of creating his own microprocessor when so many commercial alternatives are available, no programmer would create all that software from scratch when better options are right here. “Stick to what you know,” say the management experts. “Add value, don’t reinvent the wheel,” say the executives. “Help, I need more time,” reply the poor engineers. ↴

**SO MANY CHOICES, SO LITTLE TIME**

Operating systems are a necessity in any non-trivial embedded system, as we've seen. But what operating system? Just as with microprocessors, there are a lot of choices. Some of the most consumer-friendly brand names (Windows, MacOS, Unix, etc.) are out of the question. They're well-known but designed for desktop computers, not embedded systems. Would you want Windows controlling your antilock brakes?

So we enter the world of the real-time operating system (RTOS), designed specifically for embedded, real-time control of complex tasks. It's called "real time" because an RTOS is aware of actual time and take punctuality and dependability very seriously. Whereas a PC can occasionally garble an e-mail message and nobody's surprised, an antilock brake system or high-rise elevator controller can't fail – ever. They can't even be slightly late. They can't miss deadlines, can't garble messages, can't misunderstand commands, can't succumb to viruses, and can't take time out to defragment the hard drive. An RTOS has to be so completely reliable that it becomes invisible.

Figure 1



**COMMERCIAL RTOS ON THE RISE**

There was a time when programmers would create their own real-time operating systems (often just schedulers or kernels, really). It was a badge of honor to create one's own RTOS, a bit like rolling your own cigarettes or repairing your own transmission. It's likely that people used to make their own buggy whips, too, but that's also become a waste of time. Over time the home-grown operating systems have faded away as their developers retire and their products reach the end of their useful life.

Recent survey data shows that only 21% of embedded developers still use an internally developed, proprietary, or in-house operating system. Even fewer, just 19%, anticipate doing so on their next project. That's a down from just a year ago and another sign of the slow, steady decline in evidence for several years. Clearly, home-grown operating systems are on the way out.

Conversely, commercial operating systems are rapidly taking up the slack. Already 51% of developers use a commercial RTOS, up from 44% just a year ago. That's a big 16% leap in commercial RTOS usage in roughly one product cycle. Again, this is a trend that's been in place for years and shows no sign of tapering off.

**LINUX FEVER ABATING**

The chart also shows that a portion of developers are using an open-source operating system, generally some version of Linux. As a variation of Unix, Linux was never designed to be used as an embedded RTOS, but that hasn't stopped many engineers from trying. Despite the impression from the general media, however, the use of Linux in embedded systems seems to be diminishing.

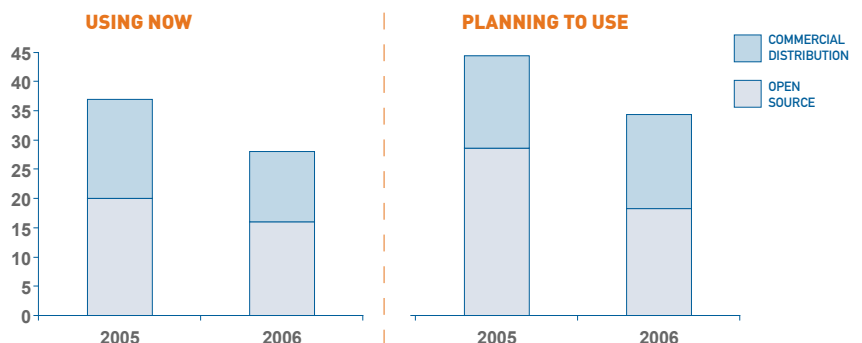
Although 28% of developers surveyed said they were using an open-source operating system currently, that's down from the 37% response rate of a year ago. Moreover, when those developers were asked what they expected to be using next year, the response dropped again. In 2005 43% expressed their intentions to use an open-source operating system but that number fell to just 36% one year later. Finally, of the 43% who expected to use an open-source OS in 2005, only 28% actually did so in 2006. Last year's respondents seemed more optimistic about Linux, while a dose of reality has made this year's open-source plans much more subdued.

Why the disillusionment? A number of factors are contributing to the rapid retreat from Linux. Foremost among them is incompatibility. Programmers said Linux was incompatible with all of their existing software and hardware, and that they'd have to start over with new applications, new drivers, new everything. For the majority of developers, there's just no compelling reason to expend the effort.

Poor performance and lack of real-time capability were also cited as major concerns, followed by support worries (Linux, by nature, has no commercial support), and high memory requirements.

Linux's ambiguous legal situation also scared off many potential users. Developers aren't certain who owns Linux or what their exposure might be after the operating system is permanently embedded into a finished product. Few can imagine a worse outcome than having to recall and redesign a shipping system because of a far-off legal ruling. Many just weren't willing to take the chance.

