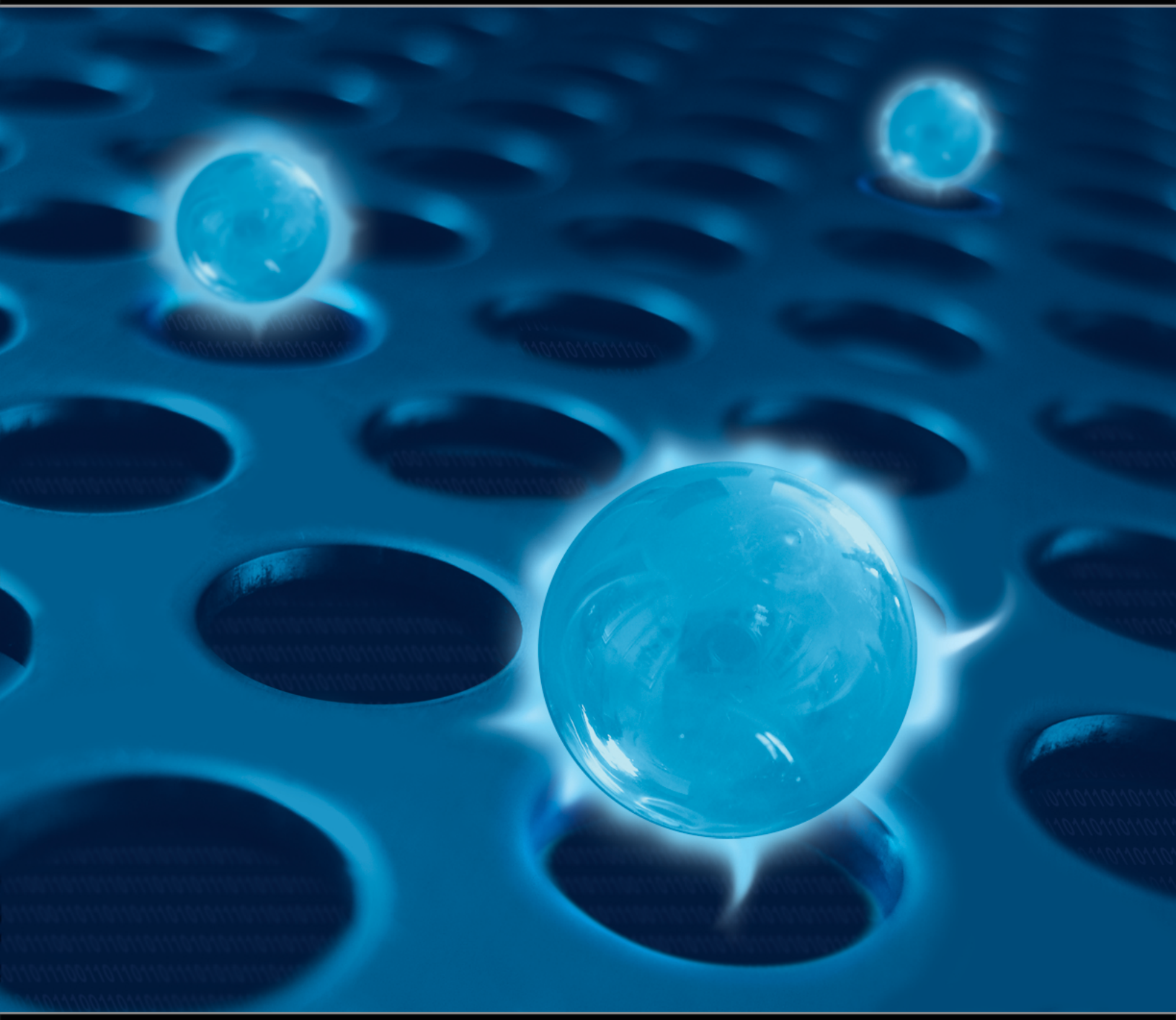
SELECTING AN EMBEDDED RTOS



Prepared by:
eg3.com
Jason McDonald, Senior Editor

eg3.com
tel : 510.713.2150
email : info@eg3.com
web : http//www.eg3.com

Insiders' Guide Embedded RTOS: Contents - January 1, 2008

Contents

Many, if not all, embedded projects need a "real-time operating system" (RTOS), whether it is a "roll your own," a commercial RTOS such as VxWorks, µC/OS-II, RTXC, ThreadX, or Nucleus, or a free/open source RTOS such as Linux or eCos. This research guide details the issues involved in selecting an embedded RTOS and also presents survey results from real RTOS users. It indexes company profiles on all known commercial or non-commercial providers of commercial, "Free," Linux, or other RTOS-related products or services.

- Executive Summary, Copyright
- Introduction
- The RTOS Buy vs. Build Decision
- The RTOS Ecosystem (Commercial)
- The RTOS Ecosystem (Free and Linux)
- Embedded RTOS Survey Results
- Interviews: General RTOS Issues
- Interviews: The RTOS / Tools Connection
- Interviews: Open Source - Linux and Beyond
- Interviews: Virtualization, Multicore, Hypervisors
- Interviews: Beyond the RTOS
- Appendices
  - A. RTOS-related New Products (Last Six Months)
  - B. Vendor Profiles: Commercial RTOSes, Broad
  - C. Vendor Profiles: Commercial RTOSes, Focused
  - D. Vendor Profiles: Linux / Embedded Linux
  - E. Vendor Profiles: Open Source (Non-Linux) RTOSes & RTOS tools
  - F. Vendor Profiles: Misc. RTOS-related Tools
  - G. Non-Commercial Embedded Linux Distributions
  - H. Non-Commercial "Free" RTOSes
  - I. Key RTOS Websites, Conferences, Seminars
  - J. RTOS Books

EXECUTIVE SUMMARY

Almost every embedded project can benefit from a "real-time operating system" (RTOS), whether it is a "roll your own," a commercial RTOS such as *VxWorks*, *Neutrino*, *ThreadX* or *Nucleus*, or a free/open source RTOS such as *Linux* or *eCos*. Indeed, your choice of RTOS can be a critical factor towards project success, or failure. This *Insiders' Guide* discusses the promises and pitfalls of selecting an RTOS, as well as presents company profiles of over 100 commercial RTOS companies, many variants of Embedded *Linux*, and over 45 free/university RTOSes.

The guide first overviews the RTOS ecosystem. What is the lay of the land? Who are the key players, and how does one find out about them? Is *building* an RTOS better than *buying* an RTOS? The guide outlines the major RTOS types - build your own, commercial RTOSes, niche RTOSes, and free RTOSes / embedded Linux. Secondly, the guide analyzes survey data from over 300 design engineers and programmers (part of eg3.com's e-clips 37,000 subscriber community) as to their *before* and *after* RTOS perspectives. Finally, we offer a series of "email interviews" and links to white papers by major RTOS vendors covering issues such as "*Windows XP Embedded*," "Understanding Licensing and Contract Terms," or "Technical Criteria for Selecting an RTOS."

COPYRIGHT, DISCLAIMER, LICENSE AGREEMENT

INSIDERS' GUIDE EMBEDDED RTOS: INTRODUCTION

Introduction

Selecting an Embedded Real-Time Operating System (RTOS) is much like getting married. It can be a good (great!) choice or a bad (disasterous!) one. This guide polls the "RTOS Insiders," is broadly concerned with all types of RTOSes (hard, soft, and in-between), and is written primarily for engineers designing commercial applications that might benefit from an RTOS.

- EG3.COM INSIDERS' GUIDE: PHILOSOPHY
- SELECTING AN RTOS / GETTING MARRIED
- DEFINITIONS
- WHO THIS GUIDE IS FOR
- ABOUT THE AUTHOR AND **eg3.com**

SELECTING AN RTOS/ GETTING MARRIED

Selecting a Real-Time Operating System (RTOS) for an embedded application is very much like getting married. First, there is the **courtship** phase of searching available alternatives, investigating each possible match (dates are like demos, so to speak), of researching partners or parents (partner software is a bit like the relatives or clan of your possible spouse). Second, there is the **wedding** at which vows are made, contracts signed, and the possibilities seem boundless – the idea that this RTOS is going to be perfect, with wonderful real-time performance, a small footprint, and wealth of features and application software that will make your embedded design both easy-to-implement and successful in the marketplace.

And third, there is the **marriage**. Marriage, it is said, is a lot of hard work. Marriage is when you find out that your spouse is not so perfect, that he or she has flaws. Marriage is when you try changing your spouse, only to learn that sometimes it is easier to change yourself. In terms of embedded software, this can be the phase of bugs, of disappointments, of long calls to tech support, of changing hardware or software to adapt to the RTOS. And unfortunately, it is occasionally the time in which you might wonder whether your choice was a bad one. You may be "locked in" to a particular RTOS (or a particular spouse), only to learn that the costs of exiting (a.k.a., divorce) are too high. So you are stuck, and you must "make the best of it."

Or, on the contrary, marriage can be a happy phase – a phase in which you find out that your spouse (or RTOS) actually works quite well, and even brings surprising new skills and insights.

- Will you make a good choice, or a bad choice?

Wouldn't it have been wonderful to have fully considered the **full ecosystem** of available RTOS choices *before* you made the choice? Wouldn't it have been wonderful to have foreseen the pros and cons, strengths and weaknesses? And to continue the analogy, wouldn't it have been useful to realize that each man or woman is unique, and that what makes a good partner for one person (or one embedded design) is not necessarily what makes a good partner for another? Wouldn't you, and your boss, sleep better knowing that you made the **best** choice given your design needs and constraints?

- This *Insiders' Guide* seeks to do just that: to help you identify the RTOS or set of RTOSes that will best match your design parameters.

EG3.COM INSIDERS' GUIDE: PHILOSOPHY

Who "best" knows the RTOS ecosystem? Who then are the "insiders" in *Real-time Operating Systems* or *RTOS*? What is the best "research strategy" to investigating your RTOS choice? Who *really* knows the technology in and out? Despite what the journalists at *EE Times*, *Dedicated Systems*, or *Embedded Systems Design* might say, the smartest people on RTOSes are the people who are already creating and/or working with RTOSes. Or perhaps the vendors of RTOSes and related tools or services. With all due respect to our fellow journalists, the real RTOS experts are not media guideers! As Woody Allen said, "Those who can't do, teach. Those who can't teach, teach gym." (http://www.humorsphere.com/movie_quotes/woody-allen.htm, 17 July 2007). Those who understand RTOSes, do RTOSes. Those of us who don't, consolidate information and write guides.

Our goal, therefore, is not to write the definitive guide to RTOSes but rather to help you (the engineer, RTOS designer, or programmer) to understand the high level issues

surrounding RTOSes and to identify the best jumping off places in the RTOS community for you to obtain useful information.

As for RTOS "insiders," there are two basic expert groups:

Insider Group #1: the engineer / programmer users of RTOSes, i.e. your peers. Those who have gone before you and already used an RTOS in their design, a design tool, or a board. Included are also those currently considering RTOSes, i.e. at the first steps in their design cycles wrestling with many of the issues common to RTOS newcomers. We shall call the first subgroup "experts" and the second subgroup, "newbies."

Insider Group #2: the vendors of RTOSes, whether commercial or Linux / Open Source. Companies like **Wind River, MontaVista, Quadros, Micrium, Express Logic, Mentor Graphics** or others.

- We use surveys to tap the knowledge of the first group, and interviews to tap the knowledge of the second. In addition, we provide a brief introductory sketch of the RTOS ecosystem with lots of links to more information and cool sites on the Web. We hope you find this guide a useful "starting point" on your personal RTOS educational journey. Comments? Questions? Please contact us at info@eg3.com or Tel. 510-713-2150.

DEFINITIONS

Real-time and embedded systems are plagued by "definition wars." It is as if a bunch of guys got together at a bar and argued over who was a better wife, Laura Bush, Hillary Clinton, or their own particular wife. What they fail to realize is that there is no single "best" choice, one size does not fit all even though for one person there may be better or worse matches.

That said, we are therefore not strict with definitions in this guide. We define "RTOS" loosely, avoiding the common polemic about what is true "hard" real-time (i.e., applications that require responses within a definable time or prioritization) vs. "soft" real-time (i.e., applications that can tolerate responses in a slower time frame and/or more ambiguous prioritization). Instead, we focus on all three broad groups of operating systems in embedded systems:

1. **Hard** real-time operating systems – i.e., operating systems that (claim to) have definable response times and/or definable response priorities;
2. **Soft** real-time operating systems – i.e., operating systems that (claim to) tolerate longer response times and/or more ambiguous response priorities; and
3. **General Purpose** operating system (GPOSes) like *Linux* or *Microsoft Windows XP* that are not really "real-time" at all, but are nonetheless employed in many embedded applications.

But what then is an "embedded system?" This, too, is a topic subject to many on-going polemics on and off the Internet. Is a cell phone application an embedded system? A military radar system? A medical monitoring device, or a network router? For the purposes of this guide, an embedded system can be any of the above – any device, or application, that is not a personal computer or in the enterprise space can be considered "embedded." An embedded application usually means a device that employs at least one microcontroller or microprocessor, one whose software is not user-defined, one that has a generally long product life span, one whose resource constraints (power, space, fault tolerance) are more or less severe, and one for which failure is painful (as measured in catastrophic failures as in aerospace/military or very costly failures as in consumer electronics).

For the purposes of this guide, if you are designing a device or system which might benefit from an operating system and that device or system is *not* a personal computer or enterprise application, then this guide is for you.

If you would like to argue endlessly as to what is "embedded" and what is not, as to what is "hard" real-time and what is "soft" real-time – those debates are better left to academics and polemicists, and not to practical designers designing real products that need to be deployed in the real world, sooner rather than later.

WHO THIS GUIDE IS FOR

This guide is targeted at **embedded design engineers** or **programmers** who are actively selecting a real-time operating system. If you are deciding whether to "build" or to "buy" an RTOS – this guide is for you. If you are deciding whether to choose embedded *Linux* or *Windows XP* Embedded – this guide is for you. If you are deciding among the many commercial RTOSes – this guide is for you.

If you are aware that "time is money" – this guide is really for you. While a diligent researcher can certainly use the Internet to research almost every RTOS out there, as well as discover many usable white papers and other technical application notes, this guide consolidates all that information in one place.

Indeed, if you realize that making the wrong choice can have deadly consequences to your design, this guide is for you. The small amount spent to purchase the guide will be repaid many times over in the savings from making the wrong choice, or in failing to consider issues brought up in this guide.

Our core audience is those who are selecting an RTOS for a business purpose. Our readers are in the job of designing an embedded product that will be sold to some "end user" – whether that "end user" is a telephone OEM, the U.S. military, or a Junior High school student purchasing the latest cellular phone. The device must work, must work reliably, and must meet cost and performance parameters.

If you are a **manager** guiding an embedded design project, this guide is for you. It helps to overview the entire universe of choice and thus to better guide your team to an optimum RTOS selection.

If you are an **RTOS vendor**, this guide offers valuable insight into the way that users approach the selection of an embedded operating system. The survey chapter and chapter of email interviews with actual designers should be of great utility to your marketing planning. That said, this is not a guide about *market size* or *penetration*, i.e. similar to those by **Gartner Dataquest** or others geared towards the analyst/financial community.

This guide is for people actively deciding which RTOS will best suit their complex embedded design. Whom, after all, should they marry?

ABOUT THE AUTHOR AND eg3.com

Jason McDonald is Senior Editor at **eg3.com**, with over thirteen years experience covering the embedded and real-time software industries. Prior to **eg3.com**, he worked at **MW Media** and as a freelance author for many leading electronics publications. Dr. McDonald received his Ph.D. from the University of California, Berkeley, in 1992 and his B.A. from Harvard University in 1985.

Thanks go out to Noelle Decambra, Assistant Editor at **eg3.com** for her tireless work on the interviews and report formatting.

Founded in 1994, **eg3.com** is the oldest and largest Web portal and email alert service for embedded systems at http://www.eg3.com/. The site delivers free information on top news, white papers, demo's, cool web resources and more for over 200 keywords as well as email alerts on these topics. In addition, the site publishes a monthly eLetter reaching over 38,000 registered subscribers.

The RTOS Buy vs. Build Decision

Before you consider which specific RTOS meets your project needs, it is worthwhile to step back for a moment and look at the "big picture" of your design and its relationship to a possible operating system (OS). Do you really need an RTOS at all? Is it better to "build" or to "buy" an RTOS? How should you as think about this problem as a business person as well as a technical engineer?

- Your Project/Product Requirements
- Resource Constraints
- Cost Constraints
- Knowledge Budget & Time to Market
- Build, Buy, Hybridize
- Source Code & Uncertainty

Insiders' Guide Embedded RTOS: Buy vs. Build

YOUR PROJECT/PRODUCT REQUIREMENTS

It all starts with a product. What sort of device is your company building? Is it a new network router? A new cell phone? A PDA device? Perhaps a factory automation system, or perhaps even a single board computer that will GPS identify railway cars in transit? As you know, embedded devices run the gamut from the very simple to the very complex, from mass market consumer applications like cell phones to single-shot devices like Mars landers. Moreover, more and more of these devices are designed to communicate over networks, whether these are ethernet, the Internet, or mobile/wireless communications technologies like Zigbee or 802.11.

- The first step in identifying an optimal embedded RTOS is to outline, in as much detail as possible, the final product features.

In addition, you should also think of "feature creep," the tendency of future versions of a product to require more and more functionality, which may put more stress on the operating system. What will the product look like in Version 1.0? What about version 3.1? And so forth. This is all easier said than done, but ultimately the product features determine the product (application) software, and the application software is interdependent with the operating system.

Here is a chart with some common verticals and some common requirements.

VERTICAL MARKET	COMMON DEVICES	HARDWARE	SOFTWARE
Automotive – human interface systems	GPS navigation system, infotainment system	Industry-standard microprocessors and ASICs.	Soft real-time, but heavy on graphics-intensive software applications. User interface is paramount.
Automotive – command and control	Anti-lock brakes, engine monitoring, anti-roll-over mechanisms	Regulatory approval may be required for many hardware components, including microprocessors.	Hard real-time, safety critical. Minimal graphics, "invisible" to consumer. Often subject to regulatory approval.
Avionics / Military	Airplane guidance, radar monitoring, secure communications technologies	Regulatory approval required for many hardware components.	Hard real-time, safety critical, encryption/security features paramount. Often subject to regulatory approval.
Consumer	Cell phones, video games, MP3 players, smart phones.	Per unit, very cost sensitive. Rapid deployment and product obsolescence.	Hard or software real-time, depending on application and performance requirements. Rapid product introductions, feature creep, and obsolescence.

Medical	CAT scanners, pacemakers, blood monitors.	Regulatory approval required for many hardware components. Very long product lifecycles possible.	Hard real-time, safety critical. Security / privacy concerns. Extreme data processing.
Network / communications Infrastructure	Network routers, gateways, back-end blades	Industry standards like VME, CompactPCI, AdvancedTCA are very important.	Hard real-time especially for communications infrastructure, encryption/security requirements.
Industrial Process Control	Factory automation, robotics, data acquisition	Single board computers	Often requires both relatively hard real-time, and sophisticated graphical user interfaces.
Military	Field robots, networked communications devices	Subject to DOD requirements. Very long product lifecycles.	Hard real-time, encryption/security paramount, "safety critical."

Vertical markets aside, every design is unique. The point is to pre-specify the product requirements in as great a detail as possible before selecting a possible RTOS. After all, your future RTOS must be able to run any required application software, handle incoming and outgoing communications, and ideally facilitate current required application software. Looking to the future, your RTOS should be able to grow with the product, allowing new features as time goes by and not collapsing under complexity or unexpected new features (demanded by marketing or management). Detailed product specification is a critical foundation to a good RTOS choice.

RESOURCE CONSTRAINTS

The sales and marketing people often give you the product requirements: what is this product supposed to do? But as Scott Adam's *Dilbert* (h t t p : / / w w w . dilbert. c o m /) so often explains, the product spec's from marketing often collide head-on with the constraints of the device itself, especially in a business cost environment in which you have to come in under a per device "budget."

- The second major issue in selecting an embedded device, therefore, is the resource constraints of the device itself.

To run application such-and-such, how much memory would you need? How much memory will you have available? What is the per unit cost of this memory? What are the power requirements? Will your device be a low-power device driven by batteries, or a device that enjoys the luxury of being plugged in? What response time is required? Indeed, what sort of true "real-time performance" will your device need? Is it a "hard" real-time device, such as a military computer system or a medical device? Or is it a consumer device that can tolerate less true real-time

performance? In sum, what is your best guess as to the optimum mix of hardware and software that can meet the product requirements while staying under the per-unit budget?

Here are some common constraints:

ITEM	DESCRIPTION
Power	Is this a low power device? Will it run on batteries? How long does it need to run before a re-charge?
Processor	What sort of processor will the device have? 8-, 16-, 32-bit? Which particular architecture? If the processor has already been chosen, this pre-limits the selection of available RTOSes.
Memory/Footprint	How much and what type(s) of memory is available for the RTOS? How much additional is needed for applications? What are the trade-offs between the two?
Regulatory Requirements	What, if any, regulatory approval is required. In certain industries such as automotive, defense, aerospace or medical, regulatory approvals are very strict and severely limit the usability of both hardware and software.
Hard vs. Software Real-time	To what extent, must the device have hard real-time capabilities? Even if certain aspects of the design have real-time capabilities, these critical requirements can constrain the choice of hardware or software.
Durability/User Interaction	To what extent will this device be "on its own" – away from humans who can reset it? To what extent must it be long-term durable? Is it a military or industrial device that will operate in a harsh temperature or vibration environment?
Networking and/or Communications Constraints	Will the device be communicating, either over wires or wirelessly? If so, are there particular standards such as Zigbee, 802.11, or ethernet that will be required? How will these influence the choice of RTOS?

COST CONSTRAINTS

Closely related to the device resource constraints are the cost constraints. Embedded devices are not designed in a business vacuum – the "perfect" device configuration of hardware and software may nonetheless be impossible because it goes over the per-device budget.

- What is your per device total budget: hardware plus software?

This is a trickier specification than would appear at first glance because it is influenced by a number of factors. First and foremost, what features are **required** in the product and what features are desired, but **optional**? At some level adding new features means adding additional hardware or software, *or* improving the device performance to do more with less. Secondly, the per device cost reflects not just the bill of materials (BOM) but also the per device software cost (RTOS plus application software) plus an amortization of the "non-recoverable engineering costs" (NRE), i.e., those personnel, licensing, and other costs that are incurred on a per project basis but must be amortized across the product run. Finally, how many devices will be shipped? This gets into the area of forecasting, but obviously the per product cost of an *Xbox* shipping millions of devices is quite different than that of a CAT scanner shipping hundreds. Yet in either case the NRE must be spread across the devices to give an accurate estimate of the true cost per device.

At a business level, this is the critical financial equation:

$$\mathbf{P > C_h + C_s + (N_h / D_n) + (N_s / D_n)}$$

Where,

P = End price of the product (price to be charged to the buyer) must be greater than:

$\mathbf{C_h}$ = Per device hardware cost +

$\mathbf{C_s}$ = Per device software cost including RTOS royalties +

$\mathbf{(N_h}$ = Non-recoverable engineering costs (hardware) /

$\mathbf{D_n}$ = Total number of devices shipped (purchased by all end buyers)**)** +

$\mathbf{(N_s}$ = Non-recoverable engineering costs (software) /

$\mathbf{D_n}$ = Total number of devices shipped (purchased by all end buyers)**)**

NON-RECOVERABLE ENGINEERING (NRE) TRADE-OFFS

As we shall see throughout this report, there are many cost trade offs between the "design phase" or "non-recoverable engineering" costs and the "deploy phase" or "per device costs." For example:

- **"Better" RTOS vs. "better" hardware:**

  Using a leaner, more code-efficient RTOS and application software may reduce the required hardware or allow the same functionality to be achieved on less expensive, older generation microprocessors. Or, using more expensive hardware (i.e., 32-bit microprocessor vs. 8-bit) may allow a less expensive RTOS like *Linux*. *That is, the better/more expensive the RTOS the worse/cheaper the hardware can be and, to some extent, vice-versa.*

- **Commercial RTOS vs. in-house Software Design Costs**

  Employing a commercial RTOS, at least according to commercial RTOS vendors, should reduce your non-recoverable software design expenses (i.e., you will spend less personnel time designing your own in-house RTOS). *That is, the more funds devoted to "buying" a commercial RTOS the less funds will be spent on labor costs "building and/or debugging your own."*

- **Royalty Free vs. Per Device Royalty**.

  Most RTOSes are sold as "royalty free," which basically means that the total cost of the RTOS is embedded in the initial cost as compared to other RTOSes like *Windows CE* which are sold on a per-device royalty. *That is, "royalty free" RTOSes shift costs to the initial NRE/design phase (from **Cs** to (**Ns/Dn**)) while "per device royalty" RTOSes shift costs from the NRE/design phase to the deployment phase (from (**Ns/Dn**) to **Cs**).*

DESIGN & BUDGET TRADE OFFS

Suppose, for instance, that you work for a large corporation with a very large "design" budget. You may be more inclined to purchase an RTOS from a vendor that offers full support with no per-device royalties because these costs are budgeted in the "design" phase. On the other hand, suppose you work at a very small company with a great idea but little start-up cash. In this case, you might be better off with an RTOS such as *Windows CE* that has lower up-front costs and higher device royalties. **Microsoft** cleverly labels this strategy "shared success," meaning that much of the payment is shifted to after your device has been created and begun to ship. If, on the other hand, your device will go into thousands or millions of units, you may not want to haggle with an RTOS vendor on per-device costs.

Another very pertinent example of cost trade-offs is the difference between using embedded *Linux* (no up front cost, no royalties) vs. a proprietary RTOS like **Green Hill**'s *Velocity*. Because *Linux* is not a true real-time operating system it may require modifications and/or have a much bigger "footprint" to get the job done then *Velocity*. Therefore, it may require more extensive hardware and memory raising the per-unit hardware bill of materials, as well as more intensive work at the beginning raising the total engineering cost. There is no simple answer that "royalty free" or "open source" means inexpensive, and proprietary and "closed source" mean expensive – each is simply a different business model, and a different distribution of costs between the "design" phase and the "deployment" phase.

Indeed, the key to your value add as a designer is "designing smart." Figure out the best mix of hardware and software to move the total functionality to the right and total cost to the left.

KNOWLEDGE BUDGET & TIME TO MARKET

A final group of considerations might be termed your "knowledge" budget. This aspect is often overlooked, yet based on our survey results, is a huge component of choosing an RTOS. Basically, the trade off is between doing-it-yourself vs. paying a company or vendor to do it for you.

Questions abound:

- Are you a "do it yourselfer?" Is designing your own RTOS or customizing *Linux* or some other RTOS a task that you would both be good at and enjoy? Would your time be better spent on applications software or some other area?
- Are you an experienced C or Assembly-language programmer?
- Are you willing to invest in your own experience to design or customize an RTOS from the ground up?

- Or, are you better off purchasing an RTOS and relying on outside vendor experience?

Even if you purchase an RTOS, however, you must be aware of learning or knowledge issues. No RTOS is really usable "right out of the box." They all involve learning their ins and outs, and they all involve some degree of customization to your product.

Consider the availability of **technical support** and **technical documentation** for your RTOS. With *Linux* there is a huge worldwide community documenting and supporting the operating system, albeit largely for the non-RTOS versions of *Linux*. The technical documentation and support from a **QNX** or **Express Logic** may be worth its weight in gold compared with the loneliness of going it alone.

But don't delude yourself that outsourcing the RTOS portion of your project is a panacea. Explaining your needs and policing that the needs are being met are all "knowledge" costs as well.

You might, in sum, ask yourself, your team, and/or your company if you really want to get in the business of creating and maintaining an RTOS, or some flavor of 'embedded *Linux*.' Or, are you better of purchasing an RTOS and relying on an external vendor for that support.

There is no perfect answer. Some people cook from scratch (build-your-own RTOS). Some cook frozen pizza from the freezer (*Linux* or low-cost RTOS). And still others go to the fanciest restaurant in town (high-cost, full-support RTOS). Each choice has a set of pros and cons.

TIME TO MARKET

Time, it is often said, is money. Hence, another cost factor in embedded systems design is "time to market" or "time to revenue." Building your own RTOS, or hybridizing *Linux* or some other "free" RTOS might be cheaper in terms of initial outlays but it might be more time-intensive. If your product needs to hit a market window – say the Fall Christmas shopping season, or the rapid deployment of 802.11n devices – you might be better off with a "full service" commercial RTOS that, although more expensive, may help you with quick time to market. Only you can answer how important time to market is for your design but it is a significant factor in many cost equations.

Another way to look at this is "time to revenue" - the issue here is that spending less on the development phase by purchasing a cheaper or even free RTOS might not make financial sense if your product can get to market, and get to revenue, more cheaply with a commercial or other alternative. Saving money on the design phase might mean missing out on substantial revenue if the *market window* is missed. It might even mean project failure.

BUILD, BUY, HYBRIDIZE

Assuming that you have already decided that your application must have not just an operating system but an operating system that has some sort of "real-time" capability, the most fundamental decision is the decision of "buy vs. build." Is it a better business decision to "build your own" RTOS or to purchase a commercial RTOS? In the middle stand options like Embedded *Linux* or customizable commercial RTOSes – in these cases you "buy" or "download" an operating system and then further customize it to meet your design needs. According to our reader survey, many designs today still are based on "home-built" operating systems. That said, most other designs fall somewhere in the middle continuum. The operating system is purchased or downloaded, and then further modified to be optimized for your application.

Three basic alternatives exist:

1. **Build Your Own RTOS ("Build").** In this scenario, you decide what your design specifications are and you create an RTOS that, to the best of your abilities, perfectly maps these specifications.
2. **Buy an RTOS ("Buy").** In this scenario, you go to an RTOS vendor such as **eSOL, Express Logic**, or **Quadros.** You "purchase" their RTOS.

3. **Hybrid or Buy/Customize.** In the "hybrid" scenario, you buy or in the case of Embedded *Linux*, download an RTOS. You then customize it to meet your design criteria.

Each has its pros and cons.

BUILD YOUR OWN RTOS ("BUILD")

Consider, for example, building your own RTOS or "Build." At first glance, this sounds like the perfect choice. Because you will be building your RTOS from the ground up, it will be perfectly optimized for your application. With this perfect match you will be able to optimize its real-time capability, memory footprint, power consumption criteria, etc., to best take advantage of and meet within the design criteria. In a world without budget, knowledge, or cost constraints, perfect optimization would indeed be the best choice.

So here is a list of **PROS** for "Build:"

1. **Optimization.** By building the RTOS yourself, you will be able to optimize it for your design better than any off-the-shelf RTOS or *Linux*. Indeed, in some situations you might even not require a full RTOS and thus can fit into very small 8-bit or other resource constraints.
2. **Source Code**. You will create, understand, and own the source code. As the creator, you will have the best knowledge of this code as well as copyright to its use.
3. **Business Independence**. You and your company will not be dependent on an outside vendor or process for your RTOS and its support. You do not have the external risk of a supplier default or bankruptcy taking away future support.
4. **Monetary Cost**. Because you are creating your RTOS yourself, you do not incur any direct monetary costs.
5. **Licensing/legal**. Because you are creating the RTOS yourself, there are no royalty or licensing issues to worry about, and fewer legal issues concerning derivative software and patents (although there are still, of course, legal issues such as liability and patent infringement).

And here are the **CONS** for "Build:"

1. **Learning Curve**. Building an RTOS is not as easy as it sounds. Unless you are an expert programmer, this is not a project to be undertaken lightly. You may get in over your head.
2. **Personnel Dependence**. Because you or your company is creating the RTOS yourself, you may become very dependent on the person or person(s) who created the RTOS. There will usually be a lack of documentation – meaning that if this person/persons moves to a different job, dies, or whatever, you may lose the knowledge resources for future modifications and/or product support. Similarly, the pool of available programming talent will be very small for an In-House RTOS vs. a commercial RTOS or *Linux*.
3. **Time Costs**. Time is not free. Time spent creating your own RTOS is time that could have been spent on application programming or other issues. By building your own RTOS you are not using your time to do other things which you might do better: perhaps you are reinventing the wheel and/or reinventing the plumbing.
4. **Application or Community Ecosystem**. Because your RTOS will be unique to your application, there will be no external community of programmers or users. Any additional applications will have to be custom made for your RTOS. Compare this, for example, with *Windows XP Embedded* or Embedded *Linux* which each enjoy a healthy ecosystem of tools and applications.

5. **Feature Creep Costs**. If/when your application goes to its next version, management may very well want new features. What was initially a simple RTOS may now need to become even more complex to handle the new features. Slowly but surely, you get into the business of maintaining a pretty complex RTOS rather than focusing on your core competencies in applications.

BUY AN RTOS ("BUY")

In this pure scenario, you buy an RTOS from a vendor. Leaving aside embedded *Linux* which has its own set of unique characteristics, this means that you go to the websites of **Wind River**, **Green Hills, Micrium, Quadros, Mentor Graphics** or other RTOS vendors. You research the pros and cons of each choice. You may or may not consult with sales representatives or field application engineers (FAEs). You purchase the RTOS. You then spend some time learning about the RTOS, and finally you customize the RTOS (to the extent that it is customizable) for your target application and hardware.

Here is a list of **PROS** for buy:

1. **Time to Market**. Because you are not "reinventing the wheel," your purchased RTOS should speed your time to market.
2. **Application Programming Efficiency**. Because you have purchased an RTOS, you get to concentrate on your own application or value-add – you are not "reinventing the wheel," and thus you should get a benefit in terms of time efficiency by focusing on those software areas where you can really make a difference in terms of the product performance or features.
3. **Established Company**. Your purchased RTOS should come with the full backing and support of an established RTOS company that is (supposedly) an expert in the area of real-time systems. This should include enhanced **technical support**.
4. **RTOS Features and Performance**. The purchased RTOS should exceed the standards and performance of any RTOS that you could have created on your own. It should be fully tested and qualified, and thus, be a higher quality product than one you could create yourself. Indeed, because it has hopefully been deployed in many applications and environment, it should have much better understood bugs than a homegrown version 1.0.
5. **Community Support**. Many commercial RTOSes, and especially the large ones such as *VxWorks*, *Neutrino*, *LynxOS*, or *Windows CE* enjoy a large developer community. There you can share your experience and hopefully learn from the community. In addition, this gives you a pool of talent to hire from and/or outsource work to, as opposed to an in-house RTOS which will have limited external support (if any). Another benefit here is the in-the-field testing of an RTOS by many users in many different applications, which is a huge advantage in identifying software bugs.
6. **Software Longevity**. As opposed to an in-house RTOS, a commercial RTOS should survive in the marketplace longer, and thus, mean that your company and your design are not dependent on the one RTOS creator of an in-house RTOS. This lessens the single point of failure if that person dies, or moves on. In most cases, source code should be available, further guaranteeing its longevity.
7. **Software Documentation**. A commercial RTOS should come with extensive technical pre- and post-sales documentation. This should, again, allow many different people to understand and work on your product throughout the product life cycle.

And, of course, there are **CONS** to purchasing a commercial RTOS:

1. **Cost**. Commercial RTOSes are not free. There is usually an up-front fee, or per-seat license. These costs vary tremendously, but commercial companies are in the business of selling RTOSes, not giving them away.
2. **Learning Curve**. All commercial RTOSes will involve a learning curve – you and/or your software team will have to devote time and resources to learning and understanding this new software product.
3. **Licensing Fees and Issues.** Some, but not all, commercial RTOSes require a per-device license or royalty. Even if the RTOS is "royalty free," you still may have to investigate the various conditions of the license and track licensing issues. Others are sold on a "subscription" model of annual payments for technical support.
4. **Software Documentation**. A commercial RTOS should come with extensive technical pre- and post-sales documentation. This should, again, allow many different people to understand and work on your product throughout the product lifestyle, although admittedly there is a "learning curve" to using someone else's OS and/or documentation!
5. **Bugs and Software Defects**. No software is perfect. So even commercial RTOSes will have bugs and software defects, and you will still ultimately be responsible for making your product meet its requirements despite any unforeseen bugs.
6. **Company Risk**. The vendor of any commercial RTOS may at any time decide to discontinue the product, go out of business, or be acquired by another company. The RTOS highway of history is littered with former companies such as **Integrated Systems** and their RTOSes, which have been all but discontinued. In addition, RTOS companies certainly have an incentive to "lock you in" to their products and tool chains, so there is a downside risk that your choice of a commercial RTOS may "lock you in" to the product for years to come.
7. **Footprint**. Commercial RTOSes may be too big for severe resource constrained environments such as 8-bit processor designs or below, and/or may have too many unnecessary features thus requiring unnecessary hardware resources as well.
8. **Maloptimization**. By their very nature, commercial RTOSes are attempting to address more than one product's needs. All of them have various degrees and ways to be optimized for a particular product, but nonetheless the commercial RTOS will never be as perfectly optimized for your product as an RTOS created for your product from the ground-up. In other words, you are trading the perfect optimization of a "build-your-own RTOS" for the time-to-market, quality, and other features of a commercial RTOS. Actual performance may be less than the expected performance at the time of purchase!

HYBRIDIZE OR BUY/CUSTOMIZE / *LINUX*

Ultimately, when the decision is made to purchase a commercial RTOS, in almost every case the real process is thus a hybrid between "build" and "buy." Almost always, you will be "customizing" the commercial RTOS to meet the needs of your particular product. In this sense, many of the pros and cons of each pure choice may become an issue, or even a non-issue.

A highly customizable commercial RTOS, for example, may bring you many of the same performance enhancements that you could have gotten from building your own software. The quick time to market promised by a "buy" decision may be lengthened when the learning curve of the new software proves more difficult that perceived at the time of purchase.

Linux, which is discussed in detail later, is in this sense a hybrid between build and "buy." In some cases you actually do purchase a *Linux* distribution from a vendor such as **MontaVista** or **Wind River.** In most cases, you simply download a "free" distribution of *Linux* and begin to optimize it for your product and its needs.

Like a "build" decision, for example, you get total access to the source code with *Linux*. Like a "build" decision there is no monetary cost to obtaining the software. And like a "build" decision, you are making a commitment to spending time (and not necessarily money) optimizing the software for your product needs.

Like a "buy" decision, however, you are enjoying the community support and technical documentation available for *Linux*. Like a "buy" decision, you are agreeing to a specific set of licensing and licensing terms – you are agreeing, for example, to the terms of the GPL (GNU General Public License). Like a "buy" decision, you are exchanging the perfect optimization of a "build" RTOS for the off-the-shelf features of *Linux* – hoping to gain time to market, but exchanging some performance and device optimization.

Despite claims to the contrary, *Linux* is really not as radical a departure for the embedded RTOS marketplace as some would have you believe. *Linux* is essentially a hybrid between buy vs. build, or in this case buy vs. download. It has its own pros and cons, and should be considered as one choice among many. Let your specific needs and the specific requirements of your project drive your choice, not an ideology that says *Linux* is "good" and **Microsoft** *Windows CE* or other is "bad."

SOURCE CODE & UNCERTAINTY

Much discussion is made about whether an RTOS vendor supplies you with "source" or not. The question to ask is why you need source from a technical or business standpoint, and what are the licensing restrictions on altering this source. Basically, there are two major reasons to require source code. First, having the source code allows you to modify the RTOS to better suit your requirements. In this sense, having the source code allows you to blend the advantages of "purchasing" or "downloading" an RTOS with the advantages of creating your own proprietary RTOS. You can have your cake and eat it too, as it were.

That said, just having the source code doesn't make you an expert at it. Some of the larger RTOSes have thousands upon thousands (if not millions) of lines of code, and mastering some other company's code is no small feat. Just having access to the source code (think of *Linux*) doesn't mean that you will be effectively able to modify it.

The second reason to require source code is reducing business uncertainty. Having an effective license agreement that includes access to source means that even if the vendor goes out of business, your company will retain the ability to modify and continue its projects on the appropriate RTOS. Again it allows you to blend one of the major advantages of "build your own" (control of and longevity of source) with the ease-of-use of a purchased RTOS.

Uncertainty is a fact of business life, as much in the RTOS business as in any other. Companies come and go, RTOSes come and go – you can never be certain what the future will bring. Building your own RTOS seems to have the advantage of the "certainty" that you are the master of your own destiny while buying an RTOS seems to intertwine you with the fate of another company. But in the real world, employees come, go, and die (even if they are your company's own RTOS guru) so not even building your own RTOS truly relieves you of uncertainty. And buying someone else's RTOS with a license that allows your source code might not be so uncertain after all. In all cases, the devil (and rewards) lies in the details.

Overview of the Basic (Commercial) Types of RTOSes

*Besides defining your product specifications, it is useful to view the universe of available real-time operating systems in broad groups. In this section, we discuss the "Build your own" in-house RTOSes, broadly focused RTOSes, general purpose OSes like **Microsoft** Windows XP as used in embedded systems, and focused or "niche" RTOSes.*

- BASIC TYPES OF RTOSES
- "BUILD YOUR OWN" / IN-HOUSE RTOSES
- "BROADLY FOCUSED" COMMERCIAL RTOSES
- GPOSES (GENERAL PURPOSE OSES)
- FOCUSED OR "NICHE" RTOSES
- BEYOND THE RTOS ISSUES

BASIC TYPES OF RTOSES

Real-time Operating Systems (RTOS) can be divided into a few basic groups:

1. **"Build your own" / In-House RTOSes**. These are RTOSes that you build "yourself," and that are highly optimized by you and/or your company for your specific product(s) and specific needs.
2. **Broadly Focused Commercial RTOSes**. These are generally available RTOSes that can be purchased on the open market. RTOS vendors seek to provide effective, usable operating systems plus a wide variety of technical and tools support. Most charge either an up-front fee ("seat" or "project" cost) and/or a per-device royalty.
3. **General Purposes RTOSes / RTOS combinations.** Most importantly, **Microsoft** *Windows XP* but also embedded *Linux*, these are "general purpose" operating systems used in an embedded systems, usually (but not always) with some additional RTOS or RTOS-like software for those aspects of the design that require real-time performance.
4. **Narrowly Focused or Niche RTOSes.** These are RTOSes that either support one or a very few processors (e.g., ColdFire, or AVR) and/or are narrowly focused at a particular vertical market like automotive or avionics.
5. **Embedded *Linux***. *Linux* ranges from standard *Linux* distributions available over the Internet at no charge to specialized embedded *Linux* distributions for which vendors charge. In some situations, vendors charge for the *Linux* distribution itself; in others money is made off of selling technical and design services.
6. **Free or Open Source RTOSes**: with the exception of *Linux* (which actually fits this category but is important enough for its own special treatment), these are RTOSes that can be downloaded over the Internet and are subject to a number of licensing restrictions. A few are widely supported (such as *eCos* or *FreeRTOS*), but there are a huge number of university or project RTOSes available on the Internet.

In this chapter, we discuss in-house RTOSes, broadly focused RTOSes, GPOSes and narrowly focused or niche RTOSes. In the next chapter, we turn to embedded *Linux* and "free" RTOSes.

"BUILD YOUR OWN" / IN-HOUSE RTOSES

According to most surveys as well as our own RTOS survey, a sizable minority of developers still choose to "build their own" RTOS. People designing their own proprietary RTOS cluster are on two ends of the spectrum: those whose project is so small, so (relatively) simple and so specific that it justifies the effort to create their own RTOS, vs. those whose project is so "big," at least in terms of the design run, that it also justifies creating their own specific RTOS.

The simple, small 8-bit design example is well known in the RTOS literature. The situation here is that the design is simple, the performance required is minimal, and the cost constraint is severe. The developer then designs his own "simple" RTOS rather than purchase one from someone else.

On the other side of the spectrum are designs in which despite their complexity and performance requirements, the manufacturer intends to make so many devices that he is justified in creating his own RTOS. An example of this is **Cisco Systems**' *IOS* software. According to **Cisco:**

> **Cisco** *IOS*® Software is the world's leading network infrastructure software, delivering a seamless integration of technology innovation, business-critical services, and hardware support. Currently operating on millions of active systems, ranging from the small home office router to the core systems of the world's largest service provider networks, **Cisco** *IOS* Software is the most widely leveraged network infrastructure software in the world. (http://www.cisco.com

/en/US/products/sw/iosswrel/ios_software_collateral_library0900aecd800d8500.html , 11 October 2007).

As a major manufacturer of network equipment, **Cisco** has clearly felt justified in creating its own proprietary operating system. The situation has also been common at major cell phone and telecommunications infrastructure companies, automobile manufacturers, etc. – all markets in which there are large product runs, and long product life cycles. Over time, however, more and more of these large manufacturers are outsourcing their RTOS to commercial vendors and/or embedded *Linux*. Another example is **Nokia**'s s60 operating system (http://www.s60.com/).

BEST OF BOTH WORLDS? – NEW TOOLS TO BUILD YOUR OWN RTOS

If you want to build your own RTOS, then are you completely on your own? Most vendors of commercial RTOSes provide architectures and commensurate tools that allow designers to "optimize" the RTOS for their application. But a few vendors approach this problem from a different angle, providing tools to build your own RTOS from scratch as it were. Here are the most notable:

- **Zeidman Technologies** (http://www.zeidman.biz/) offers a product, called *SynthOS*, that allows you to "synthesize" your own RTOS. The catch is that it is based on Java, with all the resource-requirements that that implies. According to the product literature:

  > The concept of *SynthOS* is taken from hardware synthesis; you choose the programming language that you like, whether it's C, C++, Java, assembly, or any other language currently supported by *SynthOS*. The output code, after synthesis, is in exactly the same language. This allows you to preserve all of the tools that you currently use — compilers, debuggers, interpreters, emulators, etc. (http://www.zeidman.biz/synthos.htm, 11 October 2007)

- *WhatOS* is a similar attempt. It is a small Python framework for creating applications consisting of communicating state machines. It allows a Python programmer to write state machines, called tasks, plug them together in a system, and simulate that system. Ultimately you end up with a C file that can be compiled and become a stand-alone RTOS for your application. You can read more about *WhatOS* at http://www.sticlete.com/whatos/.

- Alternatively, if you want to switch from one RTOS to another – **MapuSoft Technologies** (http://www.oschanger.com/) provides a nifty product, called *OS Changer*, that allows easy porting from one RTOS to another. Their other product, *OS Abstractor*, allows you to develop software with portability in mind – develop it once in *OS Abstractor*, and port it subsequently to a number of competing RTOSes yourself.

"BROADLY FOCUSED" COMMERCIAL RTOSES

Prior to the *Linux* revolution, a designer's only choice beyond "doing it himself" was to purchase a commercial RTOS from a vendor like **Wind River Systems** (http://www.windriver.com/), **Mentor Graphics** (http://www.mentor.com/), or **Express Logic** (http://www.rtos.com/). According to our survey data, this group continues to grow at the expense of the in-house RTOS group but is increasingly challenged by the growth of embedded *Linux*.

COMMON BUSINESS FEATURES OF COMMERCIAL RTOSES

At a business level, commercial RTOSes often differ from each other on the issue of per device royalties. Most are now "royalty free" (meaning all payment is up front, and it doesn't matter how

many devices you ultimately ship), but a few remain "run-time royalty" meaning that a fee is charged for each device shipped with the RTOS. Furthermore, with "royalty free" licensing, you pay an upfront fee usually based on variations of :

- **Per product license** – a fee to use the RTOS for one design;
- **Per product family license** – a fee to use the RTOS for a "product family," as in a family of printers or digital cameras;
- **Per microprocessor license** – a fee to use the RTOS for a microprocessor or microprocessor family, as in the 8051, ARM, or MIPS;
- **Per seat license** – a fee paid for each "developer" using the RTOS, often charged as a "subscription" fee paid annually; and/or
- **Unlimited license** – a fee paid for use of the RTOS in an unlimited number of designs.

Technical support is usually included at no cost, or in some cases, it can be negotiated for an extended time period. In most cases, you can request a published price list from a vendor sales representative and, in some cases, you can negotiate different pricing depending on your needs and your design issues.

At the licensing level, commercial licenses often differ on the level of "source" code – do you get the "source code," and if so can you "modify it?" If you modify it, what does this do to your technical support? In most cases, it is pretty straightforward, and most vendors give you access to source these days. It is a very useful thing to have, since having the source allows you to:

1. **modify** the source to deal with bugs and/or add new features to the RTOS; and
2. **maintain** the source code in the future in case the vendor goes out of business and/or abandons the RTOS (business security).

Some RTOSes are also distributed in binary form.

The great thing about commercial licensing (as opposed to *Linux*) is that you generally have clear rights and are thus free from worrying about patent or copyright infringement issues as well as are free from the GPL-related issues of having to "open source" any changes you make to the RTOS and/or derivative software (discussed below under *Linux).* Indeed, many commercial vendors will *indemnify* your application and revenue stream from later legal troubles caused by presently unknown patent or copyright infringements by their RTOS - something most *Linux* vendors are loathe to do.

COMMON TECHNICAL FEATURES OF COMMERCIAL RTOSES

At a technical level, commercial RTOSes differ substantially. The problem is that the "devil is in the details." There aren't really any good publicly available benchmarks for RTOS performance – each RTOS vendor proclaims that their RTOS is the "smallest," "most responsive," "fastest," etc., based on whatever parameters they define. Since your design is unique both in terms of features needed and in resource constraints, the ideal way to compare RTOSes is to get some preliminary access (under non-disclosure) to the competing RTOSes that make your "short list."

- Most vendor websites have demos, some have full-featured versions of their RTOSes available, and still others will agree to provide you with an RTOS copy after signing an NDA. (*Check the Appendices, and we provide the URL where available).*

What this means is that if at all possible, you should narrow your RTOS search to just a few, and then spend the time "prototyping" your application on several of the competing vendor RTOSes. After doing this "test drive," you will be able to make a good decision.

At a more abstract level, here are some common points of comparison for commercial RTOSes:

- **Hard vs. Soft Real-time Performance**. The machismo conflict of the embedded RTOS market – is the RTOS truly "hard" or "soft?" Compare each RTOS on the parameters of their hard or soft real-time performance, but keep in mind the extent to which your design actually requires "true" real-time performance. In addition, there seems to be a negative relationship between real-time performance and application software, with both *Windows XP* and *Linux* offering lots of application software but less than true real-time performance.
- **Processor Support**. Not all RTOSes support all processors. So if you are already wed to a particular microcontroller/microprocessor, this will pre-limit your RTOS selection.
- **Virtualization / Multicore.** Many commercial RTOSes have worked hard on their own support for multicore and/or software virtualization. Others have partnered with leading companies like **VirtualLogix** (h t t p : / / w w w . virtuallogix. c o m /) to embrace the virtualization / multicore trend.
- **RTOS Footprint/Modality**. Can you easily add (or subtract) items to the RTOS kernel to exchange footprint size for functions? Most RTOSes claim that their architecture offers a superior way to exchange size for function.
- **Development Tools**. Some RTOSes have a wide ecosystem of other software tools that support their RTOS and work well with it. Others, especially the smaller RTOSes, may not have such great relationships with other tools.
- **Affiliated Software**. Is your product network enabled? Will you need a TCP/IP stack? Perhaps a Graphical User Interface (GUI) for use controls? Different RTOSes have different relationships with affiliated software (either theirs or some other companies). The ease-of-use in designing with the RTOS can be influenced by the availability of these other design software products.
- **Application Software**. Another area of intense competition is the application software that is available (or can be created) for the software. This is an area where *Windows XP embedded* shines (as does *Linux*) – there is a wealth of additional software written for it as well as a huge cadre of available programmers.
- **Vertical Market Focus**. Some RTOS companies clearly (or not so clearly) focus on one particular vertical market such as consumer or defense. While they may sell their RTOS into other markets, they clearly focus their resources and expertise on one vertical. If your application is in their target vertical, then they are really worth a look.
- **Regulatory or Other Compliance**. Some RTOSes comply to certain industry standards like DOD-178B, POSIX, MISRA, iTRON, etc. Again, if your design requires or benefits from compliance, then these vendors are worth a look - especially true if your design is in the military area.

BROADLY FOCUSED RTOS SUBGROUP

Within the commercial RTOS group, there are broad subgroups of RTOSes, which we shall now analyze in turn. The first subgroup is "broadly focused" real-time operating systems. Whether from a big or small company, these RTOSes often share the following characteristics:

- **Broad microprocessor support** – a wide range of microcontrollers, microprocessors, and variants are supported, and usually there is a relatively swift "porting" of the RTOS to new chips as they enter the market;
- **Broad vertical market support** – these RTOSes aim to support the needs of a variety of application markets (usually, though not always, through software componentization), thus hoping to allow you to learn "one RTOS" and apply it to many application areas;

- **Broad affiliated software** – these RTOSes usually have a wide variety of related software, especially networking, drivers, "middleware," etc. thereby allowing you again to concentrate more fully on application software; and/or
- **Broad tools support** – in some cases, these RTOS vendors provide "integrated development environments," compilers and other software tools that work hand-in-hand with the RTOS.

RTOS/TOOLS PLAY

One very important differentiator among companies that offer "broadly focused" real-time operating systems is software tools support. Generally, all claim to work "well" with other software tools, both GNU and commercial. But a few really emphasize their tool chains and, in the case of **Wind River**, actually emphasize the tools and technical support more than the RTOS choice itself.

Consider, for example, the **Embedded Software Division of Mentor Graphics**, formerly known as **Accelerated Technology. Mentor's** *Nucleus* RTOS is a royalty-free, broadly focused real-time operating system that supports a very wide variety of hardware architectures. There are also C, C++, OSEK, m iTRON and POSIX versions of the RTOS kernels. The company also has a wide variety of software tools support (especially in the UML area) as well as their own IDE, *Nucleus Edge*. As a differentiator, the company emphasizes the relationship between its embedded systems and RTOS products, and its higher-level, EDA or SOC products. In addition, the software tools support for the product line is emphasized.

Another major vendor with both "broadly focused" RTOS products and excellent tools support is of course **Green Hills Software** (http://www.ghs.com/). Their flagship RTOS is called, "*Integrity*®," is built on the *velOSity* microkernel, the product is POSIX-certified and is focused on embedded systems that require maximum reliability. Translation: the product focuses on "hard" real-time constraints, especially in safety-critical or defense niche applications. The company has also recently launched *µ-velOSity*, a smaller less expensive real-time kernel designed to broaden its reach into other vertical markets.

A final vendor offering both a "broadly focused" RTOS and an emphasis on tools integration is **Wind River Systems** (http://www.wrs.com/). The company has recently shifted to a strategy of embracing both *VxWorks* and embedded *Linux* – uniting both with the new term, "DSO" for Device Software Optimization. The concept attempts to "broaden" the company's appeal beyond real-time operating systems into the more general area of optimizing software for devices that interface to each other and to the broader (Internet) network. At its base, however, it still has many elements of a tools strategy – indeed, it is so much a tools strategy that the tool chain itself (renamed, *Wind River Workbench)* can be seen as more important than the RTOS itself. The company has "platforms" focused on specific verticals – general purpose, automotive, consumer, industrial, network, and safety critical. It provides *Workbench* as an eclipse-based IDE, as well as a cornucopia of debuggers, compilers, and other development tools. Its architecture support is second-to-none.

For a list of "broad" RTOSes, please see Appendix B.