**EMBEDDED LINUX**



Interestingly, cost was also cited as a barrier to open-source adoption. Even open-source software isn't free; somebody has to develop, customize, debug, and test it. You can either pay for it now or pay for it later.

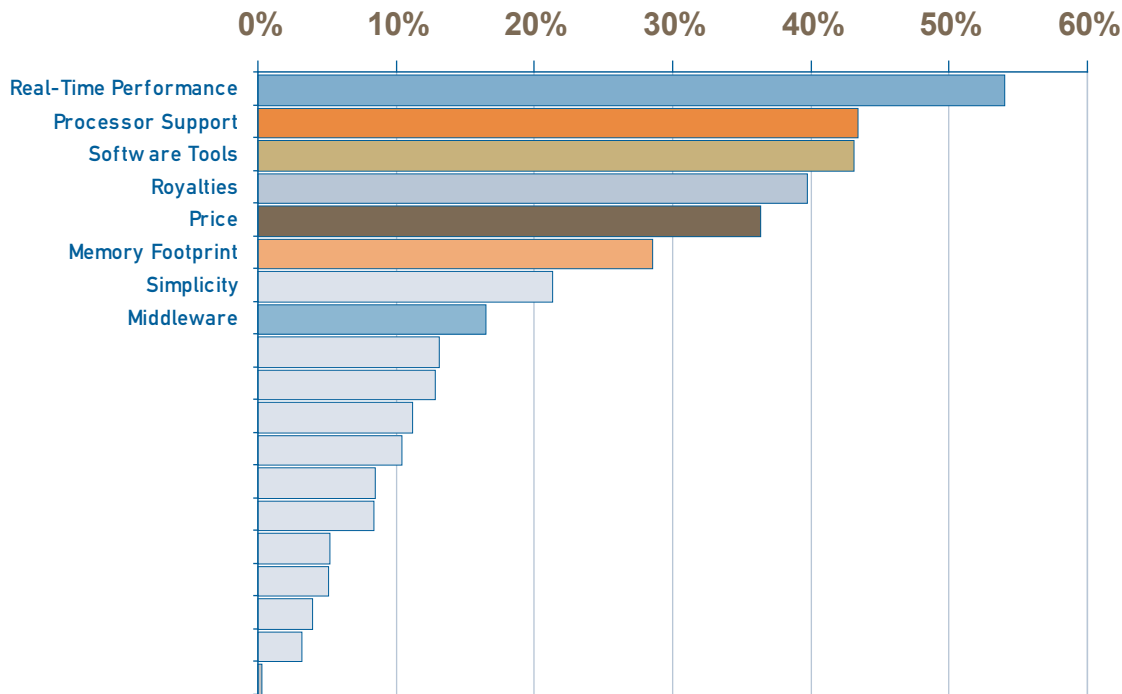


Figure 4: RTOS Evaluation Criteria

**IMPORTANT CRITERIA**

So we've seen that programmers, engineers, and project managers are concerned about cost, support, memory requirements, and performance. How do these factors rank in terms of importance? The chart shows that real-time performance is the biggest hot-button issue for most development teams. This is where a "hard" real-time operating system like OS-9 from RadiSys really shines.

A hard real-time system is one that must meet definite deadlines – it absolutely, positively, has to be ready on time. Examples are antilock brakes, elevators in a high-rise, disk drive controllers, industrial robotics or motion-control equipment, and even network infrastructure – all systems where a slight delay or miscue means failure. Being late is not okay, and "same day service" doesn't come close.

With more than 25 years of actual use behind it, OS-9 has proven itself in countless hard real-time systems. Systems running OS-9 can respond in under a microsecond – barely a few cycles for most microprocessors. Its task management and priority-based scheduling mean that it's always ahead of the game, swapping software tasks in and out as required.

**PROCESSOR SUPPORT**

Right behind performance in the list of evaluation criteria is processor compatibility. Programmers can't use an operating system that doesn't run on their processor. Compatibility is a pass/fail criterion for most developers. Either you've got it or you don't, and many RTOS alternatives don't support very many processors.

OS-9, in contrast, runs on every one of today's top six processor architectures. Whether it's RISC or CISC, standard or compressed instruction set, licensed IP core or packaged processor, OS-9 supports them all. From the venerable Intel x86 and Freescale (née Motorola) 68K families to the newest ARM, StrongARM, and XScale chips, all the way through PowerPC, MIPS, and SuperH designs, RadiSys offers factory support for dozens of chips from the major vendors and top architectures.

**SOFTWARE TOOLS**

Coming in a close third in importance is software tools. After all, even the best embedded system will never get out the door if the engineers and programmers can't get it to work. The fastest processor or the most efficient RTOS are useless without appropriate tools. This statistic closely matches similar survey data on microprocessors: there, too, engineers voted for development tools as the most important criterion.