GPOSES (GENERAL PURPOSE OSES)

General purpose operating systems (GPOSes) are decidedly not real-time. A few like **Microsoft** *DOS* have a small but steady following among embedded vendors. But none offer true "real-time" performance. Beside *Linux*¸ the most important GPOS is of course *Windows XP*. The problem is that as a GPOS, *Windows* isn't built for task prioritization. GPOSes as a general rule simply do not offer the sort of preemptive multitasking that allows an RTOS to prioritize which functions

must be completed first and which are of lower priority. They also are not usually componentized so that software footprints can be reduced and better optimized to exact device needs. What features bring developers to use a GPOS like *Windows XP* into embedded systems? Here are some common ones:

1. The end application might not require "real-time" performance and/or the tasks can be segregated into real-time functions and non real-time functions.
2. GPOSes like *Windows XP/ Vista* and embedded *Linux* have a very large availability of drivers, application software, and other software elements that make development quick and easy.
3. GPOSes offer rich graphical user interface and multimedia capabilities that may be very important in vertical markets like industrial control, kiosks, and consumer applications.
4. Cost – GPOSes can be less expensive than commercial RTOSes.
5. Availability of developer talent – there are many, many more programmers of *Windows XP / Vista* than for commercial RTOSes.
6. Longevity – particularly with **Microsoft** *Windows XP*, the software will clearly be around and be supported for a long time by a major brand that "validates" your operating system choice to senior management.

Microsoft has taken notice of embedded systems in a big way, and has three key products targeted at embedded systems:

1. *.Net Microframework* - an environment that extends the advantages of Microsoft .NET and the toolset in the Microsoft Visual Studio development system into a class of smaller, less expensive, and more resource-constrained devices than previously possible with other Microsoft embedded offerings (h t t p : / / msdn2.microsoft. c o m /en-us/embedded/bb267253.aspx 11 October 2007).
2. *Windows CE* – the company's "hard" real-time operating system supporting *x86*, *MIPS*, *SH*, and *ARM* with source available.
3. *Windows XP embedded* a componentized version of the GPOS *Windows XP* supporting only the x86 architecture, allowing the developer to select only those elements required, thus reducing software footprint and costs.

The company has a specific website (h t t p : / / w w w . microsoft. c o m /embedded) focused on embedded systems applications, numerous online tutorials and forums, and a cool yearly developers' conference. The Redmond giant is clearly motivated for its two-pronged strategy to succeed in the embedded marketplace.

Assume for a moment that you are attracted to *Windows XP*, but your some parts of your application still require real-time performance. What do you do? Fortunately, a few third party vendors have targeted just application area:

- **Ardence** (h t t p : / / w w w . ardence. c o m /) – provides *RTX*® – Real-time Extension for Control of *Windows*®, designed to perfect the real-time performance of *Windows*.
- **Tenasys** (h t t p : / / w w w . tenasys. c o m /) – provides *INTime* bringing real-time performance to *Windows* applications
- **Kuka Roboter GmbH** (h t t p : / / w w w . kuka-controls. c o m /) – offering *CeWin*™ which allows *Windows CE* or *VxWorks* to co-exist with *Windows* on the same device, thus bringing real-time performance to *Windows* applications in this dual fashion.

FOCUSED OR "NICHE" RTOSES

As opposed to trying to be all things to all people, another subgroup of RTOS vendors has taken the focused or "niche" approached. The products of these companies focus on a specific microprocessor, DSP, and/or vertical market – they seek to be highly optimized for a specific niche, and target developers and developments in that niche.

Here are some examples of focused or "niche" RTOSes:

- *Salvo*™ - designed expressly for very-low-cost embedded systems with severely limited program and data memory, supporting 8051, ARM, AVR, 68HC11, MSP430, Microchip PIC, and TI DSPs (h t t p : / / w w w . pumpkininc. c o m / 11 October 2007)
- *Rubus* - designed to support Java and RTSJ. The services supported are based on the ANSI/IEEE standard 1003.1 (POSIX). Rubus JOS has emanated from Rubus® OS kernel designed by Arcticus Systems for the automotive industry h t t p : / / w w w . arcticus-systems. c o m /productsrubusjos.php 11 October 2007)
- *Talon*™ *DSP RTOS* - suite of five (5) POSIX® Compliant Real Time Operating Systems that are individually optimized for each TI DSP family (h t t p : / / w w w . blackhawk-dsp. c o m /Talon.aspx 11 October 2007)
- *Nimble* - the System on Chip RTOS, specifically designed for deeply embedded applications from **Eddy Solutions** (h t t p : / / w w w . eddysolutions. c o m /).

For a list of "Niche" or "Focused" RTOSes, please see Appendix C.

From a business perspective, these "niche" RTOSes give up a few advantages of their broader cousins:

- Since they are optimized on one particular vertical and/or chip, you may have to migrate to a different RTOS in your next design (thus losing some of the "learning curve" benefits from a broader RTOS, as well as perhaps pricing discounts).
- Since these are highly focused companies, they may be less likely to stay in business then their broader cousins.

If, however, your design matches the focus of one of these vendors, this RTOS type can be a great match.

BEYOND THE RTOS ISSUES

Time waits for no man, and RTOS developments are no exception. Some of the most current issues in the RTOS market are:

- **Virtualization**. Many new companies such as **VirtualLogix** (h t t p : / / w w w . virtuallogix. c o m /) and **Trango Virtual Processors** (h t t p : / / w w w . trango-vp. c o m /) have begun to bring "virtualization" into the RTOS mainstream. In some cases, their products work with an RTOS and in others they can substitute for the RTOS. In either case, "virtualization" is an important new trend in the RTOS area.
- **Multicore**. Closely related to "virtualization" is the "multicore" trend. Here, processor companies like **Intel** (h t t p : / / w w w . intel. c o m /) or **Texas Instruments** (h t t p : /

/ w w w . ti. c o m /) have brought to market new multicore processors. The design challenge is to make the software (whether RTOS or application) harness the new multi-core capabilities.

- **Middleware**. Other companies like **Enea** (h t t p : / / w w w . enea. c o m /) have exploited the market opportunity posed by that fuzzy concept of "application middleware" especially in the telecommunications vertical. Still other competitors like **Quadros** (h t t p : / / w w w . quadros. c o m /) define middleware in a less trendy way but emphasize the benefits nonetheless of all their add-on software (such as TCP/IP, file system, etc.) that really make one RTOS shine out above other competiting solutions**.**
- **Linux**. *Linux* has been hyped as a revolutionary force in embedded software and it has certainly changed the market position of major companies like **Wind River** (h t t p : / / w w w . windriver. c o m /) and even **QNX** (h t t p : / / w w w . qnx. c o m /). While not a new trend, the impact of *Linux* on the RTOS market cannot be underestimated - and you should consider *Linux* as a possible solution. Even if rejected, it may have an important impact on your RTOS choice, whether commercial, do-it-yourself, or non-Linux open in the future.
- **Security**. As more and more devices are networked, the need for security in all senses of the word is becoming more and more important. For some devices such as medical or aerospace applications, the need for security is obvious: lives will be lost if a device fails. For others, such as industrial control or consumer, the need for security might not be as obvious nor the implications if security fails. In the final analysis, however, it is often the RTOS that bears responsibility for preventing device failure and thwarting external attacks - so think "out of the box" about security as you design your application.

Finally, the RTOS market is very much connected to the embedded tools market. Companies as diverse as **Quadros** and **IAR** (h t t p : / / w w w . iar. c o m /) have recently emphasized the "RTOS / Tools connection." Linux vendors like **Wind River** and **MontaVista** (h t t p : / / w w w . mvista. c o m /) have recently emphasized their tool offerings as a real value versus off-the-shelf *Linux*. Perhaps you should look at your RTOS choice within the context of not just microprocessor/controller issues but also your personal tools ecosystem. Look to the Guide's interview section below for many excellent discussions on "beyond the RTOS" issues.

The RTOS Ecosystem (Free & Linux)

Embedded Linux may be the most exciting development of the RTOS market in the last ten years. Subject to the GPL license, Linux is easy-to-get, royalty free, and enjoys a vibrant ecosystem. A number of RTOS vendors now support embedded Linux designs. In addition, there are other, less well-known "free" RTOSes available on the Net. In this section we overview these intriguing alternatives.

- WHAT IS "FREE" SOFTWARE?
- LICENSING ISSUES
- NON-*LINUX* "FREE" REAL-TIME OPERATING SYSTEMS
- EMBEDDED AND/OR REAL-TIME *LINUX*

WHAT IS "FREE" SOFTWARE?

There is nothing better than "free," is there? Alternatively, "there is no such thing as a free lunch." The RTOS market has not been immune from the age-old argument over the value of, or possible deception in, the concept of "free."

Linux and several other RTOSes are billed as "free" software. It has been great viral marketing, giving *Linux* in particular a brand image of free, almost counter-cultural software. Before we consider the technical issues of "free" RTOSes and embedded *Linux*, let's pause for a moment and contemplate what "free" means in this context.

"Free" like so many words in the English language has more than one meaning. "Free" can mean "no cost," as in a "free lunch." Or, "free" can mean liberty as in "freedom." And "free" can also mean unencumbered by legal restrictions as in a "free" license. *Linux* has all of these connotations for free:

- *Linux* is "free" in the no cost sense – one can download the complete kernel from http://www.kernel.org/ at no cost, as compared with **Microsoft** *Windows* and many other RTOSes that have an initial cost if not royalties. It is also "free" in the sense of no royalties – users of *Linux* software do not have to pay a per device "royalty."
- *Linux* is "free" as in freedom – you, as a designer, can do *almost* anything you want with *Linux*, especially if you are not charging for your software enhancements to the kernel. You also get the full source.
- *Linux* is "free" in the sense that it is not restricted by a commercial software license – you can distribute your software to others without worrying about a manufacturer getting upset at you.

Other "free" RTOSes share some, if not all, of these characteristics.

WHAT "FREE" ISN'T

There are many costs associated with all "free" software, including *Linux*. At a high level of generality, here are some of the costs:

- **Time is money**. "Free" may mean "free" to download, but the software will still have to be optimized for your project, something that can be especially onerous for *Linux* since it is a general purpose OS and not a true real-time OS. In addition, there is the "learning curve" associated with all software.
- **Support**. Technical support is usually included with an RTOS purchase, but for *Linux* and other free RTOSes, technical support is available for free only from the online community (where you may get what you pay for) and/or from reputable vendors who sell services and technical support contracts. So, whereas the software may be "free" the support is not.
- **Hardware & Bill of Materials (BOM) Costs.** *Linux* and some of the other RTOSes can be less optimal than commercial RTOSes, thereby raising the cost of hardware (more memory, more advanced microprocessors cost more money). So "free" in this sense simply shifts costs from software to hardware.
- **NRE vs. Per Device Royalties.** *Linux* and other "free" RTOSes may not have per device royalties, but lack of technical support and sub-optimal real-time performance may mean more employee time in the "design phase," thus raising the NRE cost even as it lowers the per device royalty cost.

LICENSING ISSUES

Surprisingly, licensing issues for "free" software are in many ways more (not less) complex than for commercial software. "Free," as the folks at the **Free Software Foundation** (http://www.fsf.org/), are found of pointing out means "free" as in freedom, not free as in no-cost. Their home page announces, "Free software is a matter of liberty not price. You should think of "free" as in "free speech" (http://www.fsf.org/ 11 October 2007). Free in their definition hinges mainly on access to source code, and on the "freedom" to modify this source code and redistribute it to others. Free in this sense is a rebellion against software royalties and software copyrights.

> **Indeed, uncertainty over *Linux* licensing and intellectual property issues was a major negative in our survey results.**

Most (but not necessarily all) of the free real-time operating systems base themselves on the GPL (GNU General Public License) (http://www.gnu.org/copyleft/gpl.html). The GPL grants the recipients of a computer program the following rights:

- the right to run the program, for any desired purpose;
- the right to study how the program works, and modify it. (Access to the source code is a precondition for this);
- the right to redistribute copies; and
- the right to improve the program, and release the improvements to the public. (Access to the source code is a precondition for this).

This contrasts with commercial licenses which even if "royalty free" usually prevent you from distributing copies of the software for other uses and/or distributing copies of your changes to the source code (if you even have access to it). But with the freedoms of GPL comes a huge responsibility:

> You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License. (The GNU General Public License (GPL), Version 2, Section 0.2 (http://www.opensource.org/licenses/gpl-license.php, 11 October 2007).

What this means is that the GPL requires that *derived* software also be subject to GPL "free" distributions. (The BSD license and other modified GPL licenses often allow derived software to be exempt from this restriction). (Source: http://en.wikipedia.org/wiki/GPL, 11 October 2007). For a further overview of free and Open Source licensing, see Donald K. Rosenberg, "Evaluation of Public Software Licenses," (October 1998) at http://www.stromian.com/Public_Licenses.html 10 December 2007).

The common way out for vendors is to modify the GPL or use the BSD or other Open Source licenses that do not fully require that software derived from Open Source software also be open sourced. But in the final analysis if you are a producer of embedded products with software there is no doubt that the GPL or other Open Source licenses create a new layer of complexity that does not exist with simpler commercial licenses.

IMPLICATIONS OF FREE / GPL LICENSING

How does "free" affect your design? Beyond the cost and dissemination issues outlined above, GPL gives some need for caution in approaching embedded software designs based on GPL software. One way that "open source" software can cause problems is because of the "community" nature of open source projects. When someone copies "in" copyrighted software into the open source software and then that software would subsequently be incorporated into your own product. The most famous example of this is the lawsuit by **SCO** against **IBM**, claiming that

IBM put proprietary UNIX code (owned by **SCO**) into *Linux*. **SCO** argued that **IBM** and others owed them royalties for their code. If your company is legally very conservative, there is thus some concern that open source software of any type opens up the possibility for this sort of lawsuit in the future.

Another problem is sometimes referred to as "GPL contamination." Unlike commercial software, the GPL requires that you "open source" any changes that you make to the *Linux* kernel or other GPL software. If, for example, you modify the *Linux* source you are not only obligated to release this back to the community for free, you are also barred from charging for this modification. A less clear example is application software running on top of *Linux*. Most everyone believes that this sort of software is *not* subject to GPL restrictions, but there are a few zealots that believe otherwise. So again you are exposed to legal ambiguities with GPL licensing that does not occur with straightforward commercial licenses. There is even an organization called "gpl-violations. o r g" (h t t p : / / w w w . gpl-violations. o r g/) which seeks to identify and litigate against entities that violate the GPL license.

In conclusion, one big negative about all GPL software is that it opens up a fair amount of legal ambiguity. You are wise to check with your legal department during the early phases of an RTOS design, rather than be subject to a nasty licensing surprise down the road.

NON-*LINUX* "FREE" REAL-TIME OPERATING SYSTEMS

Similarly to commercial operating systems, there are different broad groups of "free" RTOSes. Some are general purpose, some are very niche-specific, and some are university or personal projects that are not really intended to be inserted into a commercial product. Indeed, some are "ghost" projects on the Internet and others are "loss leaders" from commercial vendors, so before you spend your valuable time on a "free" RTOS make sure that the project is still active, and is not some "ghost" site on the Internet.

BROAD FREE RTOSES

eCos

- One of the most successful open source RTOSes has been *eCos*. Originally created by **Cygnus Solutions,** *eCos* was meant to stand for "embedded Cygnus operating system." Since **Cygnus** was acquired by **Red Hat**, however, *eCos* has gained its own Internet existence at h t t p : / / ecos.sourceware. o r g/. It supports a large variety of architectures – ARM, CalmRISC, FR-V, H8, IA32, M68K, Matsushita AM3x, MIPS, NEC V8xx, PowerPC, SPARC, SuperH.

eCos is subject to a modified version of the GPL, with an amendment relaxing the requirement that derivative code or software be subject to the GPL (h t t p : / / ecos.sourceware. o r g/license-overview.html).

eCos is supported in the commercial sphere by **eCosCentric** (h t t p : / / w w w . ecoscentric. c o m /), which follows a business model not unlike the embedded *Linux* vendors with a more hardened version, additional development tools, and available (paid) technical support.

FreeRTOS. o r g

Alternatively, you might consider "*FreeRTOS*" (h t t p : / / w w w . freertos. o r g/). This RTOS is provided under the GPL license with an exception for interrelated commercial applications designed to allow for commercial development more easily. Created by Richard Barry, this RTOS is ported to 68HCxx, AVR, PIC, Microblaze, ARM7, 8051, and H8 among others. If the modified GPL is not suitable for your application, a commercial license is also available. Paid technical support is available.

RTEMS

Another free RTOS is *RTEMS* (http://www.rtems.com/), developed by **OAR Corporation** for the U.S. Army Missile Command. **OAR Corporation** provides support, training, and custom development services. Its focus is thus on hard real-time, and it supports a variety of architectures – ARM, TI DSPs, x86, i960, 68K, MIPS, PowerPC, SH, and SPARC among others. Best of all, it seems to be actively maintained and supported by a number of vendors which provide services and training. It is licensed under the GPL with a similar exemption to *eCos* to allow for commercial development.

For a full list of available free RTOS, check Appendix G.

NARROWLY FOCUSED OR "NICHE" RTOSES

Similarly to the commercial RTOS market, there are many RTOSes that focus on specific processor(s) and/or very specific vertical applications. If your processor or application is supported, you may be in luck. In all cases, you need to do due diligence on the licensing restrictions, technical features, and the extent to which the RTOS is actually being supported these days (as opposed to being a dormant website). Here are a few interesting ones:

- **TinyOS** (http://webs.cs.berkeley.edu/tos/). Created at U.C. Berkeley, this RTOS specializes in just one application area – wireless embedded sensor networks.

- **PICos18** (http://www.picos18.com/index_us.htm) is an operating system based on OSEK/VDX, an open industry standard. The OSEK standard is well suited for the low computing, power, small memory embedded controllers used in automotive or robotic applications.

- **AvrX** (http://www.barello.net/). *AvrX* is a Real Time Multitasking Kernel written for the **Atmel** AVR series of micro controllers. *AvrX* contains approximately 40 API in the six categories.

- **MaRTE OS** (http://marte.unican.es/). A real-time kernel for embedded applications that follows the Minimal Real-Time POSIX.13 subset. Supports x86.

- **TNKernel** (http://www.tnkernel.com/). Real-time kernel for embedded 32/16 bits microprocessors. Supports ARM architecture. FreeBSD-like license.

(For a complete list, see Appendix G. There are also, unfortunately, many "walking dead" free RTOSes on the Internet. By this, we mean RTOSes that have not been updated in several years and that seem to be just ghost pages of some student or university project that has long since been abandoned. Just be sure that any RTOS you investigate has at least a minimal pulse of life, before you invest your time and effort!

HYBRID OPEN SOURCE RTOSES

A final category might be called "hybrid." In these the "free" RTOSes are in some sense "demos" of a company's software or are meant to be used in conjunction with other products (or services) that have a financial cost. They are often not meant so much as stand-alone products but rather as pieces of a larger puzzle.

Unfortunately, there is a lot of marketing-speak out there, so it is difficult to always determine what kind of "free" RTOS each one is. But be aware of this marketing strategy, and be sure to investigate the limitations and interweavings of any potential RTOS to other commercial projects.

Here are some examples.

- **ERIKA Educational** (http://erika.sssup.it/). The "free" GPL educational version of **ERIKA Enterprise** (http://www.evidence.eu.com/content/view/27/254/),

although the site warns that the former is for 8-bit architectures and the latter is for industrial vertical markets.

- **NicheTask** (http://www.freertos.com/). A royalty-free, free RTOS distributed under a GPL-like license (http://www.freertos.com/register.php). Ultimately the product is closely connected to network protocol stacks available from **InterNiche** (http://www.iniche.com/).

- **Neutrino** (http://www.qnx.com/products/neutrino_rtos/). New for 2007, **QNX** has released the source code to their *Neutrino* operating system under a quasi-Open Source license. Essentially, you are free to download and use *Neutrino* in *non-commercial* applications. If/when you want to use the software for a commercial application, you must purchase a license.

Two other examples of this are as follows. First, there is is DSP/BIOS (http://focus.ti.com/docs/toolsw/folders/print/dspbios.html) a free real-time kernel from **Texas Instruments** (http://www.ti.com/). Obviously this RTOS is closely tied to **TI** DSP's and is provided by the company as a service to designers using its DSPs. Is it "free"? Nominally yes, but since the DSPs are not free one can argue it is essentially a marketing tool used to make **TI** DSPs more attractive. Second there are numerous examples of "free Linux" provided by hardware or board manufacturers. In most cases, this is "sample" Linux that is meant more for evaluation purposes than for production.

- The bottom line is when you hear anyone advocating that his or her RTOS is "free," be skeptical and investigate deeply what this means.

Check out Appendix G for a complete list of free RTOSes. From this brief survey, one can see that many of the issues that are important for choosing a commercial RTOS reoccur when selecting a free RTOS:

- **Real-time performance, footprint, technical specifications.**
- **Architectures supported**. Hardware support is important, as well as any implications for BOM costs due to RTOS (in) efficiencies.
- **Stability and longevity of the supporting organization** (will this RTOS be around for a long time?)
- **Licensing issues**. Is it covered by the GPL? If so, is it modified to allow for commercial application development? Is the license clear?
- **Hidden costs**. Is this RTOS really a ploy to sell services or add-on products from a commercial vendor?

In addition, think of the "human costs" of learning each RTOS, the availability of software such as drivers supporting the RTOS, and the whole "ecosystem" around each RTOS. These factors may mitigate the advantage of the "free" RTOS. Finally, think of the future. Will this RTOS be around in the future? Will it have a healthy ecosystem around it? You are making a commitment to the RTOS and consider the advantages or disadvantages of being an expert in it.

EMBEDDED AND/OR REAL-TIME *LINUX*

The irony of embedded or real-time *Linux* is that the founder and leader of *Linux*, Linus Torvalds, has himself indicated no great desire to add real-time capabilities to *Linux*. There is fear and uncertainty, therefore, of mainstream *Linux* ever having a perfect fit with real-time capabilities.

(See his 2004 interview, here: http://news.zdnet.co.uk/software/Linuxunix/0,39020390,39169942,00.htm). That said, *Linux* has had an explosive history in embedded and real-time systems since its debut. Almost every design these days investigates the prospect of using *Linux*, even if ultimately it doesn't end up being used.

GENERAL ISSUES IN USING EMBEDDED LINUX

Leaving aside the more technical issues of embedded *Linux*, one first has to understand a few of the basic issues concerning *Linux* itself as a software product. First and foremost, *Linux* is "free" in the same sense that many of the "free" RTOSes discussed above: free relates more to the "free" in "freedom" than to the "free" in no-cost. *Linux* is subject to the GNU GPL, or "GNU General Public License." (See discussion above for pros and cons of GPL).

So if the first issue with *Linux* is its licensing status, the second issue is that *Linux* is not primarily a "real-time operating system." If your design has any real-time requirements (whether hard or soft), this may or may not become an issue. You are thus confronted with a number of options to employ *Linux* in a project that requires real-time performance (in some aspects):

- **Modify *Linux* yourself** so that it meets the real-time requirement(s) of your design, i.e. on the buy vs. build continuum you are "building" your own RTOS based on *Linux*;
- **Purchase a hardened "real-time" *Linux*** from a vendor that specializes in this very thing; or
- **Split your design** into a "real-time" component (that is then based on a true real-time operating system) and a non-real-time component (that is based on *Linux*), and coordinate the two software operating systems (there are fortunately some vendors that specialize in just this approach).

One common pattern for some users of Linux is to end up doing the first thing – getting a "free" version of *Linux* from the Internet and then modifying it to suit their own real-time project requirement(s). Not as many actually purchase a "commercial" version of *Linux* and/or outsource the *Linux* portion of their design to some outside vendor.

Linux in this sense, therefore, is a hybrid between "building your own" RTOS and purchasing an RTOS: it is an amalgamation of both. Whether you end up with the best possible combination or some nightmarish project that has the worst of each is up to you.

APPROACHES TO USING *LINUX* IN REAL-TIME / EMBEDDED PROJECTS

APPROACH #1 – BUILD IT YOURSELF

In this approach, you consider yourself a seasoned programmer, or at least seasoned enough, to download a free version of *Linux* and modify it yourself for your project. The *Linux Kernel Archives* at http://www.kernel.org/ is the place to start. You're going to need more than just the raw kernel, so **eg3.com** indexes all sorts of information on embedded *Linux* such as overview books and online resources at http://www.eg3.com/embedded-linux.htm.

In this approach, you have many of the same negatives as in the "build your own RTOS" approach:

- **Time vs. Money.** It will take time and effort for you to modify *Linux* for your embedded project. How much is this time worth? If you pay a programmer $50 an hour, and he takes only 40 hours to modify *Linux* then you have paid $2000 for the modified embedded *Linux*; if it takes 400 hours to modify *Linux* successfully that will cost you $20,000. It depends on the complexity and needs of your project vs. the available programmer skills and their costs.

- **Time-to-Market**. *Linux* is admittedly *not* a real-time operating system, so comparing embedded *Linux* to other RTOSes that are designed for real-time projects, will using *Linux* appreciably slow your time to market? What is the real cost of this?
- **Code Maintenance and Longevity.** By definition, real-time modifications to *Linux* even those of major vendors, are not generally "maintained" by the *Linux* community – thus your project's *Linux* will to some extent be a unique code variant (or forked code, as they say) – this will need to be maintained by your or your company, just as your own in-house RTOS would be. This implies a fair amount of cost and risk exposure if the code maintainers die, get uninterested, or move to other companies. Does your project need code longevity? Will the code be reused or reemployed?
- **Performance**. *Linux* is not a real-time operating system, and any modified versions of *Linux* may suffer performance degredation. In this sense, it will be worse than an RTOS built from the ground up for your design and probably worse than a purchased commercial RTOS. What is the negative value of the performance degredation? Some vendors sell "hardened" solutions for *Linux*, but there are fees and costs involved which thus negate some of the initial cost advantages.

In addition, because of its licensing requirements, *Linux* has legal risks and ramifications that a pure in-house or commercial RTOS would not.

LINUX TECHNICAL CONS

An excellent brief white paper from **On Time Software** (http://www.on-time.com/) presents other possible technical problems with *Linux*, among them:

- **Boot Time**. *Linux* requires a standard PC Bios and may need from 30 seconds to 1 minute to boot vs. the fast boot times of true RTOSes.
- **Resource Overhead**. *Linux* has a larger footprint than true RTOSes and thus requires more hardware (and more budget for hardware) than a true RTOS might. In addition, future upgradeability of your design might be hampered if you have little hardware space or budget for upgrades.
- **Development Tools**. If you like UNIX development tools, you will be comfortable with *Linux*. If not, you may find the development tools lacking as compared with those for RTOSes like *Windows CE* or others that have a large ecosystem of graphical tools.

Most interestingly, **On Time Software** calls into question the very "business model" of the hardened *Linux* vendors by pointing out the contradiction in that they usually *charge* for technical support while most RTOS vendors provide technical support for *free* or at least at a defined cost. In this sense, the vendor benefits from the complexity of *Linux* and isn't motivated to free you from their technical support. ("On Time RTOS-32 Versus *Linux* for Real-Time Embedded Systems," http://www.on-time.com/rtos-32-versus-Linux.htm, June 2, 2006)

Another vendor that has criticized *Linux* is **Green Hills Software** (http://www.ghs.com/). In a series of 2004 polemical essays, Dan O'Dowd stirred up a firestorm of controversy by arguing that *Linux* was not a good choice for security-intensive applications. Read the whole series at http://www.ghs.com/Linux.html.

Finally, vendor **Bsquare** (http://www.bsquare.com/) calls attention to the hidden development and royalty costs that might be involved in a *Linux* choice. *Linux*, says **Bsquare**, is billed as a "no cost" operating system. This is, however, for only the kernel itself. Much of the additional intellectual property needed for a typical embedded project will add costs – web browser, media player, MP3 decoder, network stacks, digital rights management, etc. In some cases, RTOSes like **Microsoft** *Windows CE* already have these features in the RTOS (and paid for in the licensing agreement), and in addition the license fee for a basic kernel is not very much – thus negating much of the cost advantage of embedded *Linux*. In sum, you have to define in

advance **all** the features needed by your device and if *Linux* doesn't have those features, factor in the cost to build or buy these features vs. what you would spend from a commercial vendor. (*The Windows® Embedded Advantage—Comparing Windows Embedded with Linux* h t t p : / / w w w . bsquare. c o m //marketing/datasheets/licensing_windows-vs-linux-MS_ds.pdf 11 October 2007).

That said, these criticisms of *Linux* are made obviously by vendors who benefit from a non-*Linux* choice, so take them into consideration but investigate the other, positive side of a *Linux* choice as well.

LINUX PROS

The pros for embedded *Linux* are:

- **Well Known Brand.** While not real-time, *Linux* is well-known with an established "brand" – this may be important to your company's marketing efforts and managerial approval. Indeed in interviews we have learned that for some companies the fact that *Linux* is not a commercial product and thus does not **require** that the final product carry its brand - i.e., the product's RTOS and brand can be concealed from the user - can ironically be a "brand advantage" vs. certain commercial RTOSes (noteably **Microsoft**'s products) that often may require a co-branded product.
- ***Linux* Community**. *Linux* is supported by a global community of users, which allows your company to benefit from that collective knowledge. Even if your company creates its own flavor of "real-time *Linux*," you will be able to benefit from this community – new hires will at least be somewhat familiar with your code, and thus the learning curve for them may be less than it would have been if they were starting with an in-house RTOS.
- ***Linux* Applications**. This is probably one of the greatest selling points of *Linux*: there is not only a wide community of *Linux* programmers, there is a cornucopia of *Linux* applications. If your device has any need for a wide variety of applications, *Linux* offers this in a way that no other OS with the possible exception of *Windows XP*. The wealth of *Linux* applications and the vibrant *Linux* community make *Linux* the biggest **platform** in the embedded software market.

Like the free RTOSes, *Linux* has the pros that come with free -

- **Low cost**. The software is free to download over the Internet, and there are no royalties.
- **Ease of Getting Started**. Because it can be easily downloaded, you can get started quickly and easily without the fuss of licensing agreements and NDAs.
- **Well Known Tools**. *Linux* is based on the GNU tool chain and is basically UNIX. If you know UNIX, you will be very comfortable with *Linux*.

The biggest problem may be the real-time requirements of your application and whether *Linux* can fit in your resource contraint and/or meet the real-time performance requirements. In this case, you may need to consider "hardened" *Linux* alternatives available from commercial vendors.

APPROACH #2: PURCHASE A HARDENED OR EMBEDDED *LINUX*

There are a number of vendors competing for your *Linux* business. What each has done, in slightly different ways, is to take the basic *Linux* distribution and alter it to achieve real-time *Linux* performance. Here are a number of leading *commercial Linux* vendors and their product offerings:

General *Linux* Vendors These vendors focus on meeting the requirement needs of a wide range of target applications.

- **MontaVista Software** (http://www.mvista.com/). Offers a number of varieties of "embedded or real-time *Linux*" optimized for vertical such as embedded devices, consumer, communications, or mobile. *Especially strong in the communications vertical.*
- **LynuxWorks** (http://www.lynuxworks.com/). Offers "BlueCat *Linux*" hardened for real-time, with a variety of development tools and board support packages (bsp's). Emphasis is also on upward compatibility to the company's proprietary RTOS, *LynxOS*.
- **TimeSys** (http://www.timesys.com/). Works closely with chip makers to offer specific board support packages, and defined *Linux* distributions. Customers sign up for *LinuxLink*, which is an update stream for their particular "flavor" of embedded *Linux* as well as access to a "*Linux* community" hosted by **TimeSys**.
- **WindRiver Systems** (http://www.windriver.com/). A newcomer to the *Linux* market, **Wind River** offers a hardened *Linux* that is integrated with their development tools, especially *Wind River Workbench*. *Especially strong in the communications vertical.*

The murkiness with embedded *Linux* is that while, on the one hand, these vendors appear to be selling you a stand-alone software product – a *Linux* RTOS, they are, on the other hand, offering you a longer term stream of "services" and/or "upgrades" to *Linux*. The model is generally a hybrid of an off-the-shelf software with a stream of services. In addition, while royalties are not charged per se (this would violate the GPL), there is an implicit embedding of cost in the per deal negotiations. The business model in this sense is not that far from that of the true commercial RTOS vendors. Since there may be no standard published pricelist, you should expect some detailed negotiations on a per deal basis for the cost of your "embedded *Linux*" and/or its affiliated services. There are, in short, many *de facto* ways to charge royalties without charging royalties and to charge for "free" software just as much as to charge for "commercial" software.

APPROACH #3 – GO WITH A LINUX / HARDWARE SOLUTION

Similarly to the commercial RTOS market, there are embedded *Linux* non-profits and vendors as well that create specialized or "niche" versions of *Linux*, optimized for specific vertical applications. One solution is to go with a hardware company that provides a version of Linux already pre-configured to support their hardware. **Concurrent Computer Corporation** (http://www.ccur.com/), for example, offers *RedHawk™ Linux*, a *Linux* variant optimized for simulation, data acquisition, and industrial systems control. **Empower Technologies** (http://www.empowertechnologies.com/) offers *LinuxDA*, a variant focused on consumer products. And **Softier**'s *MediaLinux OS* is a port of the *uCLinux* Kernel and additional *Linux* components optimized to run natively on TI 64x DSPs (http://www.softier.com/mediaLinux_os.asp). Finally, **Performance Technologies** (http://www.pt.com/) offers their *NexusWare™* embedded Linux, which is essentially Linux optimized for their hardware boards and other products (http://www.pt.com/products/nexusware.html).

In each case, these are simply efforts to make *Linux* real-time *and* optimize it for a specific application area. If your application area happens to enjoy a company that has expended effort on it, so much the better for you.

APPROACH #4 – DOWNLOAD A UNIVERSITY OR NON-COMMERCIAL REAL-TIME *LINUX*.

Linux, popular as ever with the research and university community, has spawned a number of non-commercial efforts at improving its real-time applicability. In each case, investigate the website and organization providing the *Linux* distribution. In some cases, they allow for commercial application development with various fees or restrictions; in others, these are solely for university or learning development. That said, there may be some benefit in researching

university embedded *Linuxes* because by learning how they optimized *Linux*, you might better be able to create your own "personal" embedded *Linux*. Here are some of the more popular:

- **Embedded Debian** (h t t p : / / lists.debian. o r g/debian-embedded/). Discussion about improving *Debian* for use with embedded systems, including building cross-compiler toolchains, cross-compiling packages, creating and updating system images, using alternate libraries, compile-time configuration of packages, etc.
- **ELKS: The Embeddable *Linux* Subset** (h t t p : / / elks.sourceforge.net/). A project to build a small kernel subset of *Linux* (which will provide more or less UNIX V7 functionality within the kernel) that can run on machines with limited processor and memory resources.
- ***ucLinux*™** (h t t p : / / w w w . uclinux. o r g/). As an operating system this variant includes *Linux* kernel releases for 2.0 2.4 and 2.6 as well as a collection of user applications, libraries and tool chains.
- ***VMELinux*™** (h t t p : / / w w w . vmelinux. o r g/). A port of *Linux* optimized for VMEbus systems.
- **Red *Linux* Project** (h t t p : / / linux.ece.uci.edu/RED-Linux/). Billed as an "open kernel for real-time and embedded systems.

There is a pretty complete list of real-time *Linux* projects at the **Real Time *Linux* Foundation** (h t t p : / / w w w . realtimelinuxfoundation. o r g/). Also consult Appendices D and E on available *Linux* distributions, as well as **eg3. c o m** 's *Linux* coverage at h t t p : / / w w w . eg3. c o m /embedded-linux.htm.

The problem with many of the free, or university real-time *Linux* distributions is because they are maintained by "volunteers" or "university people" they are only sporadically cared for and updated. Many are very out-of-date, and the idea of depending on them for a commercial project is questionable. If you are interested in *Linux* and can't go with a major vendor, you are probably better off rolling up your sleeves and creating your own specific version of embedded or real-time *Linux* from h t t p : / / w w w . kernel. o r g/.

***Linux* Tools.** Given the popularity of *Linux* for embedded and real-time systems and the fact that there is no single dominant embedded distribution, one would think that a smart vendor had created some sort of management tool for taking a general *Linux* distribution and optimizing it into a real-time distribution. No such luck. However, there are a number of tools designed to make the task of formulating an embedded *Linux* easier.

Scratchbox (h t t p : / / w w w . scratchbox. o r g/), for example, is cross-compilation toolkit designed to make embedded *Linux* application development easier. It also provides a full set of tools to integrate and cross-compile an entire *Linux* distribution. Currently only ARM and x86 targets are supported.

RTAI – the RealTime Application Interface for *Linux* from **DIAPM** (h t t p : / / w w w . rtai. o r g/). The Realtime Application Interface consists mainly of two parts: a patch to the *Linux* kernel which introduces a hardware abstraction layer, and a broad variety of services which make realtime programmers' lives easier.

RTOS Survey Results

How do real users of RTOSes feel about their choice? In October, 2007, eg3.com sent out a survey request to the 37,000+ registered subscribers of e-clips, and received 353 detailed responses - from students to RTOS first-time buyers to experienced users with many years experience. Here is an analysis of the "before" and "after" RTOS experience by knowledge group.

- RTOS Survey Demographics
- RTOS Experience & Vertical Industries
- Business Factors in the RTOS Decision
- In-House RTOS Users: Before & After
- Commercial RTOS Users: Before & After
- *Linux* Users: Before & After
- Open Source (Non-*Linux*) Users: Before & After
- Comparison of Different RTOS Groups
- Anecdotes

Insiders' Guide Embedded RTOS: RTOS Survey Results

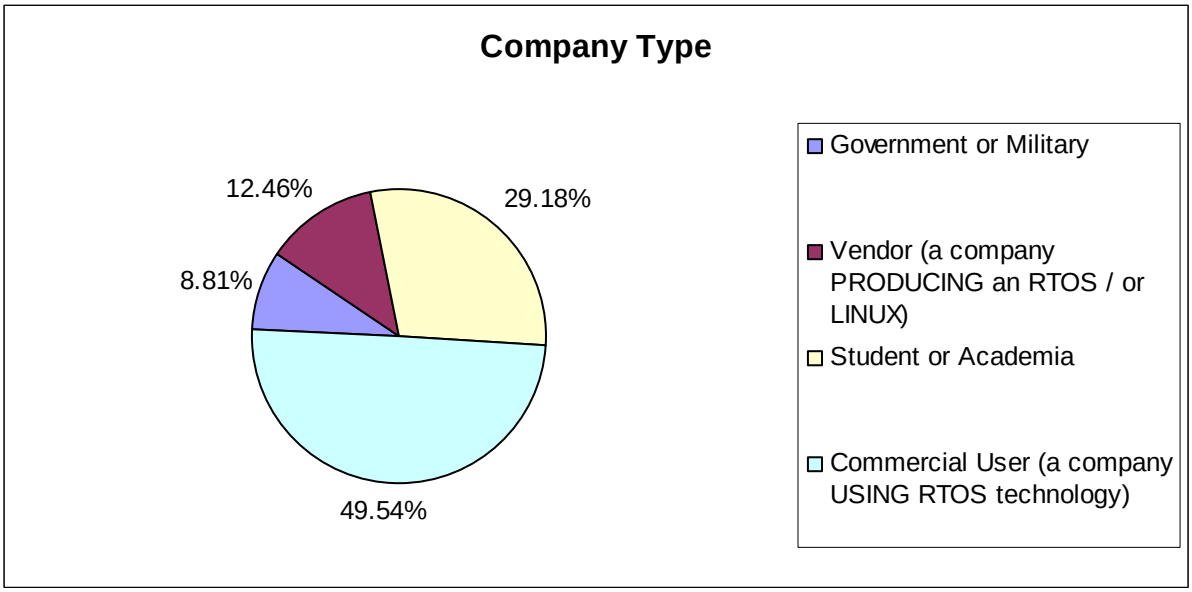
RTOS SURVEY DEMOGRAPHICS

eg3.com is well positioned to analyze how the embedded and RTOS community makes the choice of an RTOS. We have over 37,000 registered subscribers to e-clips (our news alert service by keyword), and a strong demographic of designers who employ or might employ RTOSes in their current or next embedded design. In October, 2007, we sent out a survey request and received 353 detailed responses.

In a few cases, we have also used survey data from our summer, 2007, engineering survey which had 1,445 respondents.

RTOS GROUPS

The Internet is an open medium, and thus we reached many students and RTOS vendors as well as commerical or military designers who already use and/or are investigating an RTOS choice. We had this proportion of responses by company type:



For most questions of the survey, we excluded both the student group and the vendor group because we were most interested in the opinions of real engineers who have already used an RTOS or are considering one. As is often true with embedded systems surveys, this makes the "n" of total respondents rather small. The reader is therefore cautioned that the degree of survey error is relatively high, and it increases as the respondent number for a question decreases.

That said, the survey provides a window into how different groups perceive the RTOS choice. For the analysis that follows, we designate groups as follows:

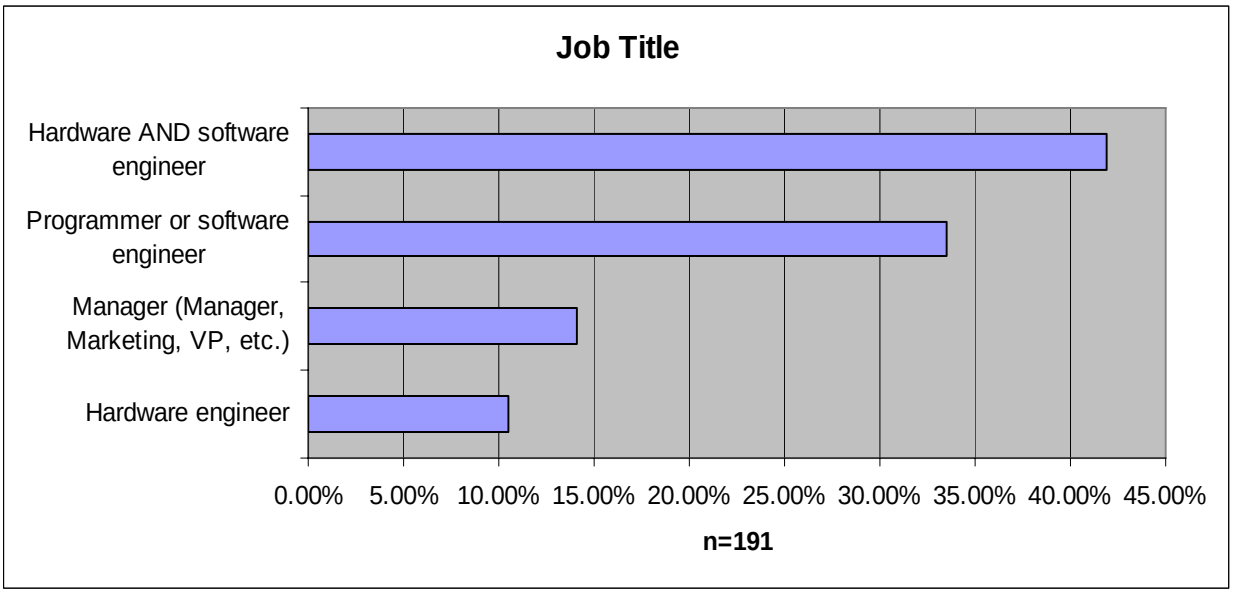
Group:	Description:
Total group (n=192)	Everyone who took the survey, *except* students and RTOS vendors. **All groups below *exclude* students and RTOS vendors.**
Commercial group (n=105)	Users who indicated interest and/or experience

	in commercial RTOSes.
Linux group (n=77)	Users who indicated interest and/or experience in embedded *Linux*.
Open Source Group (n=72)	Users who indicated interest and/or experience in non-*Linux Open Source* RTOSes.
In-House Group (n=62)	Users who indicated interest and/or experience in in-house RTOSes ("roll your own")
First-time group (n=29)	Users considering an RTOS for the first time.
Newby group (n=33)	Users with 0-1 years RTOS experience
Expert group (n=110)	Users with more than 1 years RTOS experience
Non-expert group (n=62)	Users with <1 year or no experience (i.e., first-time + newby group)

Note that a person can be a "member" of the different RTOS groups (Commercial, *Linux*, Open Source, and/or In-house) because he or she might have experience with any or all. Users belong, however, to only one expertise group measured by years of experience.

JOB TITLE

What sorts of engineers answered the survey? As with most **eg3.com** surveys we found a mixture of hardware, software, and management in the current RTOS survey.



As is typical for embedded systems, hardware and software are tightly interconnected with 41.88% indicating both hardware and software, followed by 33.51% indicating software only. The RTOS choice is clearly intertwined with the hardware choice, and vice-versa.

RTOS EXPERIENCE & VERTICAL INDUSTRIES

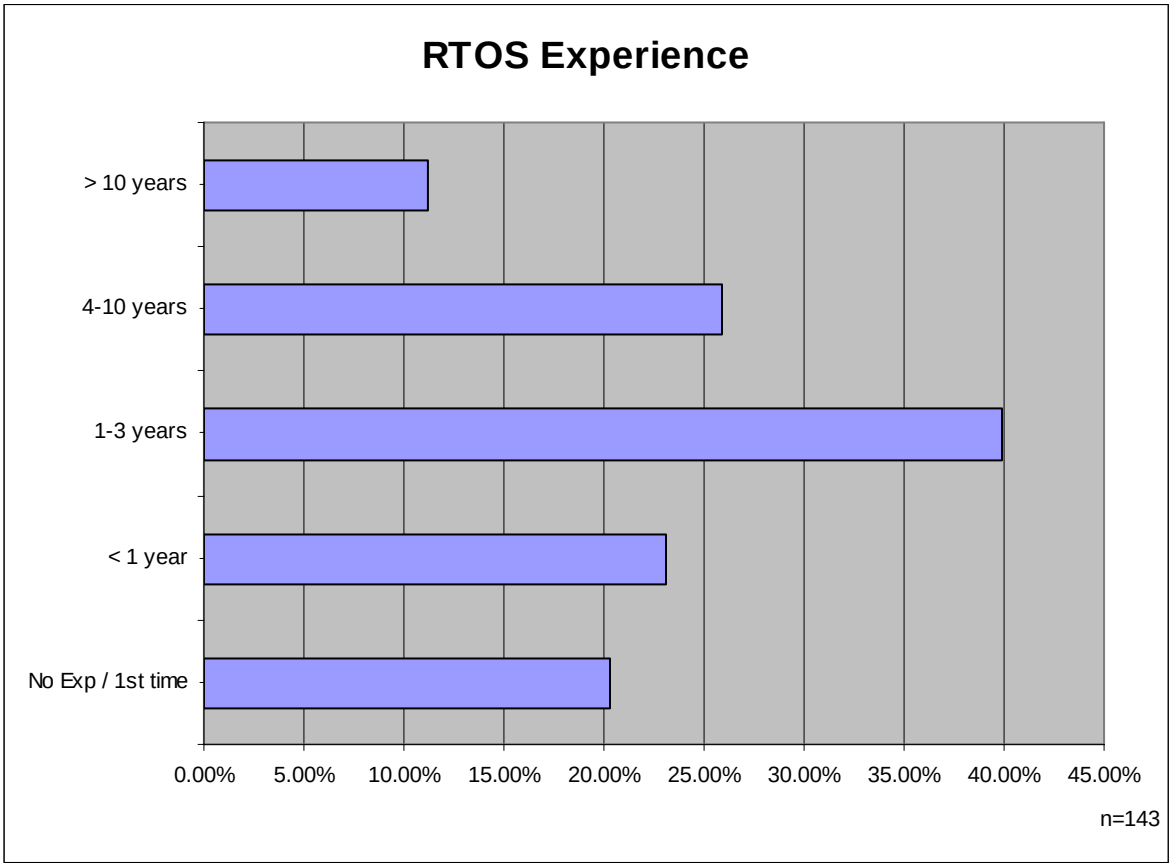
Speaking on a general level, how experienced are users with real-time operating systems? What types are most well-known? What types are new? What vertical markets pull designers into having to select an RTOS? And, most importantly, what factors do designers feel are most important in selecting an embedded OS?

RTOS USE

First of all, how many embedded designs actually use real-time operating systems? To answer this question, we turned to a larger **eg3.com** survey conducted summer, 2007. In that survey, we asked 1,445 engineers, "Are you using a real-time (deterministic) operating system in your current design?" 897 engineers answered the question, and of that 51.5% said "yes" and 48.5% said "no." Looking at the entire universe of embedded systems, the penetration of RTOSes is thus about 50%. In most cases in which no RTOS is used the reasons are generally that the device resource limitations are too small to allow for a full-blown RTOS.

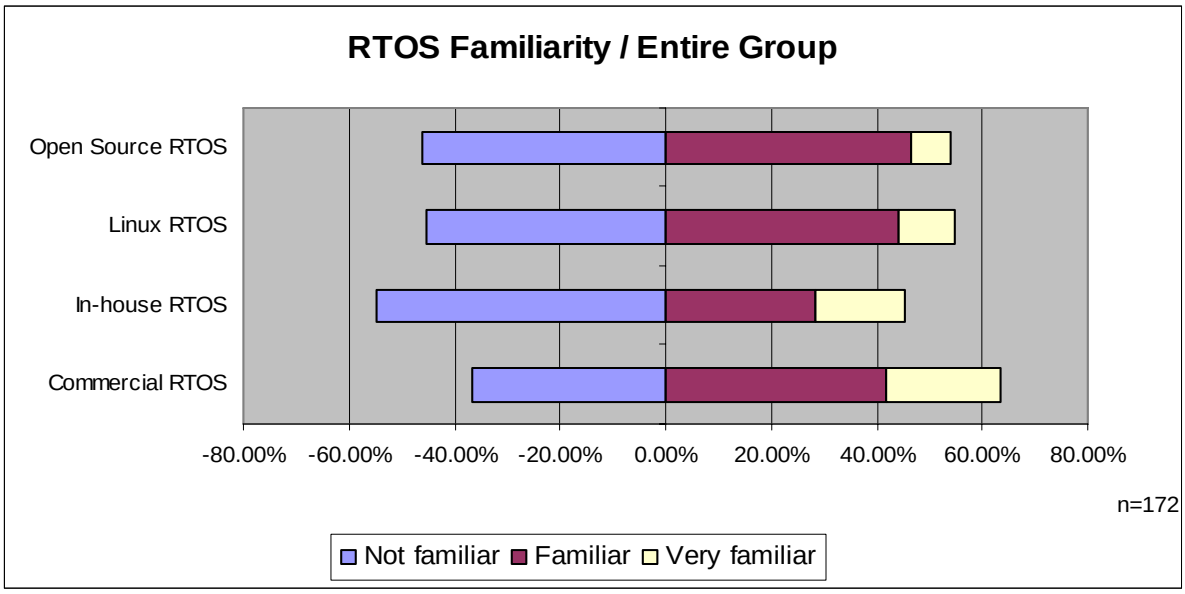
RTOS EXPERIENCE

Of those that use an RTOS, how experienced are they with RTOSes in terms of years?



Almost 40% have from one to three years RTOS experience, while a good percentage (11.19%) have more than 10 years RTOS experience. 20.28% of our sample was considering an RTOS for the first time. Their value to the survey is the fresh perspective of the innocent - developers whose expectations have not yet been affected by the real, practical trade-offs of the embedded software experience.

What about experience with each RTOS type? While we polled on specific types later in the survey, we also asked a general question about experience with our four major types of RTOSes - Open Source (non-*Linux*), *Linux*, In-House, and Commercial.



There are not huge differences in RTOS familiarity. The most important divergence is that 21.51% indicted that they were "very familiar" with commercial RTOSes vs. 10.47% for *Linux* and 7.56% for Open Source.

The first-time group, in contrast, was **most** familiar with non-*Linux* Open Source at 34.48% either "familiar" or "very familiar," followed by *Linux* exposure at 27.59%. This probably reflects the penetration of Open Source into the university and teaching environments, as well as the ease of free downloads of Open Source software.

> **Take away.** *Linux* and Open Source RTOSes are easy to get your hands on - hence there is greater familiarity with them among students and the younger generation.

VERTICAL INDUSTRIES

What drives developers to use an RTOS? One factor may be the type of end application or "vertical industry" for which a design is intended. Here is a chart showing the respondent percentage indicating each vertical industry for which they are employing or are considering employing a real-time operating system:

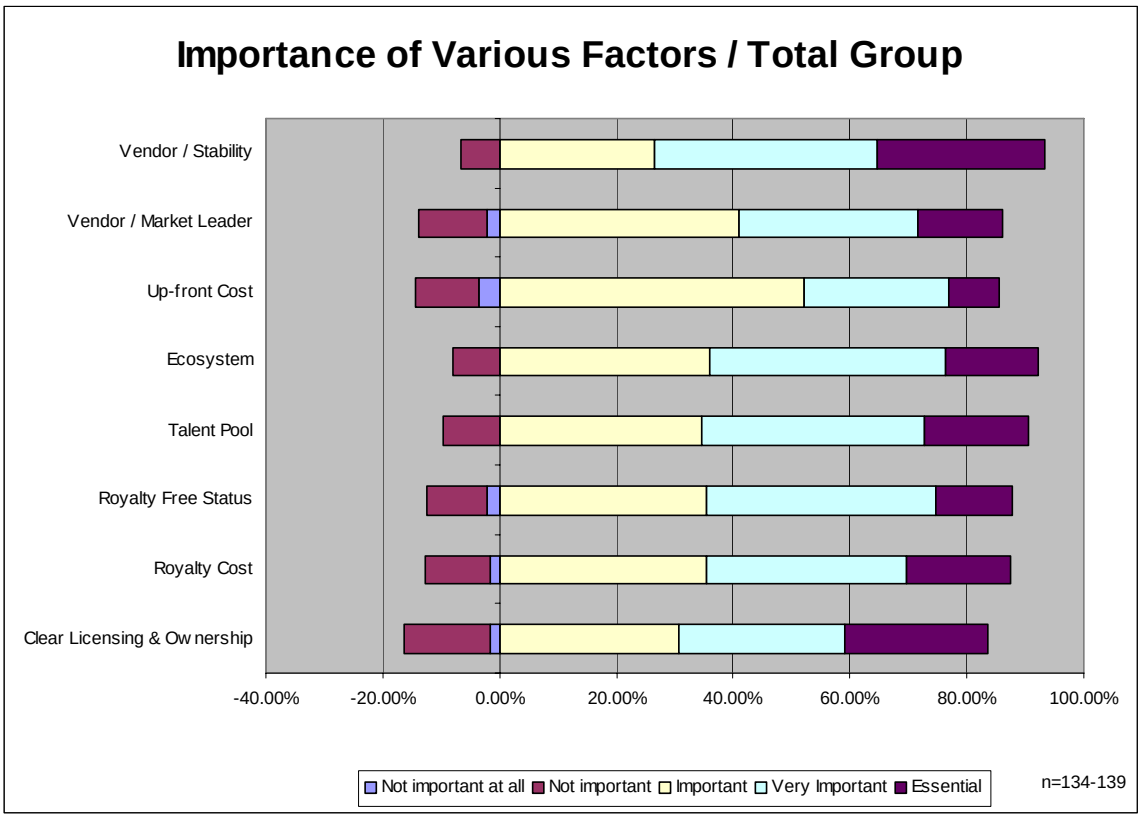
Vertical Industry:	Group:				
	Total	Commercial	*Linux*	In-House	Open. Source
n=	163	105	77	62	72
Automotive and/or Transportation	24.54%	28.57%	29.87%	25.81%	29.17%
Computer peripherals	21.47%	19.05%	19.48%	24.19%	25.00%
Consumer electronics	30.67%	29.52%	35.06%	27.42%	33.33%
Industrial automation/process control	37.42%	40.00%	41.56%	41.94%	38.89%
Medical	14.72%	19.05%	12.99%	12.90%	15.28%
Military and/or Avionics	25.15%	24.76%	25.97%	22.58%	25.00%
Mobile / Wireless (cell phone, pda, mobile devices)	36.81%	39.05%	37.66%	32.26%	37.50%
Networking, Communications, and/or Telecom Infrastructure	40.49%	40.00%	38.96%	41.94%	41.67%
Test and measurement	18.40%	20.00%	24.68%	22.58%	20.83%
Green: RTOS more likely than average (+ 3%) Red: RTOS less likely than average (- 3%)					

This can be interpreted to mean that "Commercial" is more likely to be used in Mobile/Wireless for example (39.05% vs. 36.81% for total group) vs. In-house for Mobile/Wireless (32.26% vs. 36.81%). It shows *Linux* has relative strength in automotive, consumer, industrial, and test & measurement. In-house, in contrast, is weak in consumer, military, and mobile but relatively strong in computer peripherals, industrial, and test & measurement.