OS-9 comes as part of a package that includes everything the development team needs to get started – and more important, to finish. Starting with the award-winning



CodeWright programmers' editor and the optimizing C/C++ compiler, through graphical debug tools, all tied together with a customizable IDE (integrated development environment), developers have what they want and need. Because the C/C++ compiler and debugger understand the nuances of the OS-9 operating system they provide unparalleled code visibility.

Want more? HawkEye is a graphical visualization tool that displays process interactions in a way that makes the overall organization clear. System calls, context switches, interrupts, forks, and more are all laid out visually by HawkEye. With its flexible triggering, programmers can see events before, during, or after the trigger and it allows completely user-defined triggers.

### OVERALL COST

Purchase price and royalty rates rank fairly high in developers' minds, showing that it's not all just about technology. Successful development teams keep their eye on the bottom line, too. Cost is a tricky thing to quantify. As the cool response to open-source showed, cost isn't just about up-front fees. There's a very real cost to after-sales support. The biggest cost of all is a product delay.

With OS-9, the cost of the operating system itself is minimal. It varies based on the features the project actually uses, so customers are never "over buying" software or technology they don't need or don't want. Conversely, those features can be added later (as we'll see) so customers are never locked out on an upgrade, either.

On the support front, OS-9 is second to none. RadiSys provides around-the-clock support in several countries and numerous languages. In addition to the traditional telephone and e-mail support channels, programmers can even connect live with RTOS support experts who can see and modify everything on the user's screen, exchange files, and take control of the desktop. In a new development, OS-9 can now collect and transmit all the relevant debug information directly to RadiSys support engineers so they have a complete picture of the situation. It's a unique way in which OS-9 works to maximize uptime.

### MEMORY REQUIREMENTS

It's not unusual for embedded designers to spend more on memory than they do on the microprocessor. No wonder, then, that memory "footprint" ranked highly on their list of concerns. No embedded system has unlimited resources; most are severely cost-constrained. It's an old programmers' axiom that code expands to fill the memory available.

At the same time, there's always demand for new and better features. Network interfaces have to support new protocols they didn't have before, or add encryption and digital rights management (DRM) where there was none. Assuming these features are even available in the operating system, how can they be included?

Conversely, if a system doesn't require all the latest features, why should it be burdened with them? Why can't an operating system have all the features you need (even if you don't know it yet) without any of the features you don't? That's exactly what OS-9 delivers.

OS-9 is completely modular, so that absolutely everything apart from the kernel itself is an optional standalone software module. Networking, memory management, user interface, peripheral drivers – everything is optional. Whatever the final system turns out to be, it will include only those portions of OS-9 that it actually uses. No dead code, no bloated libraries, no linked-in baggage. Every section of OS-9 is optional, selectable, and updatable.

Better still, OS-9 can load and unload these software modules at run-time. This is worlds away from traditional build-time linking, where a master software image is compiled and linked into the kernel at the factory. Instead, OS-9 can receive new software modules while it's running and execute them immediately, all without rebooting. It even works in reverse: unused software functions can be unloaded and flushed from the system, again without rebooting.

This flexibility gives programmers unprecedented freedom. Code components can change in response to system requirements long after the system has shipped. Applications, drivers, protocol stacks, and more can all come and go as needed, like on a PC. You can even download code patches or updates onto a running system, all without rebooting or reconfiguring the operating system.

A final note on memory: OS-9 allows all applications to be ROM'ed. In other words, everything from the RTOS kernel to application code and drivers can execute directly from ROM, EPROM, or flash memory. No RAM is required for execution, only for normal data storage, stacks, and workspace.

### SIMPLICITY

The whole concept of simplicity would seem to be at odds with the complex embedded systems of today. But that doesn't stop engineers from wishing, and they wish their RTOS choices were simpler. Who wouldn't want to simply snap together a new product? The simplest produce is one that's already out the door, not stuck in the engineering lab with wires and probes dangling out of it.

Over the years OS-9 has grown simpler and more modular instead of more complex and bloated. Its modular architecture, mentioned above, is a big part of that simplicity. But there's more to it than that. OS9's application programming interfaces (APIs) are also simple, regular, and orthogonal meaning they all work the same way. Even if the development team moves from one microprocessor to another (say, from 68K to ARM or Pentium) the APIs stay the same. It's surprisingly easy to switch the hardware out from under the system and keep the software intact.

OS-9 also scales up and down for big systems or small. Programmers can use the same operating system across multiple projects, even if some are low-cost systems and





others are huge team projects. OS-9's modular feature set offers "snap together" simplicity that grows (or shrinks) to suit any size project – all without relearning the operating system or its interfaces.