> **Warning**. Relative "popularity" should not be construed to mean that one RTOS type is a better choice than another for a particular vertical industry. Even though an RTOS is popular, it may still be technically inferior to an alternative. There are, after all, many reasons for popularity!

BUSINESS FACTORS IN THE RTOS DECISION

Choosing an RTOS is as much a business decision as a technical one. Royalty issues, vendor stability, up-front costs, tools and other factors are important to your choice. How does each group rank the various business factors in terms of a good RTOS choice? We hoped to tease out the relative importance of factors, but what we found is that asked *a priori* developers basically indicate that "everything" is important. Pity the poor commercial RTOS sales representative, because his customer wants everything:



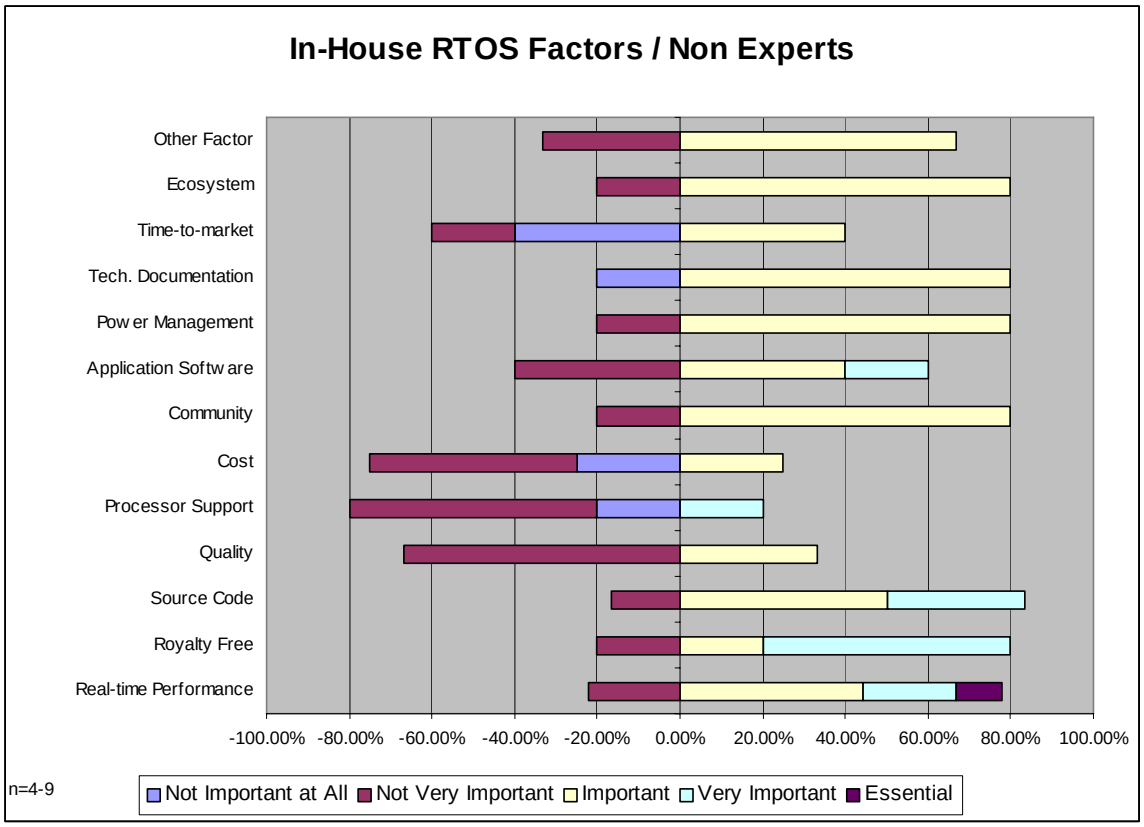
Comparing the subgroups to each other, first-time users seem slightly more likely to think that "up front" costs, "royalty free" status, and "clear licensing" (11.11%, 11.11% and 5.56%) were "not important at all." This contrasts with the expert group that indicated at 0%, 0% and 1.06% as "not important at all" for these factors, and 2.06% indicating "market leader" was "not important at all." First time users might not realize the importance of costs and licensing, while experts are more likely to be unimpressed at a vendor's "market leadership." But, that said, there were not really significant differences among any groups - everything is important.

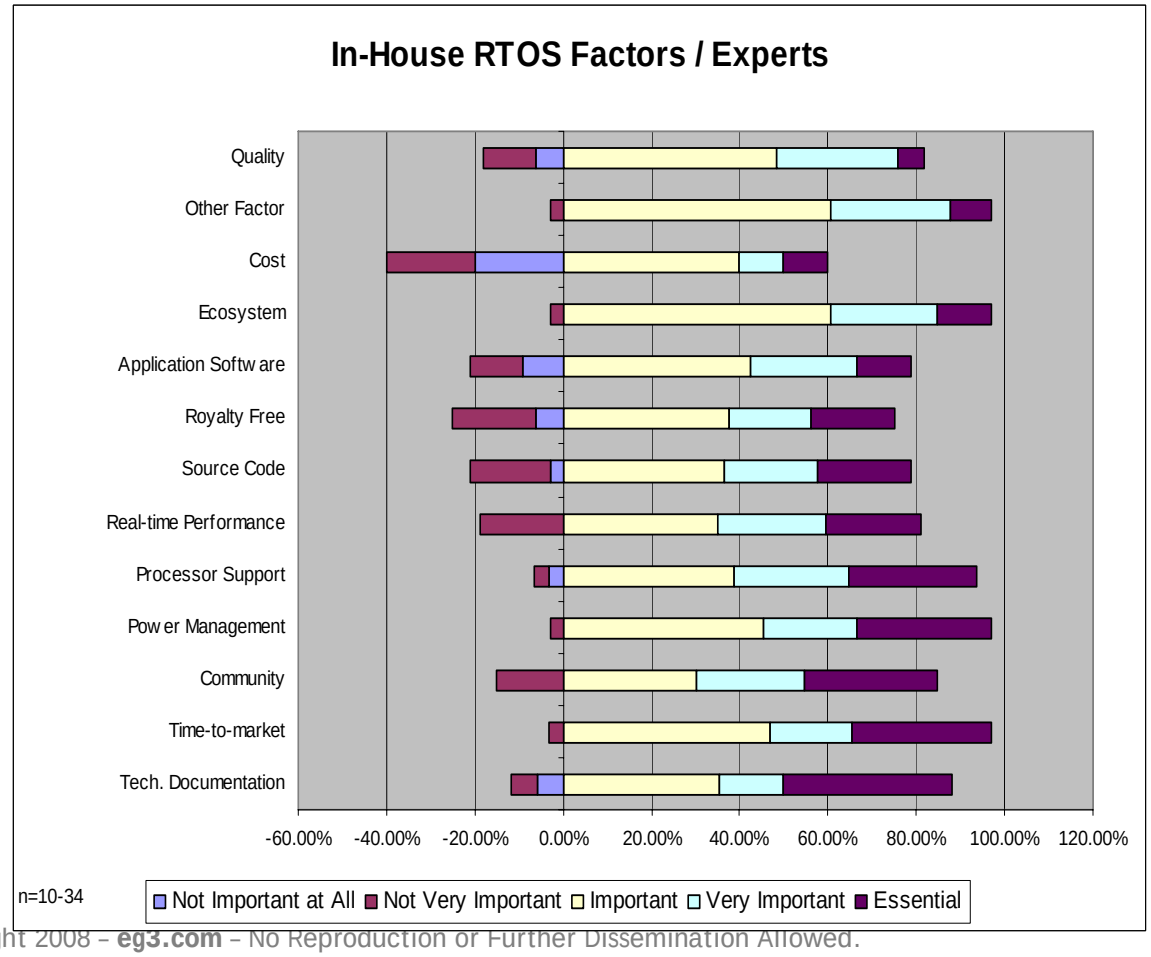
IN-HOUSE RTOS USERS

Sixty-two respondents were users of "in-house" RTOSes. These might range from a ground-up "roll your own" RTOS project to the use of a proprietary, in-house RTOS at a major OEM - one created by others in the company or before one's employment. In any case, this RTOS choice is not one that is commercially available or benefits from a "user community" outside the company.

What factors encourage this choice? How do in-house users feel "before" and "after" their project towards their in-house RTOS.

First of all, let's look at the factors identified by "non-experts" as important **before** their design project. And compare this with "experts" who have experience with in-house RTOSes:



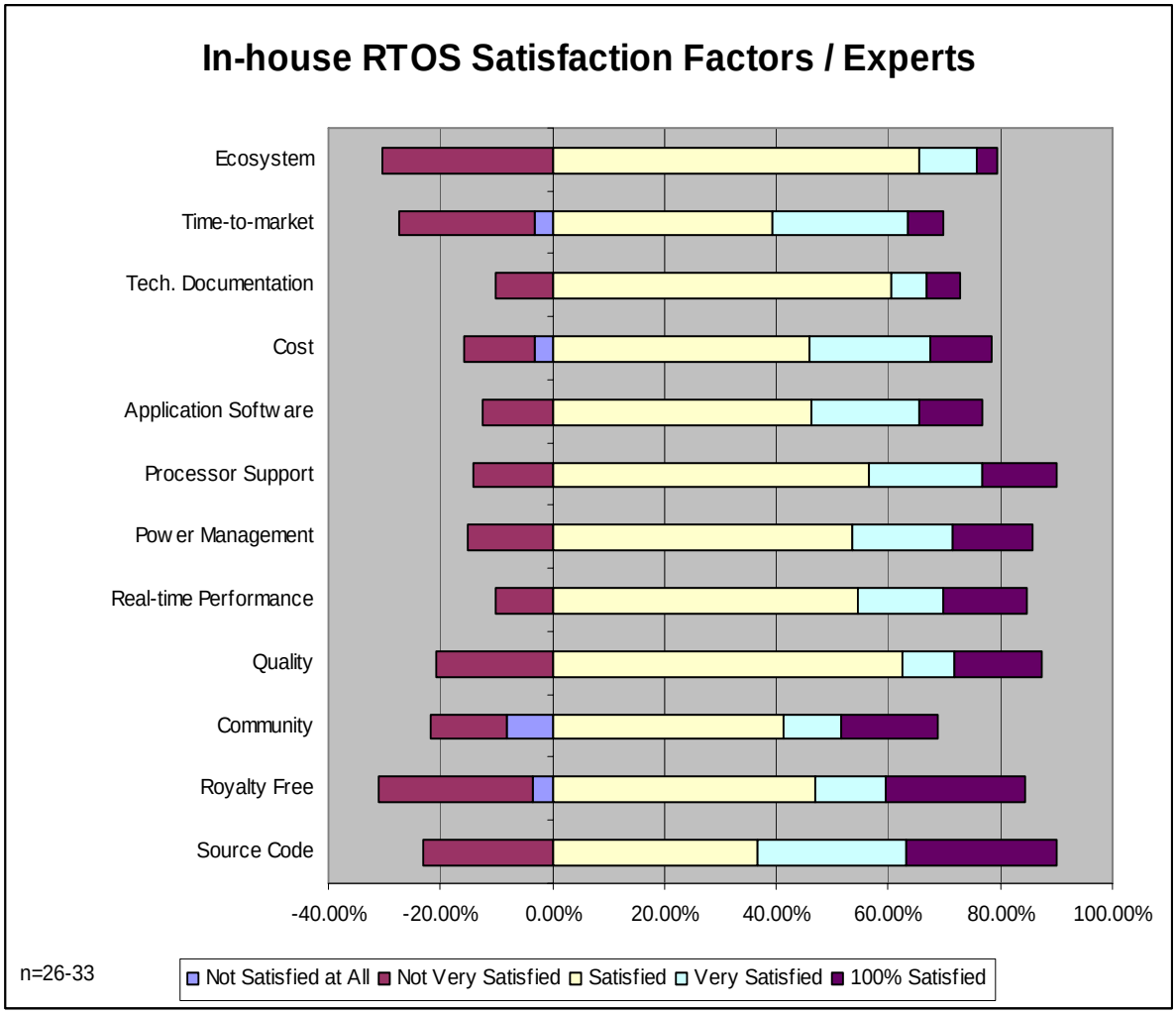


These graphs are ranked in order of those marking a factor "essential." Using that parameter, some differences are striking:

- Non-experts rank "real-time performance" as their second most important factor, whereas experts rank it sixth.
- Non-experts rank "technical documentation" as their fourth from last factor, whereas experts rank it first.
- Many non-experts rank "processor" and "ecosystem" relatively low, whereas very few experts do so.
- Non-experts rank "time to market" third from bottom, whereas experts rank it second from the top.

One suspects that experts realize that - especially for an in-house RTOS - having good documentation and friends or colleagues with experience with the RTOS might mean that a project can meet its "time to market" deadline. In other words, the lonely nights and weekends spent with an RTOS will be easier if there is good documentation and some sort of "technical support" ecosystem. Concerns about royalty-free, source code, and true real-time importance may fade to the background once the project is begun.

How satisified is the In-house Group, after use? Here is data from those most likely to know - the experts, who have real RTOS experience of more than one year:

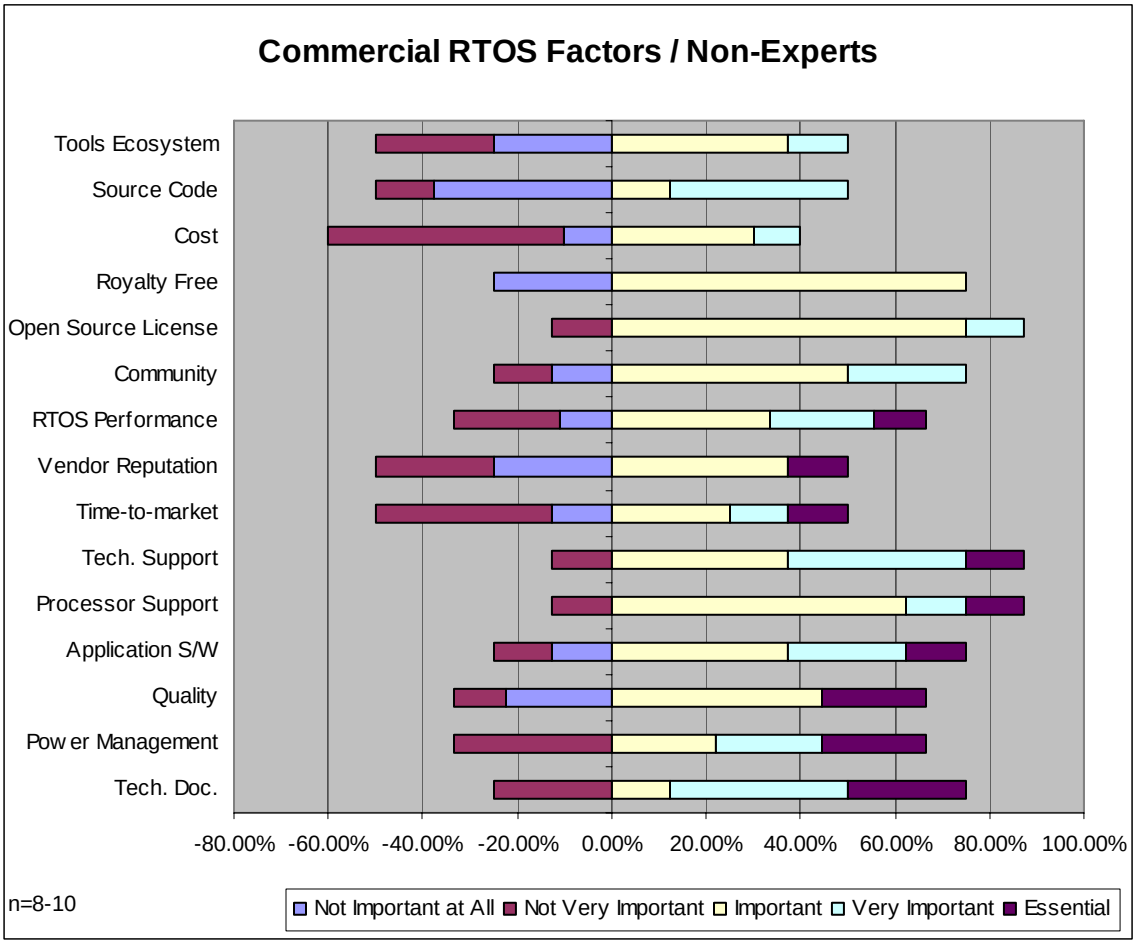


Time-to-market is an interesting compare/contrast issue. It is ranked high in importance yet receives a large percent rating their experience as "not very satisfied." This confirms the conventional wisdom that time to market is a weak spot for in-house RTOSes. The ecosystem around the in-house RTOS also scores poorly in terms of satisfaction. Building your own RTOS means to some extent going it alone - so there is little, if any, community to help you.

COMMERCIAL RTOS USERS

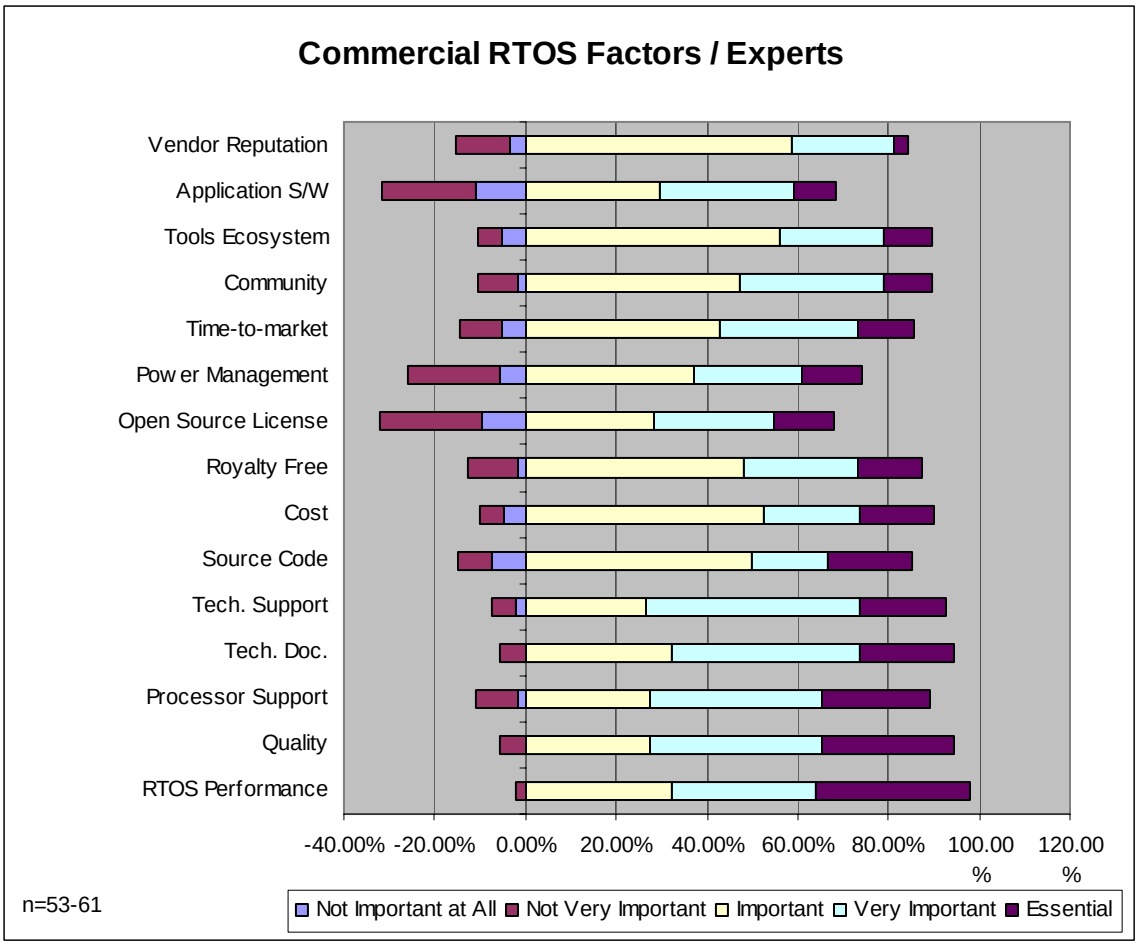
Many users of "roll-your-own" or "in-house" RTOSes dream about transitioning to a fully supported commercial RTOS. How did this user group perceive their "before" and "after" experience?

Here are how non-experts ranked various factors "before" their decision to embark on a commercial RTOS:



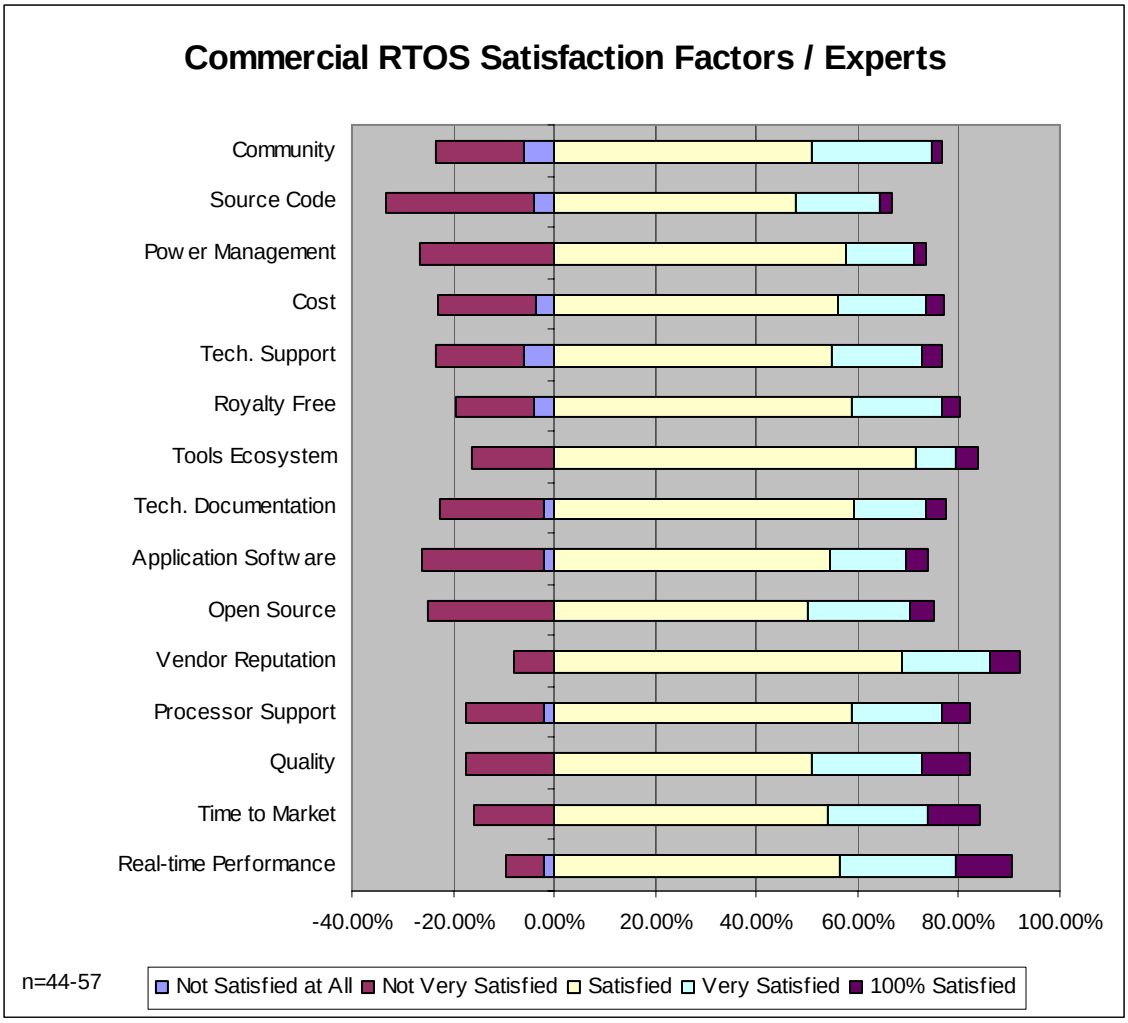
Generally speaking, technical documentation, power management, and quality were the three most important factors. Ecosystem, source code, and cost were the least important.

Here is the ranking by the expert group for commercial RTOSes:



Experts ranked RTOS real-time performance, quality, and processor support as the most "essential" factors and vendor reputation, application software, and the tools ecosystem as the least. Not surprisingly, "open source" license had a high percentage of people indicating that that was not important; a predictable result since only a few commercial RTOS vendors have anything like an open source license.

How satisified were users of commercial RTOSes? Let's look to the experts.



Comparing commercial RTOS users to in-house RTOS users, the data hints of a striking difference. Whereas in-house users were often *most dissatisified* with their more important factors (e.g., time-to-market), commercial RTOS users were *most satisified* with their most important factors (e.g., real-time performance and quality). Indeed, commercial RTOS users scored "time-to-market" at the second highest placement as ranked by those "100% satisifed." Low ratings went to community, source code, and power management.

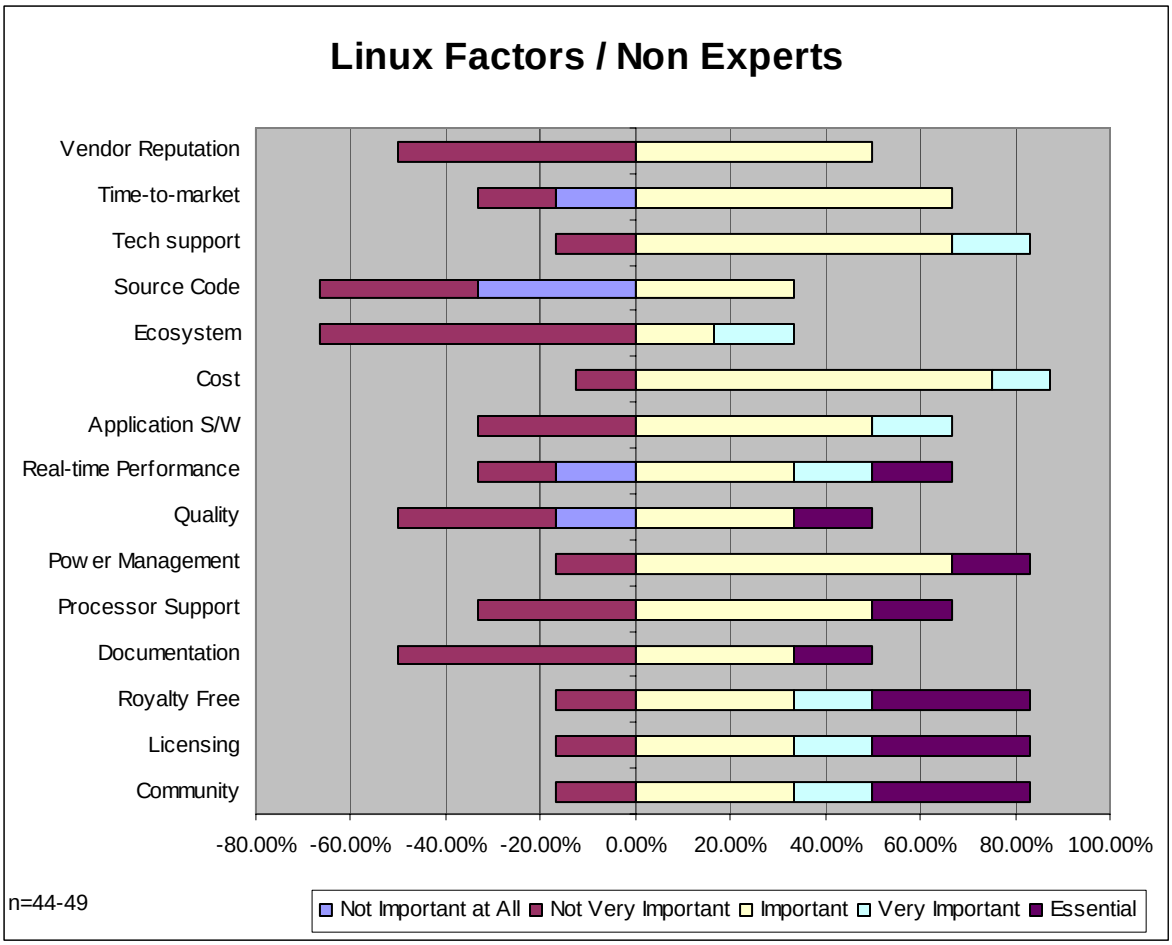
Take away. The appeal of "building your own RTOS" may fade away in a business environment, especially where "time to market" can spell the difference between project success or failure. *Commercial RTOSes excel at time to market.*

LINUX USERS

Linux is now a viable choice for many embedded applications, having gained a strong foothold in the telecommunications infrastructure vertical and now penetrating the mobile space. Vendors such as **MontaVista, Wind River,** and **TimeSys** offer commercial versions of *Linux* and

extensive support services. *Linux* has received more media buzz than Paris Hilton. But how does *Linux* really fare? What is the "before" and "after" experience?

Here is the "before" picture in terms of *Linux*:

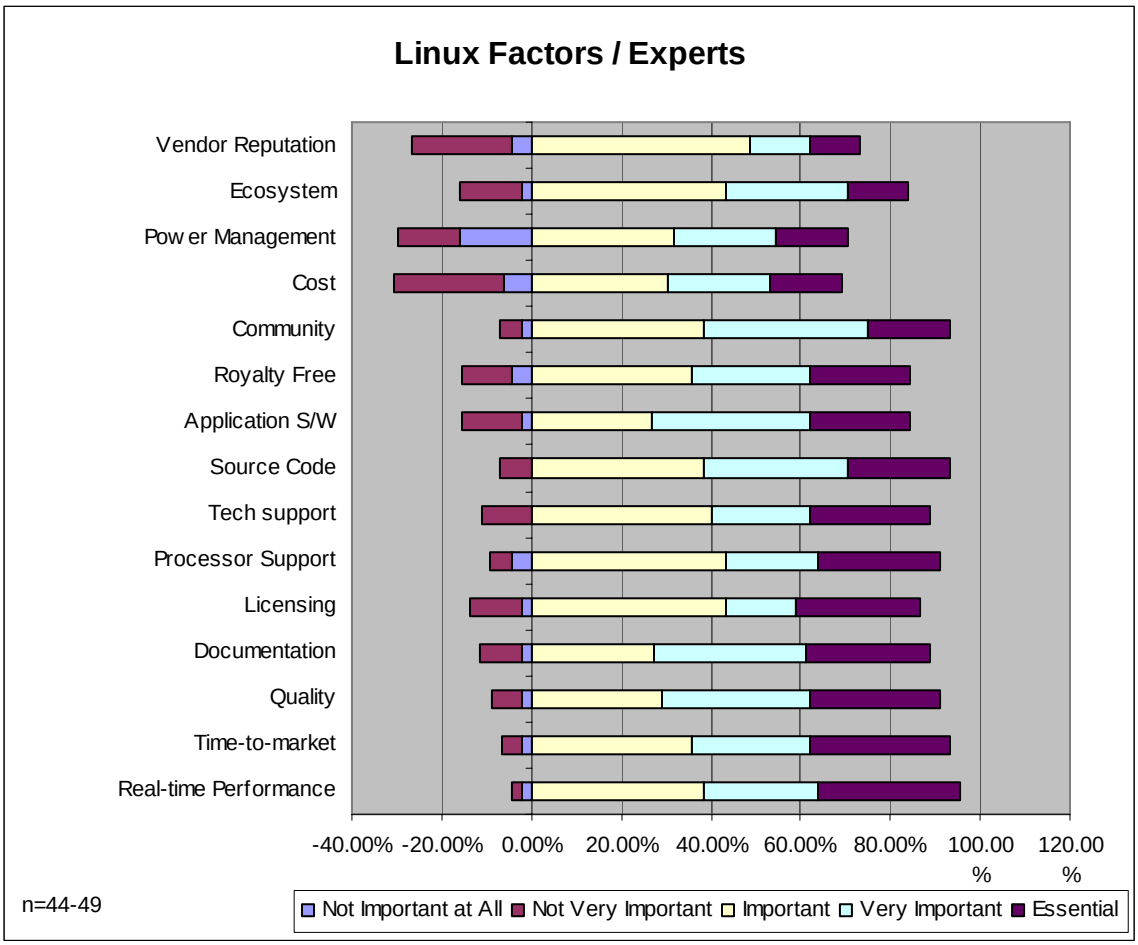


Not unexpectedly, the vaunted *Linux* community is indicated as the most important factor along with licensing and *Linux*'s royalty free status.

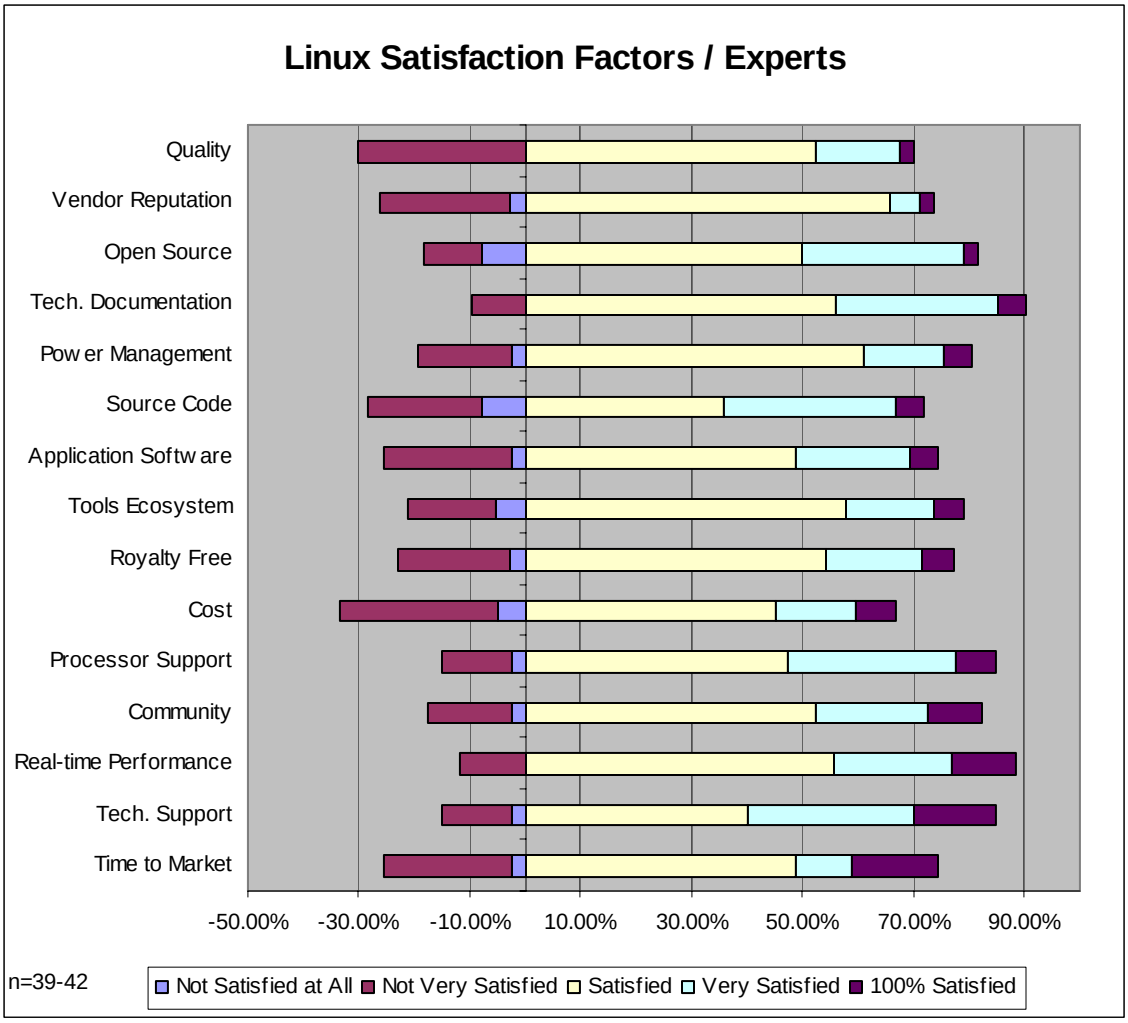
Turning to the expert group, the striking thing here is that the *Linux* community drops to eleventh place, and the most important factors for the expert group are real-time performance, time-to-market, and quality. None of these are factors that are particularly unique to *Linux*, a fact may indicate that the experts choose *Linux* as they choose other RTOSes for its technical characteristics and not its much-hyped community or "open source" spirit! Surprisingly, "ecosystem" ranks near the bottom of the list as well.

> **Take away**. If you are considering *Linux* look behind the *Linux* community and open source buzz at its technical characteristics. These may be more important to your choice than the "community."

Here is the chart of how experts perceive the relative importance of various factors in terms of *Linux*:



How do *Linux* expert users rate *Linux* after an RTOS project? Here is the "after" picture:



Again, not unlike a commercial RTOS, *Linux* experts are satisified with the time to market, technical support, and even the real-time performance of embedded *Linux*. They are disproportionately not satisified with software quality, vendor reputation, and "open source" licensing. Fortunately, the community is ranked fourth in terms of 100% satisfaction.

But how would *Linux* compare head-to-head with a commercial RTOS solution?

To find this out we calculated "satisfaction" scores for each RTOS type. By satisfaction score, we have given negative values to the percent "not very satisified" and "not satisfied" and positive values to the percent "satisified," "very satisfied" and "100% satisified." By adding these up, a positive (or negative) score results, which allows us to compare one RTOS choice to another. Below we produce a a chart comparing the satisfaction scores of *Linux* vs. Commercial RTOS.

Factor	Satisfaction Score		
	Commercial	*Linux*	Difference
Application Software	47.83%	48.72%	0.89%
Community	52.94%	65.00%	12.06%
Cost	54.39%	33.33%	-21.05%
Open Source	50.00%	63.16%	13.16%
Power Management	46.67%	60.98%	14.31%
Processor Support	64.71%	70.00%	5.29%
Quality	64.71%	40.00%	-24.71%
Real-time Performance	81.13%	76.74%	-4.39%
Royalty Free	60.78%	54.29%	-6.50%
Source Code	33.33%	43.59%	10.26%
Tech. Documentation	55.10%	80.49%	25.39%
Tech. Support	52.94%	70.00%	17.06%
Time to Market	68.00%	48.72%	-19.28%
Tools Ecosystem	67.35%	57.89%	-9.45%
Vendor Reputation	84.31%	47.37%	-36.95%

(We have highlighted in green those factors for which one choice exceeds the other by 10%.)

The data implies that commercial RTOS users are more satisfied with the cost, quality, time-to-market, and vendor reputation than their *Linux* counterparts. *Linux* users are more satisfied with the *Linux* community, open source licensing, power management, source code availability, technical documentation and support.

Take away. If a particular factor is important to you, look for the green markers above to indicate a relatively higher satisfaction score from real users.

PARADOXES

There are a few paradoxes here. One, commercial RTOSes outscore *Linux* in terms of cost satisfaction. This is probably due to the "hidden" costs of *Linux* - the sweat equity spent on *Linux* code vs. a commercial RTOS, for example. The irony is that *Linux* is "free" yet its users are less satisified about cost. For the commercial vendors, in contrast, the warning is on technical support and documentation. Here *Linux* outscores the commercial products even though its technical support and documentation are largely community driven.

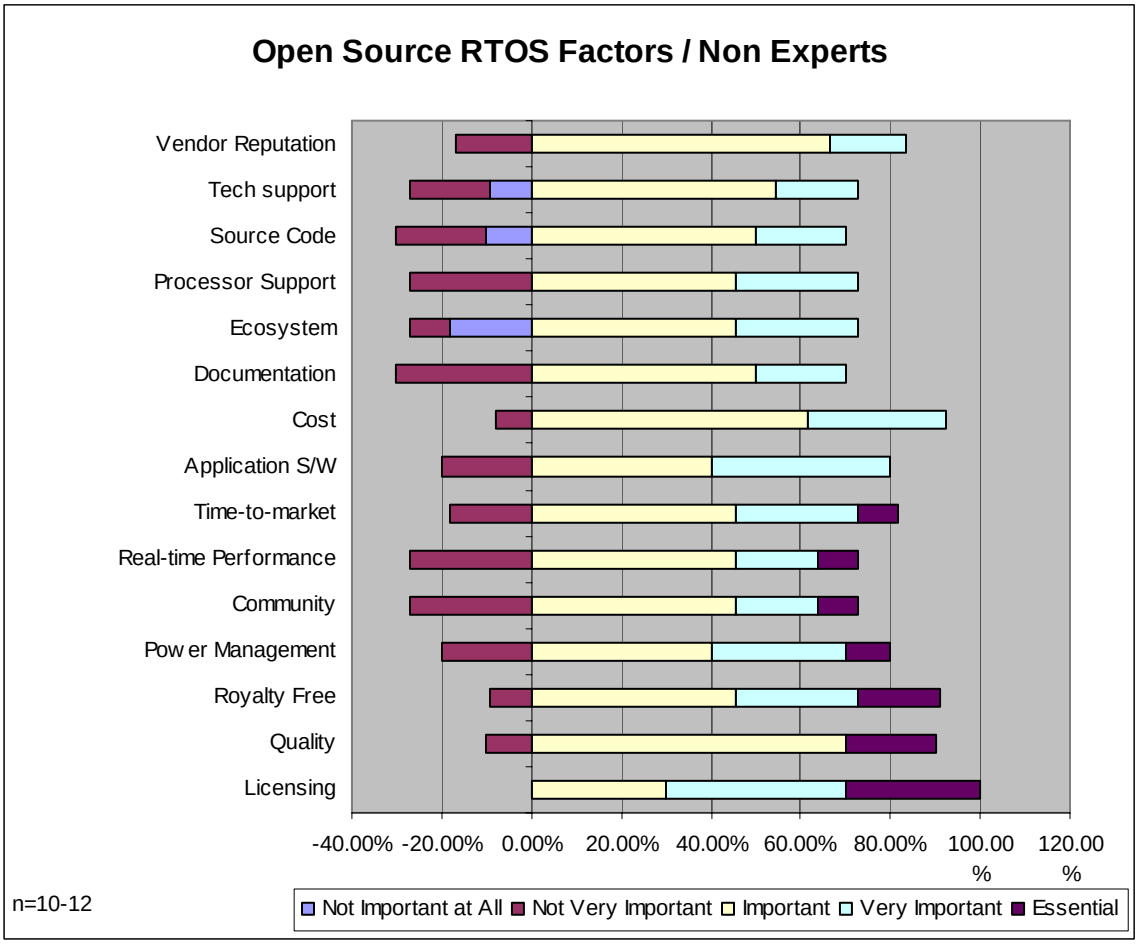
Another (implicit) paradox is that a user's satisfaction depends on his preconceived standard. If, for example, *Linux* users had a relatively low standard on "power management" and found *Linux* to perform better than this standard, they will indicate a high "satisfaction score." Contrariwise, if commercial RTOS users had a high standard for power management and the RTOS did not live up to this (high) standard, they will give a low "satisfaction score" to the power management capabilities of the RTOS.

Warning. The relative high, or low, satisfaction score of *Linux* vs. a commercial RTOS does not indicate that the **actual** performance of one is better than the other. Rather, it indicates a pleasant (or unpleasant) surprise in terms of the developer experience.

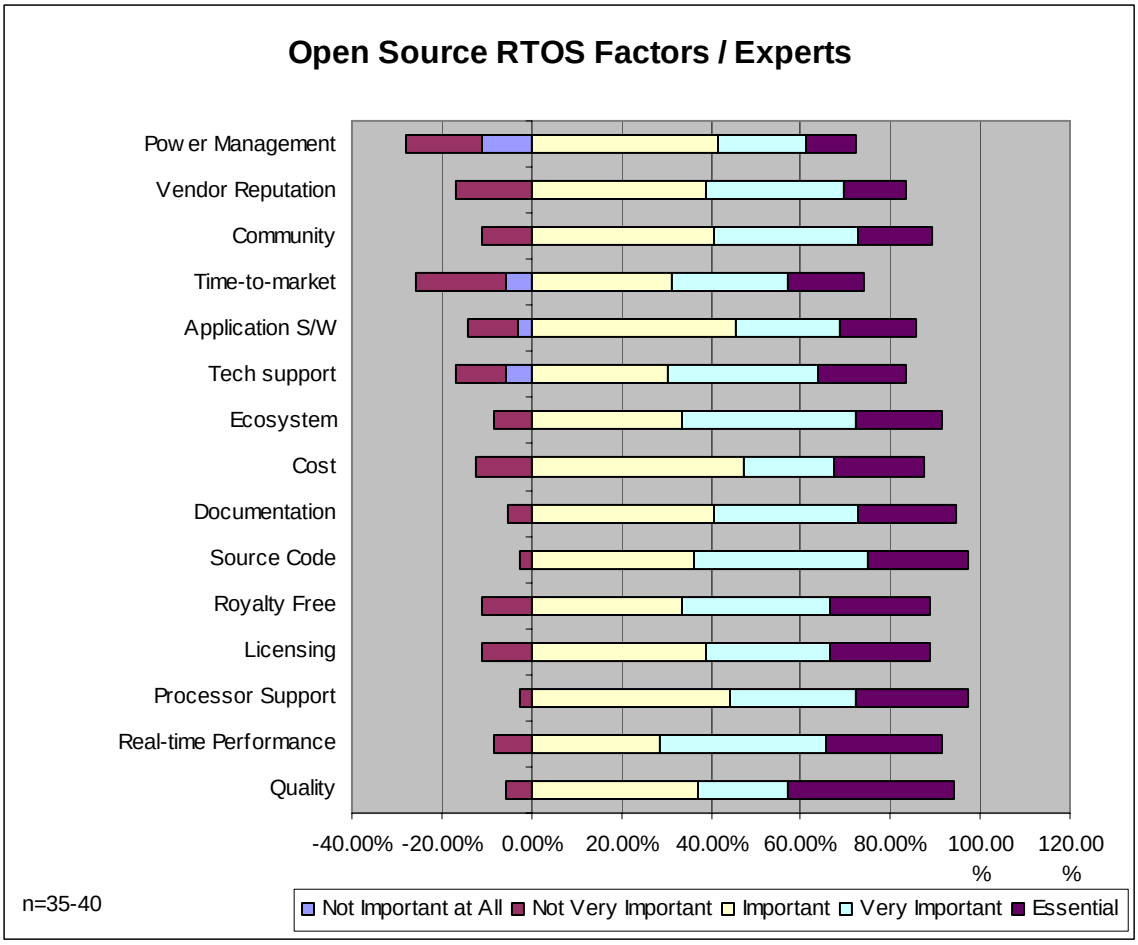
OPEN SOURCE USERS

Linux is, of course, an open source RTOS. But for the embedded systems market there are other open source choices such as *eCOS* or *freeRTOS*. This is a smaller group than either the *Linux* or commercial groups, but important nonetheless. These non-*Linux* open source RTOSes can solve many important design challenges that their more famous brethren cannot.

How do "non-experts" perceive the most important factors driving them to consider a non-*Linux* Open Source RTOS?



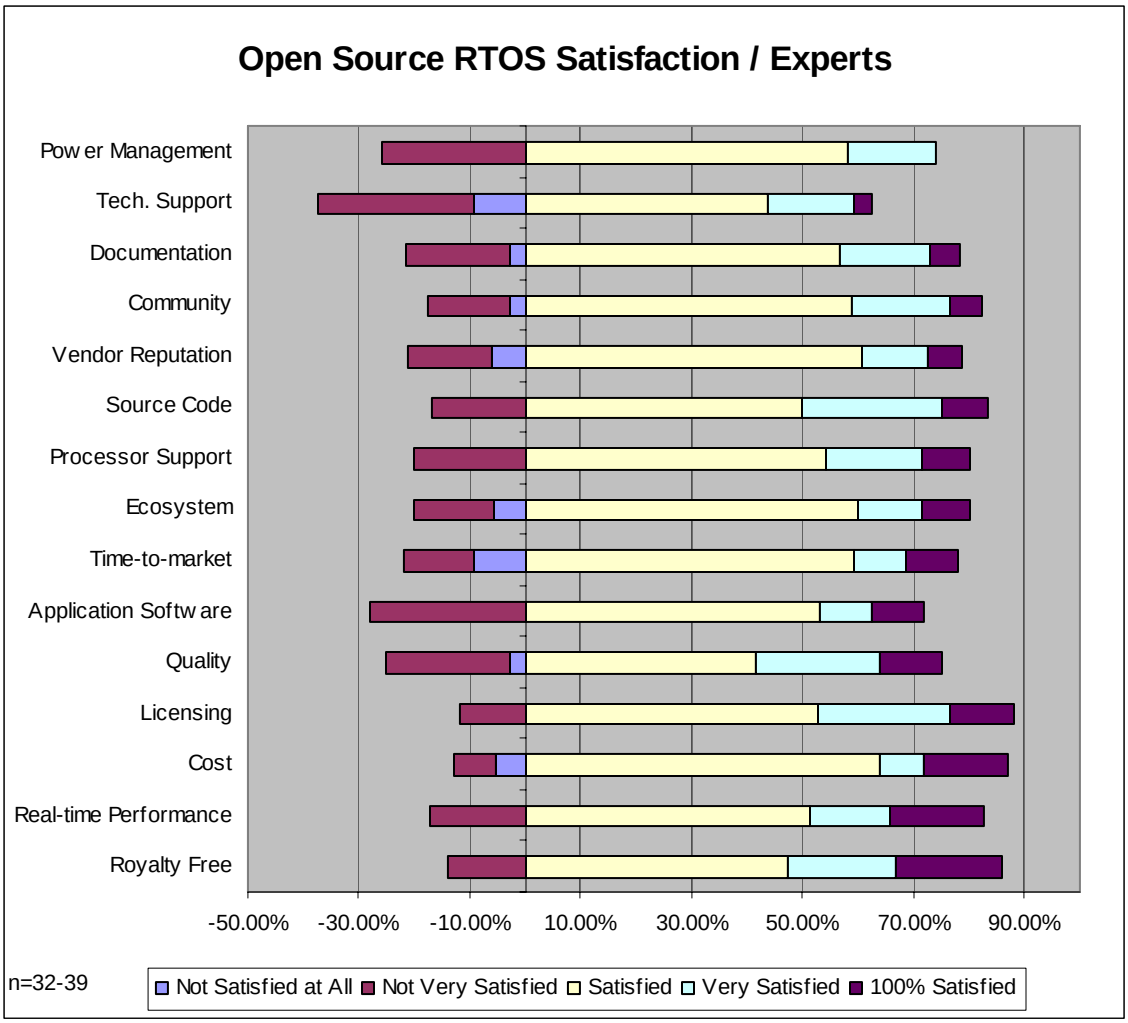
Non experts perceive "open source licensing" as the most important factor followed by software quality and the royalty-free status of open source RTOSes. What, in contrast, do experts view as important?



Expert users rank "quality," "real-time performance," and "processor support" as their three most important factors. Open source licensing follows at number four. As with *Linux*, there is a divergence between those new to the RTOS choice who perceive the "open source licensing" as a very important factor and the more experienced developers who focus on technical standard such as real-time performance or processor support.

> **Take away**. For all RTOS choices, whether open source or commercial, the technical factors predominate among experts over more "business positioning" factors such as licensing status or community ecosystem.

How do experts evaluate open source RTOSes after a project?



Unlike *Linux*, the other open source RTOSes score well on their "royalty free" and "cost." "Power management," "technical support," and "documentation" tend to disappoint.

COMPARISON OF DIFFERENT RTOS GROUPS

Every RTOS choice is unique. You should choose the RTOS that *best* fits your own application and business needs. Nonetheless, everyone is interested in comparisons. And it is beneficial to compare and contrast each RTOS choice across a range of factors. It is especially interesting to poll the "experts" and find out where they were "most satisfied" and where "least satisfied" with their RTOS choice.

To accomplish this, we gave negative weight to dissatisfaction based on percent and positive weight to satisfaction to come up with a "satisfaction score" for each RTOS type. The results are presented in the table below.

Factor	RTOS Type & Satisfaction Score			
	In-House:	Open Source:	Linux:	Commercial:
Application Software	53.85%	43.75%	48.72%	47.83%
Community	37.93%	64.71%	65.00%	52.94%
Cost	56.76%	74.36%	33.33%	54.39%
Open Source	NA	76.47%	63.16%	50.00%
Power Management	71.43%	48.39%	60.98%	46.67%
Processor Support	80.00%	60.00%	70.00%	64.71%
Quality	75.00%	50.00%	40.00%	64.71%
Real-time Performance	69.70%	65.71%	76.74%	81.13%
Royalty Free	68.75%	72.22%	54.29%	60.78%
Source Code	80.00%	66.67%	43.59%	33.33%
Tech. Documentation	45.45%	56.76%	80.49%	55.10%
Tech. Support	NA	25.00%	70.00%	52.94%
Time to Market	39.39%	56.25%	48.72%	68.00%
Tools Ecosystem	58.62%	60.00%	57.89%	67.35%
Vendor Reputation	NA	57.58%	47.37%	84.31%

We have marked in green the highest value for each factor, and in red the lowest value. What this means, for example, is that the "In-house" RTOS had the highest satisfaction in terms of quality and *Linux* had the lowest. Many of these scores confirm the conventional wisdom about each RTOS type.

- **Community**. *Linux* is #1 in terms of "community" and In-house is last. This is not surprising, since an In-house RTOS is a "go it alone" choice, whereas *Linux* is a community choice.
- **Quality**. In-house is #1 in terms of "quality" and *Linux* is last. This is not surprising, since an In-house RTOS means that the developer has optimized the code for this application, and since it is "his baby," he is less inclined to be critical. *Linux's* poor score in terms of software quality is also probably due to the fact that it is really a GPOS and must be modified for the severe constraints of the embedded systems market.
- **Real-time Performance**. Commercial RTOS is #1 in terms of "real-time performance" and "Open Source" is last. Commercial RTOS companies have spent fortunes and years in development, so it is not surprising that they do best in terms of performance.
- **Source code**. In-house is #1 and Commercial is last. Since In-house by definition means that the designer not only created but understands the source code, whereas some Commercial RTOSes do not provide source and/or it will be more difficult to obtain, therefore it is not surprising that In-house scores well in terms of "source." Commercial in contrast may not even come with source and it requires a learning curve to master, plus there is no external *Linux* or Open Source community to help.
- **Time-to-market**. Commercial RTOSes dominate "time to market" and In-house is last. Doing it yourself will be less time effective, while buying an off-the-shelf solution will be fastest.