Finally, OS-9 adheres to POSIX conventions in a lot of important areas, following one of the few software standards available in the embedded world. That means software can more easily be transferred (ported) from other systems into OS-9 with a minimum of fuss. It also means it's easier to hire programmers with experience with POSIX interfaces.

### MIDDLEWARE

An operating system's no good if there isn't any software to run on it. Adding features and functions – whether it's custom code or commercial programs from a third party – needs to be easy and profitable. Following the model of Web browser "plug-ins," OS-9 makes it easy for customers and third parties to plug in their own extensions to the operating system.

One popular class of plug-in is networking capability, which has shot up the rankings in importance as more developers work on network-enabled or communications-enabled embedded systems. The need for networking has grown just as rapidly as programmers' distaste for writing the necessary protocol stacks. Writing, debugging, and testing network code is dreary and uninteresting work. It's tightly defined and standards-driven so there's little room for creativity – but lots of room for error. Networking protocols are another pass/fail item: your network stack either works or it doesn't, and most managers would rather buy code that works than risk developing code that doesn't.

What programmers want is the same things that every VCR or DVD owner wants: a big selection of software to "play" on the device. With OS-9 that software comes from RadiSys and from an array of independent third-party vendors. As with everything else about OS-9, it's all plug-in modular and ready to go.

Communications interfaces are in the bag. For example, OS-9 has ready-made support for USB, TCP/IP (including the complicated and important IPv6), and more. OS-9's modularity even extends to its networking stack. Protocols can be switched in and out during a live session. New protocols can even be invented and added, all while the system's running. Of course, conventional IPv4 and IPv6 support is already provided.

Sometimes the best features are the ones you didn't think of. Take flash memory. Flash is just like any other memory, right? No indeed – flash requires special drivers, and those drivers change for each type and manufacturer of flash chips. As popular as flash memory is, it's tricky to use correctly. Flash chips have a limited life span, too, and they wear out rapidly if they're not managed properly, something many programmers forget.

But OS-9's TrueFFS flash-file system handles all that and more automatically. Wear-leveling is built in, as is support for all makes and types of flash memory chips, including Intel's newest StrataFlash devices. With OS-9 and TrueFFS, programmers really can treat flash as just another type of memory.

It used to be that a "Web server" meant a big computer with lots of disk drives and a hundred Internet connections. Some still are, but even tiny Web servers are now embedded into simple appliances. An embedded Web server is a great way to provide remote control, configuration, and diagnostics from anywhere in the world. Everything from lawn sprinklers to washing machines have their own Web server embedded right in. OS-9's RomPager software provides everything you need to make that happen. For larger or more traditional Web servers, OS-9 also supports Apache, the phenomenally popular server software used all around the world.

MAUI and JamaicaVM take programmers to places they've never been. MAUI (multimedia application user interface) is a pre-made set of graphical user interface (GUI) functions. Now you can add a full windowing interface to your embedded system without starting from scratch. JamaicaVM runs real-time Java code under OS-9, opening up new horizons for third-party software. It comes with its own static compiler (as opposed to strictly run-time interpretation) so all pre-compiled code becomes predictable and reliable, fixing the two biggest drawbacks with conventional Java bytecode. And of course, all code can be ROM'ed just as always.

### AND IN THE END...



Engineers love their jobs. Programmers love their jobs. Even project managers love their jobs, but none of them wants to slave over the same project forever. Eventually, you have to – you want to – ship the product and move on to the next one. And the faster the current project is finished the sooner everyone gets paid. Yes, developing new products is fun, but developing more new products in less time is even more fun. And more profitable.

Every development team, large or small, has some of special skills that sets it apart. Whether they're experts in network connectivity or automotive electronics, they're different from everyone else. That's their forte, their métier, their value-add, their core competence. To remain the best at what they do, they stick to what they know and leave the rest to outside experts. At RadiSys, they're experts in real-time operating systems. For 25 years they've developed, refined, tweaked, updated, and used OS-9 on every popular microprocessor architecture and every imaginable product and market. They're the best at what they do. Let OS-9 show you how good experts can be. ///

**RadiSys**  
THE POWER OF WE

#### World Headquarters

5445 NE Dawson Creek Dr  
Hillsboro, OR 97124 USA  
Phone: 503-615-1100  
Fax: 503-615-1121  
Toll-Free: 800-950-0044  
www.radisys.com  
info@radisys.com

©2006 RadiSys Corporation. RadiSys is a registered trademark of RadiSys Corporation. Conveda, Microware and OS-9 are registered trademarks of RadiSys Corporation. Promentum, and Proclerant are trademarks of RadiSys Corporation. \*All other trademarks are the properties of their respective owners.  
07-1333-01 1006

**RadiSys**<sup>®</sup>  
THE POWER OF WE