Some indications, however, are surprising. On the "Cost" factor, for example, Open Source is #1 and *Linux* is last. A lack of satisfaction may actually reflect a sense of frustration. The marketing message from *Linux* is that *Linux* is "free," *Linux* is "open source," the *Linux* community is "there to help," etc. This implies that *Linux* will be easy, but the survey reflects the frustration that *Linux* which bills itself as "free" actually has many hidden costs in terms of software development and optimization. Hence, *Linux* creates frustration on the cost front and scores low in satisfaction in terms of cost.

- This does not, however, mean that *Linux* is "more expensive" than other choices but rather that the frustration over *Linux* cost is more painful.

Another surprise is in "Technical Documentation." Here *Linux* shines with an 80.49% satisfaction score. The *Linux* community and its openness clearly make for a well-documented source of software. That In-House scores the least well is not surprising, since In-House RTOS creators are notoriously poor at documentation.

The high scores of In-house RTOS on factors like "application software" or "processor support" are surprising. Perhaps this is because these users interpreted the question to be regarding the quality of their own RTOS application software, or their own RTOSes support for their specific processor. Since In-house by definition is built for one company or even one project, this might explain the paradoxical high marks by In-house here. Or perhaps something else is at work.

A final surprise is in the category "royalty free." *Linux* is royalty free but has the lowest satisfaction score for this factor - how can that be? Our suspicion is that again the **hidden** costs to *Linux* and in terms of licensing the not-so-obvious non-GPL applications and other *Linux*-related software add-ons create frustration in terms of the "royalty free" status of the software. In addition, the complexities of the GPL both in terms of having to open source any code modifications and worries about non-GPL software getting into the *Linux* kernel make many corporate lawyers (and hence developers) uneasy.

> **Take away**. *Linux* licensing issues are more (not less) complex than commercial variants, but this fact is not well perceived by the developer community. Hence the low satisfaction score in terms of "royalty free" *a.k.a.* licensing and hidden costs.

From our Engineer Survey (summer, 2007) data, we can produce a table comparing and contrasting RTOS choices across a range of criteria. In this question set, we asked "What is your primary reason for choosing such-and-such RTOS?" Users were able to make multiple selections, so we have computed percentages based on the total "popularity" of each RTOS choice with users able to have multiple choices. This chart, therefore, is more a reflection of relative awareness and relative choice attractiveness.

We have **bolded** the highest value for each column (RTOS Type) and *italicized* the lowest value for each column. We have made green the highest value for each row (selection criterion) and red the lowest value for each row.

Primary Criterion:	Commercial RTOS	Linux	Open Source	In-House
Determinism	2.70%	1.34%	3.00%	3.70%
Features	8.49%	10.92%	8.00%	3.70%
Memory Footprint	4.50%	4.21%	3.00%	11.11%
Price	5.41%	8.05%	**18.00%**	7.41%
Processor Support	8.24%	5.56%	8.00%	3.70%
Realtime Performance	**13.90%**	9.96%	6.00%	11.11%
Royalty Free	2.70%	9.20%	7.00%	7.41%
Simplicity	6.95%	4.21%	10.00%	3.70%
Software tools	8.37%	5.56%	6.00%	7.41%
Middleware	2.83%	1.72%	2.00%	0.00%

Open-source	5.41%	**19.54%**	14.00%	3.70%
Used before	4.25%	3.45%	4.00%	7.41%
Hardware support	6.95%	5.36%	2.00%	**14.81%**
Supplier's reputation	*1.80%*	*0.77%*	*0.00%*	7.41%
Technical Support	4.76%	2.11%	2.00%	*0.00%*
Company selected it	9.78%	4.98%	7.00%	7.41%
n=	777	522	100	27

This chart also confirms much of the conventional wisdom about choosing an RTOS:

- **Real-time Performance**. Commercial RTOSes shine in this category, and are often selected because of their valuable hard real-time performance.
- **Price**. Open source and *Linux* are often chosen because of their (perceived) low price, but other survey areas show later frustration when hidden costs become apparent and narrow the true price gap.
- **Software Tools**. Commercial RTOSes are often selected for their high-value integrated software tools, which translates into quicker time-to-market and less project headaches albeit at an out-of-pocket cost.
- **Hardware Support and Memory Footprint**. In-house RTOSes are often used when a close match between the device hardware and RTOS is necessary. This is also the case when executives or kernels are used when resource constraints prevent a full-fledged RTOS.
- **Technical Support**. Commercial RTOSes have the highest score here, again indicating a choice when ease-of-use and quick time to market factors are imperative.

Are there surprises? Of course:

- **Features**. *Linux* is chosen for features at 10.92%. This makes sense when "features" is perceived as "application software." One primary reason to choose *Linux* is the wealth of programming talent and usable drivers and application software available on the Web.
- **Simplicity**. Open source, non Linux scores high here, but we are inclined to be skeptical due to the small sample size. Perhaps this group just loves the open source concept?
- **Used before**. With the proliferation of *Linux* in universities and its ease-of-download, one might have expected *Linux* to score better in terms of how previous use translates into current use. But this is not so. In fact, the surveys both indicate that previous use or experience is not that important in *new* embedded designs - people are relatively open to new choices.
- **Company Selected it**. Many developers across all types are forced to use the RTOS their company prefers. If this is your situation, you might consider ways to persuade management to make your choice, their choice.

ANECDOTES

We asked users a series of open-ended questions about their RTOS experiences. Here are the questions and some of the more interesting anecdotal resposes.

Q. As a potential user or evaluator of RTOSes, what has been your biggest difficulty so far? What sort of information would you like to find available?

- "Memory Leaks, Hard real time support, Priority assignment and thread scheduling all these problems can be avoided by using proper decision of an RTOS."
- "Find(ing) a stable business and good technical support."
- "Really difficult to compare different RTOS."
- Finding "good documentation and support of *Linux* based free RTOS."
- "The lack of user friendliness and flexibility of the IDE (integrated development environment). (The lack of) Object oriented programming tools."
- "Technical documentation, availability of experts, tools and support for processors and peripherals."
- "Technical documentation related to exactly how scheduling is handled and ISR functionality exposing. Some RTOS does not say how the system shall behave during execution of the few of the API functions."
- "Different names and different approaches of the different vendors to support the same functionality, so must learn each new RTOS you deal with."
- "BSP integration."
- "To find a simple, easy to use rtos that can work on Microchip's controllers as well as on Atmel. In other words, an RTOS for simple 8 bit controllers."
- "Need to learn about it immediately!"

Q. As a user of RTOSes, what WARNING message would you like to convey to other, future users of RTOSes, whatever type? What has been your greatest problem(s)?

- "Good Design of your application and understanding of complete source code."
- "Do you feel like you got value from your OS investment? What did the RTOS get correct and what did they not see as a future problem?"
- "Primarily the availability of support software."
- "Real time is as good as your design and code."
- "The financial side of the vendor, and how long he will be around. The lack of user communities."
- "Be smart. Love your boss. Hate your wife (or vice versa)."
- "Suitability of Proper RTOS to the Application."
- "Try to get all the costs of the entire implementation in the design stage so that there are no surprises later."
- "Time to Market. It takes a lot of planning to make sure you hit the window. Plan for a lot of development time, especially with a new system."
- "Documentation and support for various h/w platforms."
- "Be aware of royalties and be concerned (that you can reach) qualified and reachable support."
- "Clear licensing policy, good suppport and good documetation, portability."

- "To make sure that you get good support. My biggest problem was that RTOS support was much more expensive than what I thought."

Q. As a user of RTOSes, what POSITIVE message would you like to convey to other, future users of RTOSes? What has been your greatest success or positive experience with RTOSes?

- "Use the RTOSes, especially use the Commersal version, will speed up you development."
- "Buy what you can and build the rest yourself."
- "They have improved and are easier to use and deploy than before."
- "Learn RTOS concepts, and apply them properly."
- "Hard real time is essential. Deterministic response is the heart of real time without it you might as well use a non Real Time OS. I consider real time OS (to be) an OS that you can determine execution time down to micro seconds."
- "Source must be clear not obfuscated."
- "We used the RTOS that fit the application. It had all the features and we weren't doing anything too oddball. The built in features were very well done for the most part. We got to market faster than anticipated on at least 2 projects!"
- I "love the way it makes development easier. Develop software in the same way you would on a PC, in some cases, using high level languages other than C++."
- "Commercial RTOSes have the advantage of available talent pool and pretty good documentation. For established processors, I've been able to get the chip/board up and running with the RTOS rather quickly."

Interviews: General RTOS Issues

Many of the real experts on RTOSes are the vendors themselves. We interviewed many leading vendors and tried to tie each to an important "theme" to help those selecting an RTOS choice. Here are interviews relating to general issues in selecting an embedded OS.

- AVIX-RT: TECHNICAL CRITERIA FOR SELECTING AN RTOS
- CMX SYSTEMS: RTOS FOR SINGLE-CHIP ROI
- ESOL: ADVANTAGES OF A COMMERCIAL RTOS SOLUTION
- EXPRESS LOGIC: THE BUSINESS SIDE OF SELECTING AN RTOS
- GENERAL SOFTWARE: FIRMWARE AND BIOS ISSUES
- INTOTO: SECURITY FOR EMBEDDED NETWORKING
- MENTOR GRAPHICS – TECHNICAL CRITERIA FOR SELECTING AN RTOS
- MICRIUM *µC/OS-II*: BENEFITS OF A COMMERCIAL RTOS & UNIQUE LICENSING
- MICROSOFT .NET MICRO FRAMEWORK: A NEW APPROACH TO EMBEDDED SOFTWARE
- QUADROS SYSTEMS: ADVANTAGES OF A COMMERCIAL RTOS
- ROWEBOTS: RTOS FOR DSP

AVIX-RT: Technical Criteria for Selecting an RTOS

22 October 2007 – Technical Criteria for Selecting an RTOS

INTERVIEWEE. LEON VAN SNIPPENBERG, CEO
TEL. +31615285177
EMAIL. info@avix-rt.com
COMPANY. **AVIX-RT**
WEB. *http://www.avix-rt.com/*

Q. **First of all, tell us a little bit about yourself and your company.**

A. I am Leon van Snippenberg, founder and owner of AVIX-RT. After obtaining my Bachelor of Information and Communication Technology in 1982, I worked in the field of embedded systems development. I have worked on quite diverse systems varying from X-Ray scanners to small scale laboratory equipment. From the beginning I have had a special interest in concurrency aspects and this became one of my specializations. Besides my day to day work, I've also found teaching courses to be very interesting and inspiring. Over the years I have developed a number of courses which I teach on a regular basis. During my work I found many embedded system developers are having trouble correctly applying concurrency aspects. I found this to be caused by insufficient knowledge about the subject but also by the complexity of the RTOSes being used. All this made me decide to start a company for developing and marketing an RTOS which reflects my personal philosophy of how an RTOS should look and behave. This RTOS is called *AVIX*, an acronym for *Advanced Virtual Integrated eXecutive* and the company is called AVIX-RT. Other activities of AVIX-RT are developing and teaching courses and workshops.

Q. **What, in your view, are some of the most important technical criteria in evaluating a potential RTOS choice? How would you advise an engineer "compare" one RTOS to the other?**

A. As far as technical criteria are concerned there are many. There are the more obvious ones like memory footprint and speed but I will focus on a number of criteria I consider at least as important as those two. The first is the influence of the RTOS on interrupt latency. Most real time systems have to react to events in the outside world in a timely fashion. Often interrupts are used for this. The applicability of an RTOS for a certain system is also determined by its influence on the interrupt latency, which quite often appears to be relatively large. Also consider many embedded systems have a long expected life time during which updates will be made and new functionality will be added. When evaluating the applicability of an RTOS, one should also try to anticipate on potential shorter required response times in future versions of the system and evaluate the latency against this expectation. Another criterion I find important is the RTOS user-friendliness. This starts with the API. A lot of different approaches are being followed by RTOS vendors. This varies from an API where the programmer is almost programming the RTOS itself because of the way RTOS data structures are accessed, flags have to be set etcetera, to an RTOS offering a pure function-call based interface where all of the internal data structures and related complexity are entirely hidden behind the RTOS API. I consider this an important aspect to evaluate since the second approach leads to faster system development and less errors. Evaluating the interrupt latency criterion often proves to be difficult since many vendors do not specify this and even if specified, the numbers might change in a forthcoming version of the RTOS. Still I think vendors should publish interrupt latency figures because of their importance. Evaluating the API is easier. A quick scan of the user manual should provide the required information.

Q. **We hear a lot about "hard" real-time vs. "soft" real-time with definitions usually explained in terms of applications like military ones or industrial control that need "hard" or reliable responses vs. consumer or commercial applications that can tolerate slower or less predictable responses. What are the technical criteria by which an engineer can judge whether an RTOS is "hard" or "soft"?**

A. True, there are a lot of definitions for this. I think the most generic being that a hard real time system is a system where in case deadlines are not met, disaster is said to have occurred. So the dominant aspect to judge whether an RTOS can be used for constructing hard real time systems is whether the RTOS functions are guaranteed to finish within a predetermined time, specification of which should be part of the total RTOS specification. A second aspect to judge is whether the RTOS autonomously changes priority levels like the priority boosting mechanism in Windows. This mechanism raises a threads priority when it has not been active for a long time. The effect can be the thread with the raised priority becomes active in favor of another thread which is supposed to become active based on its programmed priority. The second thread is faced with a longer latency and thus timing of the system differs from the 'designed in' timing which potentially leads to problems. Mechanisms like this may not be present in an RTOS for hard real time systems and thus presence of this feature can be used to judge whether an RTOS is hard or soft real time. Just to be clear; priority raising to prevent priority inversion or a priority ceiling mechanism do not fall into this category and presence of one of two those mechanism actually is a prerequisite for an RTOS to deserve the title 'hard real time'.

Q. **Tell us about your own RTOS, *AVIX*. What are its unique offerings?**

A. Actually *AVIX* has a number of unique offerings, the most important ones being Zero Latency and Thread Activation Tracing. Both are dealt with in more detail further on in this interview. *AVIX* also offers an easy to comprehend API whose functions are not cluttered with all kinds of arguments needing 'don't care' values in case they are not used. For all function arguments meaningful values can be supplied. This makes the user code easier to read thus preventing errors. Another feature is precise round robin timing. Many RTOSes suffer from the problem that the round robin time bookkeeping is based on the 'system tick' which generally is one or more milliseconds. This potentially leads to thread starvation. If a round robin thread is repeatedly preempted before another system tick occurs, it might remain run able and 'monopolize' its priority, leading to starvation of the other threads at that priority. Round robin time management in *AVIX* uses a resolution of a few microseconds, effectively removing this risk. Finally performance, a feature mentioned by many RTOS vendors. Tested against the open source test suite Thread Metric, *AVIX* proves to be the fastest RTOS in its class.

Q. **There are few commercial RTOS offerings for 8- or 16-bit microcontrollers. Indeed, many of these applications are "too small" to support a commercial RTOS. How does your product adjust to the very severe resource contraints imposed by resource-constrained embedded systems?**

A. To start, the targeted microcontrollers of the *Microchip PIC24* and *dsPIC* familes are not that resource constrained. Family members exist with up to 256K Flash and 30K RAM. Still in the design of *AVIX* a number of measures are taken to minimize resource usage. An important one is the system stack. Interrupts occur at unpredictable moments meaning each thread stack has to provide space for the worst case situation as far as saving state in case of an interrupt is concerned. Especially since the targeted microcontrollers offer nested interrupts, this can place quite some load on the stack sizes. To prevent this 'waste' of RAM, *AVIX* offers a software system stack. Optionally interrupt handlers can use this system stack which is shared by all nested interrupts. This is accomplished with a minimal load on the thread stacks so these can be smaller. Also the API is constructed such that almost all arguments are passed by value instead of passed

by reference. An argument passed by reference consumes RAM since it is placed on the stack while processor registers are used for arguments passed by value. This too leads to a smaller stack size requirement, thus again saving RAM. As far as code size is concerned, although *AVIX* is coded efficiently, it is not the smallest of all RTOSes. One should however realize that the rich functionality of *AVIX* leads to smaller application code since functionality that otherwise would have to be custom made is now offered by *AVIX*. An example of this is the FIFO buffer mechanism offered by pipes.

Q. **What do you mean by "Zero Latency" with respect to *AVIX*? Is this applicable to other RTOS choices?**

A. As said before, I am convinced one of the more important aspects of an RTOS is its influence on interrupt latency. Most RTOSes frequently disable interrupt handling during system calls, having a negative influence on interrupt latency and increasing the chance of missing interrupts. To me this sounds contra intuitive. One of the reasons to apply an RTOS can be that it assists in integration of interrupt handling, while at the same time the RTOS has a negative influence on the interrupt latency. *AVIX* is designed from the ground up with the requirement never to disable interrupts. As a result, when using *AVIX*, interrupt latency is exactly the same as the latency specified by the underlying hardware. *AVIX* does not add a single cycle to the hardware specified interrupt latency. This not only makes it easier to statically analyze the interrupt latency of all interrupts in the system but it also gets rid of the risk that future versions of the RTOS will not be usable due to increased latency. Very important is that, although *AVIX* never disables interrupts, full communication between interrupt handlers and threads is offered. Other RTOSes are also known to offer a feature with the same name but when studying their documentation one finds the term "Zero Latency" to be applicable to interrupts whose handlers are not allowed to make any system call. In my opinion this makes the feature useless since in this case the designer will have the same problems integrating interrupts with the application as in the case where no RTOS is used. Only very few RTOSes offer the Zero Latency feature the way it is meant to be offered. Compared with other RTOSes for the targeted microcontrollers *AVIX* is the only one, making this feature deserve the term unique.

Q. **On your website, you are critical of "regular tracing" in terms of RTOS design. Can you explain this?**

A. First of all it is important to realize what tracing is used for. There are a number of reasons to apply tracing the most important ones being able to find errors in the user code and to determine correct system timing. Tracing is often added to user code in the form of print statements or writing information to some circular buffer. This form of tracing has a large impact on the timing of the system. Even worse, I have seen designs where print statements in the different threads were synchronized using a mutex. This effectively changes the scheduling of the system such that the conclusion drawn from the trace results is useless. To obtain usable tracing, the tracing mechanism should be integrated in the RTOS as many vendors luckily do. Only when integrated in the RTOS, deterministic information about the scheduling of the threads can be obtained since it is the RTOS that is responsible both for the scheduling and the tracing. Often these RTOSes are delivered in two forms, one with and one without tracing or a special version of the RTOS is built by defining a preprocessor symbol. Still the problem here is the version with tracing has a different timing than the version without tracing. This makes it hard to detect errors like race conditions but also timing information obtained by this form of tracing is inaccurate since the version with tracing is slower than the version without. Actually this form of tracing is intrusive, it changes the behavior of the system being tested so one might ask himself the question, "What is it that I test". This problem might be solved by using the version with tracing in the production system but often tracing consumes too many resources for this solution to be viable.

Q. **What do you mean by "Thread Activation Tracing" with respect to *AVIX*? Is this applicable to other RTOS choices?**

A. As mentioned with the previous question, I am convinced tracing capabilities should be part of the RTOS and the code being tested should be the same code as used in the production system. This guarantees the behavior of the production system to be the same as that of the system being tested. This is exactly what *AVIX* offers. Thread Activation Tracing is a mechanism where to each thread one of the I/O ports of the target microcontroller can be assigned. When activating a thread, *AVIX* will set the assigned I/O port high and when deactivating a thread, the port is set low. By observing the applicable ports on a logic analyzer one can see the exact scheduling pattern of the different threads. This form of tracing gives clear insight in the operation of the system based on *AVIX* and offers unprecedented test facilities. By using the trigger facilities of the logic analyzer it is for instance possible to test for latencies being to long. The influence on the scheduling because of changed system parameters like thread priorities and size of pipes can be observed. Learning to work with the RTOS becomes easier since the effect of code constructs and changes hereto can be observed. Thread Activation Tracing is online and real time. It is built into the very heart of *AVIX*. There is no need to transfer data from the system to some offline analysis tool. Of course, like all forms of tracing, Thread Activation Tracing is optional. The system designer has to activate it on a per-thread basis. Important aspect is that system timing is exactly the same whether or not Thread Activation Tracing is used. This is feasible due to the time it consumes in the scheduling process which is only 250 ns. To my knowledge this form of tracing is unique and partly feasible because *AVIX* is developed specifically for the 16 bit Microchip microcontrollers. Most general purpose RTOSes can not rely on I/O ports being available on the target platform, a reason making it difficult for these RTOSes to offer a comparable mechanism.

Q. **You are also critical of "time outs"? Why? How does this impact the *user* of a commercial RTOS as opposed to the *designer* of RTOSes?**

A. Actually I am not critical of time-outs in general. I think an RTOS should offer the option to use time out functionality with blocking functions. What I am critical about is the way time outs are offered by RTOS designers and used by RTOS users. Software in general should behave in a deterministic fashion. Two threads wanting access to a critical section do so by trying to lock a mutex. In a correct working system, each of these threads is guaranteed to occasionally receive mutex ownership, be it that waits can be involved. When a deadlock occurs, this means a programming error is made which should be solved during testing. In practice I see time outs being used to detect a deadlock where some recovery mechanism is implemented to escape from this situation. Bottom line is the deadlock is a symptom of a programming error which should be solved by changing the software instead of activating some dynamic recovery mechanism. Second I see time outs being used for timing functionality. I think this is principally wrong. Time outs are an error detecting mechanism and timing in an RTOS based system should be based on timers which are meant to be used for this purpose. Both situations lead to complex code which is hard to comprehend and in itself leads to errors. One of the reasons the above constructs are used is that most RTOSes offer a time out argument to all blocking functions. If not applicable this argument is set to something like 'infinite' but since the time out argument is present in the API, a programmer is easily tempted to change the value of this argument for purposes I consider wrong. Where time outs are required is on the interface between the core of the embedded system and the outside world. The system must anticipate on the outside world behaving different than expected. A communication cable can be detached; a sensor can break down etc. These are not errors in the system implementation but errors in the domain and time outs are an ideal mechanism to detect those situations so the system can react. *AVIX* uses a different approach for using time outs, reducing the risk of using it for purposes it is not meant to be used.

Q. **How important are software add-ons, such as networking protocol stacks, drivers, GUI development, etc., in the selection of an RTOS? Are there any parameters you can suggest on selecting these?**

A. Systems based on an RTOS always interact with the outside world. This is done through drivers, a mechanism for which most RTOSes offer support. For this reason it is fair a customer expects support from the RTOS Company to at least provide sample code showing how to create those drivers. In my experience sample code often is considered sufficient since driver functionality and for instance their error recovery capability differ a lot from customer to customer so it is hard to offer a 'one size fits all' approach here. It is of course true a large number of customers can use prebuilt drivers and do not need specific customizations so offering this does give the RTOS added value.

Q. **Thank you for this interview.**

CMX SYSTEMS: RTOS FOR SINGLE-CHIP ROI

29 October 2007: RTOS for Single Chip ROI

INTERVIEWEE. CHUCK BEHRMANN
PRESIDENT
TEL. 904-880-1840
EMAIL. CMX@CMX.COM
COMPANY. CMX SYSTEMS, INC.
WEB. http://www.cmx.com/

Q. **First of all, tell us a little bit about yourself and your responsibilities at CMX.**

A. I am president and CEO of CMX. I started CMX Systems in 1990, with the goal of providing engineers with a comprehensive set of software to help program their embedded applications. Today, I oversee the day to day operations and strategic planning. I also do software coding and remain active in our product development strategies.

Q. **CMX is one of the older RTOS companies - can you give us a quick run-down of your RTOS and related products?**

A. CMX Systems core business is to develop and support real-time, multi-tasking operating systems (RTOS), TCP/IP stacks, Flash File Systems, USB stacks and the CANopen stack for a wide variety of 8-bit, 16-bit, and 32-bit microcomputers, microprocessors, and digital signal processors. The company's RTOSes support more than 50 processor families and over 30 C-compiler vendors. CMX also offers the tiny *CMX-MicroNet*, which is a unique TCP/IP stack that is targeted for 8-, 16-, 32-bit and DSP processors with limited ROM and/or RAM and *CMX TCP/IP*, a full-featured TCP/IP stack designed for 16-, 32-bit and DSP processors. CMX additionally offers four different Flash File Systems to best meet the memory management needs of embedded developers. CMX-USB is offered for designers wishing to add USB connectivity to their products.

Q. **When we set this interview up, you indicated that you were both pleased at the media buzz around CMX's previous announcements (RTOS capabilities for 8-bit architectures) and also had some new angles to share on that. First of all, tell us about how CMX helps 8-bit designs enjoy the benefits of a commercial RTOS.**

A. CMX started out with a RTOS that was tailored specifically for 8 bit processors. We keyed on a very small ROM footprint as well as very minimal RAM usage. There were a number of applications that can utilize a RTOS and were specified to run on an 8-bit processor, especially back in the 90s. We were also one of the first commercial companies to introduce a TCP/IP stack specifically designed for 8-bit processors, again with a very small ROM footprint and RAM usage, which was critical given the constraints of the processor memory. There were a lot of applications that needed internet connectivity but customers did not want to have to go to a 16- or 32-bit processor for a number of reasons.

As more 16-, 32-bit processors became available, with reduced pricing, more features, etc. more and more companies were starting to use them. Therefore, we took the same software for 8-bit processors and ported it to the 16 and 32-bit processors, as well as DSPs, while maintaining a very small ROM/RAM footprint. In addition, we produced new software that would run on higher end 16-, 32-bit and DSP processors but also required a larger ROM/RAM footprint.

Q. **Many observers see a transition from 8-bit architectures to 32-bit, namely ARM. Do you agree that this transition is occuring? How do your products address the needs and capabilities of 32-bit chips?**

A. Yes, we have seen a lot of this recently. As companies need more processing power, peripherals, ROM and RAM, we see many migrating from 8-bit processors to 32-bit processors. There are many 32-bit processors that are "single chip solutions" and are quite inexpensive. CMX products fit extremely well, since these 32-bit processors still have a limited amount of ROM and RAM. Usually, RAM is the more precious resource, since many have 64K to 512K of ROM, but only 16K to 64K of RAM. So, many RTOSes and TCP/IP stacks, etc. would still need additional RAM but our products were tailored for an extremely small RAM footprint. For example, our CMX-*MicroNet TCP/IP* stack can run in as little as 2K of RAM. Many times when a company switches to a 32-bit processor, they are using more software, such as RTOSes, TCP/IP stacks, USB stacks and Flash File systems. So as engineers add software to their application, they can easily start running out of RAM and that is why RAM usage is a very important consideration when deciding which software to purchase.

Q. **You mentioned to us privately that CMX's solutions can help single-chip solutions because by virtue of your small footprint, CMX can "free up" RAM or ROM resources for other software applications. How is this achieved in terms of technology?**

A. This achievement was accomplished in many ways. Our CMX-*MicroNet* stack was intentionally written from the ground up to use as little RAM as possible. The same applies to our RTOSes, USB stacks and Flash File systems. We have always been concerned about RAM usage since the beginning. That is one of our key goals when producing new software. I have been in the field a long time and used the Intel 8048 that only had 1K of RAM so being brought up in that era, RAM was extremely precious and we feel it still is today.

There are tricks you can use when you write your code as well, such as using very few locals, keeping nested calls at a minimum, number of parameters passed, use of link list or not, using bits within a byte, etc.

Q. **Is this primarily a story about customers "saving" in terms of external RAM/ROM? What is their benefit here?**

A. I think it is more a story of savings within the on- chip ROM/RAM. If a processor addresses external ROM and RAM and the engineer has potentially, many megabytes of it, the concern for ROM/RAM usage is greatly reduced. Even if they populate external ROM and/or RAM, that still costs money, and most companies are very concerned about cost of product, especially in today's very competitive marketplace. There are many "single chip solutions" that do not address external ROM and RAM but are attractive from a cost standpoint. Using these "single chip solutions" makes it much more critical to be aware of RAM and ROM usage of course.

Q. **Are there benefits as well in terms of additional device capabilities? Can you share with us what sorts of new features, capabilities, or smaller footprints your 32-bit customers have been able to achieve in this way?**

A. Yes, there is. Many times (though not always of course) as you climb into the 32-bit world, you will find built-in features like a MAC for Ethernet connectivity, USB support for Host and/or Device, VGA display, etc. We offer two TCP/IP stacks for instance, one with very little ROM/RAM usage and one with a fair amount of ROM/RAM usage. The difference is that the smaller one does not offer all the capabilities of the larger stack but in many cases, the added features of the larger one are not needed. Our USB stack was written to conserve ROM/RAM usage and that characteristic applies to our other products as well. We have many, many customers using our products in 32-bit processors that do not need to add external ROM and/or RAM. This saves in board real

estate, cost, and procurement while increasing the ability of the end product to compete or produce better margins.

Q. **Thank you for this interview.**

eSOL: Advantages of a Commercial RTOS Solution

25 October 2007: Advantages of a Commercial RTOS Solution

Interviewee. Senthil Kumar
Field Application engineer
Tel. +81 3-5302-1360
Email. s-kumar@esol.co.jp
Company. eSOL
Web. http://www.esol.co.jp/

Q. **First of all, tell us a little bit about yourself and your responsibilities at eSOL.**

A. I have been appointed by eSOL as a Field Application Engineer for foreign operations. Working from Tokyo headquarters of eSOL Co Ltd, I support customers outside of Japan by responding to the queries on their existing and new project implementations, which use eSOL's core technologies. I also act as a liaison between our customers and our technical team to resolve any technical related issues.

Q. **eSOL is a commercial RTOS vendor. Can you outline in very brief detail the product solutions offered by your company?**

A. We offer multiple RTOS suites, from micro-kernel to POSIX compliant operating systems and RTOS that supports a blended, scheduled mechanism of AMP and SMP, along with tightly coupled integrated development tools with a cross compiler, source-level remote debugger with task-awareness, and system and real time analysis tools. We offer a wide range of middleware packages such as file systems, USB host/device components and TCP/IP stack as well as application specific middleware through our 3rd party network.

Q. **What is eSOL's relationship to the *TRON/iTRON* standard?**

A. The original *uITRON* specification was developed in Japan. Due to its excellent real time performance when used in CPU's with less power and memory capacity, it is also used in many embedded consumer devices in Japan and most recently has expanded to other Asian markets. *uITRON* is also being used successfully in space and aviation systems, the automotive industry and factory and office automation devices, among others.

eSOL is a regular member of the *TRON* Association who facilitates the RTOS standardization based on various *TRON* derivatives, of which one of them is *uITRON* (micro *ITRON*) specification (the latest specification is version 4). Based on this specification, we here at eSOL have developed high performance and well optimized RTOS kernel, which is called *PrKernelv4*. As the member of *TRON* Association (*ITRON* Working Group), we are deeply involved for technical discussions where we help define the *ITRON* specifications as well as promotion of the standard.

Q. **What is eSOL's relationship to the T-Kernel standard?**

A. An eSOL representative serves as Vice-Chairman on the *T-Engine* forum and we actively participate in the kernel and development-environment-working group (KD-WG). eSOL is also instrumental in the analysis, proposal and review phases of the core technical specification design of the *T-Kerne*l.

T-Kernel is part of the *T-Engine* specification, which provides a standard for an open embedded systems development platform, ranging from simple to complex systems. Being a successor of the *uITRON* standard, *T-Kernel RTOS* specification fulfills the needs of today's large scale embedded systems. *T-Kernel* OS specification considerably reduces the burden of the software development process and middleware development, and distribution becomes relatively easy for the vendors. It not only supports the memory

protection feature and process model but also is flexible enough to address various application requirements in the embedded industry.

Q. **Development tools are increasingly important to selecting an RTOS. Tell us about your *eBINDER IDE*.**

A. *eBinder* is a powerful integrated development environment for real time operating system-based development. *eBinder* addresses the embedded market's development needs by providing:

- Integrated development platform
- Multi-tasking debug tools
- Real-time debugging capability

eBinder is a complete, graphical Integrated Development Environment that runs on Windows-based PC. It can connect to a board using serial, Ethernet, or JTAG/ICE interfaces. All of *eBinder's* features are available through the JTAG/ICE interface: eBinder uses the JTAG as a debug port while preserving the JTAG functionality for execution control.

The *eBinder* package provides a full RTOS-based development environment. eBinder includes the *PrKERNELv4* and all the eT-Kernel operating systems with full source code, pre-integrated with a Board Support Package (BSP) for reference platforms. *PrKERNELv4* is eSOL?'s open standard multitasking kernel, and is compliant with the *ITRON 4.0* specification. The BSP provides hardware-dependent code to support selected evaluation boards. *eBinder* also supplies a target monitor ported to the development board. Multitasking applications require a reentrant C library. Either the bundled compiler or *eBinder* provides a thread-safe C library that includes support for stream I/O. A Unix-like file system simplifies stream I/O access to media. eSOL's middleware libraries for USB, TCP/IP, and FAT file system integrate with *eBinder* to provide a full base solution. Together, all of these features add up to a complete platform that allows developers to start creating their application without worrying about low level porting.

eBinder brings value to embedded development by providing a complete solution to debug complex, multi-threaded real time applications. While most other tools offer some features for multi-threaded debugging, *eBinder* was designed from the beginning to provide complete support for real time, multi-threaded environments. Using *eBinder* decreases time to market by letting product groups focus on enhancing their application instead of concentrating on the platform and tools. *eBinder* provides an RTOS-based development platform with powerful tools for multi-threaded and real time debugging. *eB-SIM* provides a way to quickly start development before the hardware is available. By using *eBinder*, developers can create a high-quality, full-featured product that goes to market quickly.

Q. **Middleware is also very important. What is *eParts*?**

A. Yes, that's correct, middleware plays a vital role in RTOS development suite. *eParts* is not just a middleware package, it is a complete RTOS suite from eSOL, consisting of an RTOS kernel, TCP/IP stack, FAT based file system, USB host and device components for leading 32-bit and 64-bit embedded microprocessors. All *eParts* components are modular and licensed individually.

Recently, through our strategic alliance with the industry's leading middleware vendors, we have added a significant number of middleware packages to the *eParts* family, from multimedia codecs to DLNA stack, in a response to the growing need from the field of consumer electronics, the home ground for eSOL where we have many of Fortune 500 customers using our software.

Q. **eSOL supports ARM and TI DSP's (OMAP). What chip architectures are supported?**

A. Apart from supporting *ARM* architecture, eSOL's embedded OS kernels support *MIPS*, *SH series* processors, *PowerPC*, *MicroBlaze* (soft core processor) and *x86* architectures.

Q. **Tell us about multi-core. What is eSOL's take on the whole "multicore phenomenon?"**

A. The definition of multi-core differs from engineer to engineer. The multi-core we are strategically focusing here at eSOL is one physical silicon package having multiple homogeneous cores inside as opposed to heterogeneous cores such as one microcontroller and DSP, which is a very popular multi-core configuration today, and many developers deploy this method. For the type of multi-core eSOL supports, *ARM11 MPCore* is a good example that can host up four *ARM11* cores.

The software in modern complex embedded systems, including consumer electronics and auto-infotainment devices, mobile phones, medical equipment, etc., performs a wide variety of processing. The growing problem in those areas, from the hardware perspective, is the computation power and power consumption, and naturally, multi-core, though it is not the only solution, offers the most practical solution today, and many developers are looking into deploying the multi-core in their future design.

Some designs require the highest throughputs the system can crank out, some are very real-time (determistic), and some are soft real-time. They may require several device drivers with a lot of I/O, and many of them are concurrently executed. This is a result of ever increasing demands for more and advanced features, and the increased demand poses several challenges from the software perspective. Especially, an RTOS would play even more significant role in the multi-core environment. Common models are SMP and AMP, and each model has its advantages and disadvantages in the area of throughput, concurrency, real-time determinism, reuse of existing software, programming model and debugging environment, to which eSOL has the solution.

eSOL provides its next generation RTOS kernel called eT-Kernel multi core edition for *ARM11 MPCore* processors. Although it is basically a symmetric multi processing (SMP) based OS, it offers a unique feature that blends the SMP mode (called True SMP Mode or TSM mode) and the Single Processor Mode to reap the benefits of both symmetric and asymmetric multiprocessing models. In the basic scheduling mode, four configuration combinations are possible. Legacy asymmetric multi processing systems can easily be ported to *eT-Kernel MPcore edition* by selecting the appropriate combination.

In addition to the basic scheduling modes, *eT-Kernel multi core edition* supports two more special modes called "*Single Processor Mode on TSM cores*" and "*Serialize threads on TSM cores*". For example, when you want to use both the TSM mode and Single Processor Mode with the *MPCore* configured as a dual-core processor; this can be achieved by selecting the "*Single Processor Mode on TSM cores*". In another case, when you want to reuse the single processing software without sacrificing the benefits (like auto load balancing) of symmetric processing, then you choose the "*Serialize thread on TSM cores*" mode configuration. Thus, eSOL has embraced the whole "multi-core phenomenon" by implementing the various possible mode-blending combinations to support the different needs of the users.

Q. **Currently you support only the *ARM11* for multi-core. What other architectures do you plan on supporting?**

A. We currently support *ARM11* only for our multi-core kernel because there is a comparatively high market need for *ARM* chips. *ARM* holds 2/3 of the total embedded chip market. *eT-kernel multi-core for other chip architectures* will be based on the market needs and trends in the future.

Q. **eSOL seems to have a special relationship with Texas Instruments. Tell us about your *DaVinci* solution, please.**

A. We are a long term partner with Texas Instruments The relationship goes back to almost a decade when TI introduced the first system-on-chip based on *ARM7* and DSP, aiming at digital still camera, namely for Japanese and Asian OEMs, where eSOL offered the RTOS and also developed multiple middleware. Since then, eSOL has been supporting generations of TI's SoCs in the field of consumer electronics where we license our RTOS, middleware and IDE as well as providing professional services. Because of our long-term involvement and proven track record, eSOL was appointed as an Authorized Software Provider for TI's *DaVinci* processors.

We wear two hats when it comes to *DaVinci* solution. As an RTOS vendor, our embedded solutions support for Texas Instruments' *DaVinci* series processors *TMS320DM644x* and *TMS320DM355* includes *PrKERNELv4 RTOS* along with *eBinder IDE*, file systems with media drivers, USB host and device, TCP/IP protocol stack, and so on. We also team up with middleware vendors from codecs to DLNA compliant software stack. As TI's Authorized Software Provider, we sublicense TI's leading codec for all (not only *DM355* and *DM644x*) *DaVinci* processors with pre- and post-sales technical support. This includes free four hours of pre-sales support for all customers who purchase development kit from TI.

Q. **eSOL is very active in the consumer electronics space, automotive etc. Are there certain applications that you target as especially relevant for your technologies? What are these? Why are they a good fit for you?**

A. Although RTOS can be used for any products, we tend to have first tier vendors as our primary customers, especially, in the field of consumer electronics and the automotive industry. Stability and reliability of the RTOS is the number 1 priority for them, and they choose a RTOS as a long-term strategic investment. As such, they have a long list of requirements for the RTOS and RTOS vendor. With our well-balanced product portfolio and professional services, we have earned an outstanding reputation with world-class product manufacturers, and this has established eSOL as a market leader in those fields.

Q. **Why do people go for commercial RTOSes while there are Open source OSes?**

A. Commercial RTOSes come with stable professional support for their products either by e-mail or on-site support. Most customers prefer one window reliable solution for their development projects to minimize risks and ensure on-time development. Open source does not come with support or engineering assistance and can lead to project delays or failure. This is unacceptable to our customers.

Q. **How well does your *PrKERNELv4* implements the *µITRON* specification?**

A. Our *PrKNERNEL v4 RTOS* is not only fully compliant with *µITRON 4.0* specifications, but also adds its own features such as support for mutex, variable size memory pool, dynamic stack allocation, alarm handler and more.

Q. **How do you predict the impact of the *ITRON* and *T-Kernel* based RTOSes in the near future? And what will be the eSOL's role in that?**

A. Regionally speaking, *ITRON* has become the standard RTOS specification among Asian countries, and this will continue. *T-Kernel* based RTOS is growing and more and more developers are finding it very appealing to use in their future projects, but this would probably still take some more time. To be honest, we do not know if *ITRON* and *T-Kernel* would be used in North America and Europe or would become a mainstream (probably not), as much as currently used in Asia, but this is not an important point. The point is this is the standardized RTOS and the specification is open for those who wish to obtain. The standardization is important as you see in the high-tech industry, both software and hardware.

Embedded RTOS has been unique in a way that almost no standardization efforts have been made. This is true when you see there are still so many and highly proprietary RTOSes being offered in the market. This is mainly due to a fact that average embedded systems have been relatively small and simple, but not any more now when we talk about consumer electronics contains million lines of code. We see that the RTOS standardization is an inevitable process in order to increase efficiency for the whole embedded systems design. And, as the RTOS gets standardized, people will uncover the importance of development tools as you see what is happening to embedded *Linux*. The RTOS is important, but supporting development tools would be important as well, in some cases more important than RTOS. To this end, eSOL is one of few vendors who provide RTOS and development tools tightly coupled. eSOL will continue to develop high-end RTOS and make sure that its corresponding development tools have significant values to its customers.

Q. **Thank you for this interview.**

EXPRESS LOGIC: THE BUSINESS SIDE OF SELECTING AN RTOS

8 November 2007: The Business Side of Selecting an RTOS

INTERVIEWEE. JOHN CARBONE
VP, MARKETING
TEL. (858) 613-6640 X-202
EMAIL. jcarbone@expresslogic.com
COMPANY. EXPRESS LOGIC
WEB. http://www.rtos.com/

Q. **First of all, tell us a little bit about yourself and your responsibilities at Express Logic.**

A. I'm a former software developer, having worked on RTOSes and real-time applications in aerospace, industrial control, and scientific computing. I bring that technical background to Express Logic, where I manage the marketing activities related to our *ThreadX RTOS* and compatible middleware products.

Q. **Although we are sure that Express Logic's products are second to none, we talked before the interview and decided to focus more on business issues. Before we do that, however, can you just give us a quick sketch of Express Logic's products?**

A. Express Logic's products include the *ThreadX® RTOS, FileX® file system, NetX™ TCP/IP stack, USBX™ USB stack, PEGX™ graphics toolkit*, and our new *TraceX™ graphical event analysis program*. Our products feature a small memory footprint, low overhead operation, efficient real-time performance, and are all sold royalty-free. They are designed to be easy to use, and to help our customers bring products to market as quickly as possible. To-date, over 450 million electronic products have been produced and shipped with *ThreadX* inside, making *ThreadX* one of the most widely deployed RTOS products in the world.

Q. **Now let's talk about business. Why do you think that selecting an RTOS is as much a "business" decision as a "technical" one?**

A. Technical criteria for selecting an RTOS are certainly important, but perhaps not as all-important as some might assume. For example, there is much emphasis on real-time performance in an RTOS. But, do developers really know just how much performance their application actually needs? If they do, then an RTOS that delivers that performance might be suitable, but is one that delivers more performance actually any better? Many RTOSes might meet the performance needs of an application, so how would one choose among them? Likewise, memory footprint is important. But, if an application (including the RTOS) must fit in 256KB of memory, is it really relevant whether the RTOS occupies 2KB rather than 4KB of that?

So, in my opinion, many times the selection of an RTOS cannot be resolved through technical criteria alone. You may find that several choices appear to meet all your criteria, or at least offer comparable overall technical scores. In those cases, you can consider other factors to find the RTOS that represents the best choice for your project.

Moreover, if the RTOSes you're considering all fall within a relatively small envelope of technical criteria, a major distinguishing factor outside that realm might overwhelm the small technical advantage one RTOS has over another.

In short –many RTOSes can "do the job" from a technical perspective, but selecting one that you'll be happy with might very well depend on several non-technical criteria. Now,

just what are those non-technical criteria? I think the following are all potentially deal-making or deal-breaking issues that demand consideration:

1. Cost
2. Licensing Terms
3. Legal Risk
4. Time-to-Market

Q. **One phenomenon that we have bumped into in our RTOS survey is that engineers or companies often seem to "undervalue" their own time (in terms of building their own RTOS or choosing to modify *Linux*) vs. the out-of-pocket costs of going with commercial software. How would you advise an engineer or manager to better evaluate the true costs of different RTOS choices?**

A. This is a huge cost issue, relevant to many embedded development projects. "Cost" has many components, and counting only some of them can easily mislead the evaluator. Typically, management might fall in love with a high-level cost benefit (eg: *Linux* is "free") and let that blind them to other cost factors that might obliterate that benefit.

Regarding true cost, I suggest consideration of the following components:

Cost Component	Linux	In-House	Commercial
Acquisition Cost – how much does it cost (up-front) to procure (license) the RTOS?	Indeed, in this area, *Linux* is free. But, only if you download it yourself from the web.	By definition, an in-house RTOS isn't "procured," it's developed in-house. The person-hours required to design, develop, and maintain the RTOS must be cost-estimated.	Typically, commercial RTOSes are licensed for an up-front fee, with or without royalties (which would show up under "Manufacturing Cost" below).
Development Cost – how much will it cost to develop your application using this RTOS?	*Linux* might score well here, based on its broad ecosystem of compatible middleware and freeware, minimizing what has to be developed in-house as part of the application. Conversely, *Linux* might	Depending on how much is put into the in-house RTOS, this component can be less expensive or more expensive as a result. Regardless of the feature-set designed into the RTOS, the availability of	Commercial RTOSes generally offer an ecosystem of compatible middleware that can be procured (adding to the cost above) for use in your application (decreasing the cost in this area). Development tools generally support commercial RTOSes through kernel-aware debugging, event trace and analysis, and builder integration, all serving to help speed the development process.

	suffer here due to its complexity, which will slow down the learning curve and increase the error rate. In addition, if *Linux* is procured directly from the web, someone must find it, download the right elements, configure it for the target system, and build it. That person's time is a cost.	compatible middleware from 3rd parties will likely be non-existent, meaning that all code must be developed in-house. Also, development and debugging tools might not be geared to support the in-house RTOS, making development less convenient.	
Manufacturing Cost – how much does the RTOS cost "per unit" manufactured?	*Linux* does not require royalty fees per unit, so one might conclude that manufacturing cost is $0. But, there is a "royalty" for *Linux* in the form of the 2MB of memory that must be added to every product just to hold *Linux.* That 2MB of memory isn't free. Either it's an extra 2MB, or it's 2MB of existing memory that the application can't use.	Typically, the in-house RTOS scores well in this area. No royalties are required (unless your corporation charges an in-house transfer fee), and memory size can be controlled to avoid significant expense.	For commercial RTOSes, this reduces to a royalty vs. royalty-free issue. A royalty-free RTOS may come at a higher procurement cost up-front, but many royalty-free RTOSes are also LESS expensive than royalty-bearing alternatives! The developer is well advised to add both factors to determine true (acquisition+royalty)/volume unit cost.

Support Cost – How much does it cost to support the RTOS?	If *Linux* was downloaded directly from the web, then all support is your responsibility. That means keeping up with bug fixes, new code, enhancements, and training. These person-hours also represent cost that must be considered.	In-house RTOSes require support from in-house resources, which have a cost. With nowhere else to go for support, all costs for bug fixes, new code, enhancements, and training become yours.	Commercial vendors generally offer full support for an annual fee. Optionally, developers might opt to self-support, especially if full source code is licensed from the vendor.

Q. ***Linux* is "free." Obviously, *Linux* might take more time than a commercial RTOS, but are there other factors that you feel increase the business cost of *Linux* vs. a commercial OS?**

A. In addition to the cost factors discussed above, *Linux* carries an additional cost in the legal area. This cost manifests itself as risk related to infringement of the intellectual property rights (IPR) of others. Contributors to *Linux* might (and "might" is the key word here) assert patent, copyright, or other IPR on work they've contributed to *Linux*. This "might" occur some time down the road after many units have been deployed. What is the cost of a negotiated settlement with such a claim? Even if baseless, lawyers cost money (at rates as high as $700/hr), and claims can consume many hours. Even if no claims are encountered, a lawyer might need to review licenses and contracts related to your development, to make sure no such claims are likely to succeed, and that itself costs money. Then, there's always the potential that you've missed something.

With a commercial RTOS, generally there is no such risk. Commercial RTOS vendors universally indemnify licensees against any such claims, making all such problems the responsibility of the vendor, and not the licensee. This can be a huge difference between a commercial RTOS and Linux or an In-house solution.

Q. **Another business differentiator is licensing models. There are royalty models as used by companies like Microsoft, subscription models like those used by Wind River, royalty-free as used by many companies including Express Logic, and of course GPL / Open Source as used by *Linux*. How would you advise an engineer or manager to sort out these licensing models from a business perspective? What would you recommend he look at for in terms of costs?**

A. It's easy to hide costs with creative licensing plans, so developers are advised to calculate "total cost" as the sum of acquisition cost (up-front) and royalty (per unit manufactured). That eliminates the games that might be played by moving cost from one place to another.

Another area for scrutiny is tools cost. Some RTOS vendors also bundle tools with their RTOS, even requiring the use of their own tools. Such tools have a cost, and that cost might be used to hide some of the cost of the RTOS, making the RTOS appear to be less expensive. For example, which RTOS is less expensive:

	RTOS-A	RTOS-B
Up-front	$9,000	$10,000
Royalty	$0	$0
Tools	$6,000/seat	$1,000/seat

As you can see, such bundles can be misleading if one considers just the RTOS cost. The point is that some RTOSes have an influence on related costs, like tools.

Q. **What about patent infringement and indemnification? As you know there have been lawsuits about this between companies, and there are always concerns (especially vis-a-vis *Linux*) about embedded software exposing a manufacturer to a lawsuit down the line. Do you think an engineer should worry about any legal issues in choosing his or her RTOS partner?**

A. Patent and Copyright infringement should be concerns of every software developer. It's generally considered unethical to copy someone else's work and pass it off as one's own. The ACM states exactly that in its code of ethics (http://www.acm.org/about/code-of-ethics). Sadly though, it's not always illegal. If you decide to copy any part of someone else's work, you should make sure to seek good legal advice regarding any potential liability. Conversely, if you use a product that includes work copied from another, you may be liable. Again, you should seek legal advice to assess the risk. As mentioned above, commercial indemnification is one way to bound these risks, or eliminate them altogether. Without such indemnification, how can you be certain that the software you're using doesn't infringe someone else's rights? In effect, you're indemnifying yourself!

Q. **One thing that our RTOS survey indicates is that many engineers worry about the "company stability" or "longevity" of their RTOS partner. Do you think this is important? How would you recommend that an engineer assess the business viability of his RTOS partner?**

A. This is a huge consideration. Developers will end up with an in-house RTOS if their commercial vendor disappears or decides to stop supporting the RTOS for some reason. As a partial protection against this, licensing full source code can be very comforting, especially if the RTOS is relatively simple. Still, the company stability should be considered in the following areas:

a) How long has the company been in business	If <5 years, a more careful analysis might be warranted
b) Is the company profitable	Unprofitable companies are going out of business – it's just a question of when. Public companies must publish P/L reports quarterly, while private companies do not. Ask the vendor for this information, especially if in business less than 5 years.
c) How widely has the RTOS been used	The more deployments the better, as high volume translates to field-testing and validation. A widely deployed RTOS is less likely to have latent bugs, and is more likely to facilitate successful development and timely product

	completion.
d) What do other users say about the company and the RTOS	Check forums, message boards, and the company's own web site for customer testimonials and comments.

Q. **Engineers often want access to RTOS source code. Do you think that there are business reasons why this might be important? How so?**

A. Apart from any technical reasons, source code provides a "safety net" in the event of vendor abandonment. In the "olden days," companies would pay for "bonded storage" of source code, to be freed up upon bankruptcy or other catastrophic event. Today, source code is generally available at an affordable cost. This makes sense as a business move, but also has technical benefits for developers.

Q. **What about the hardware and software "ecosystem" around an RTOS? How important do you think partnerships are in terms of the developer perspective? How is he to assess whether Express Logic has a "strong" or "weak" partner ecosystem?**

A. The ecosystem tends to be more important as the complexity of the application increases. Some developers can license everything they need from the vendor, and develop the rest themselves. Others prefer to integrate 3rd party components and focus their value added somewhere else. The right balance will vary greatly from company to company, and depending on the product being developed.

Assuming the ecosystem is important, developers can investigate it easily. Most RTOS vendors expose their ecosystem partners on their web site. Others will make such information available on request. Still others can offer "just-in-time-integration" to meet customer needs. In any case, this is worth exploring as part of any RTOS evaluation, but tempered with the realities of your actual project needs. Remember, making a decision based on speculative needs might result in the best solution for the problem you never face. Better to make sure you solve the problem you know you have, then to worry too much about what you MIGHT need down the road. Of course, the right balance between the two is best.

Q. **Thank you for this informative interview.**

GENERAL SOFTWARE: FIRMWARE AND BIOS ISSUES

19 November 2007: Firmware and Bios Issues

INTERVIEWEE. STEVE JONES

FOUNDER & CTO

TEL. (425) 576-8300

EMAIL. SJONES@GENSW.COM

COMPANY. GENERAL SOFTWARE, INC.

WEB. http://www.gensw.com/

Q. **First of all, tell us a little bit about yourself and your responsibilities at General Software.**

A. I founded General Software, Inc. in 1989, after leaving Microsoft's Windows NT kernel group, then still called the "Portable Systems Group". I've always been a technologist in the *X86* operating system space, having written several DOSes and OS clones for the PC (*Wendin-DOS, PCUNIX, PCVMS, Operating System Toolbox,* and *Embedded DOS*), as well as networking stacks (Microsoft's NETBEUI and TDI/DDI/PDI/NDIS architecture, and TCP/IP at General Software.) Altogether I've written a little over 1,000,000 of commercially published code; nearly all below the API.

At General Software I developed the first versions of our DOS, BIOS, real-time kernels, Flash file system, BIOS and SMM stack. As the company has grown I have transitioned to my CTO role, guiding us into new spaces, technologies, and solutions at the firmware level. Today that is our focus—firmware for every application: for servers; telecom/datacomm; targeted PCs like Microsoft's *FlexGo* and AMD's *PIC*; *UMPCs* and *MIDs* (Mobile Internet Devices); industrial and automotive; and even targeted mainboards, providing BIOS for AMD, Intel, VIA and other silicon vendors' reference boards and development boards.

Q. **General Software is not an RTOS company, but for this year's guide we are focusing on "beyond the RTOS" issues. Tell us what products you have that might be used as an alternative to an RTOS, or offer a totally different design perspective to the embedded developer.**

A. General Software's focus has always been developing and licensing system software for deployment on targeted *X86* applications—not IT-based equipment, but all the market segments mentioned earlier, especially embedded. Our IP portfolio includes about 10,000,000 lines of commercially-released code that gets leveraged across several product families, including *Embedded DOS*® (both real-time and ROMmable forms); *Embedded LAN*® (a real-time LAN for embedded systems); *Embedded BIOS*® (a BIOS adaptation kit with over 1,000 configurable options at the source code level today); and the *Firmbase*® *Technology SDK*, an adaptation kit that enables developers to write 32-bit firmware applications as portable executables using Windows development tools.

Over the last decade General Software evolved from a multi-layer solutions company to a single-layer one—we now focus exclusively on the firmware level, leaving DOS and OS to our partners. We host firmware cores, including BIOS, UEFI (Unified Extensible Firmware Interface), *FlexGo*, *EC* (Embedded Controllers), and others, on silicon supplied by AMD, Intel, SiS, VIA, *ServerWorks,* STMicroelectronics, and others in the X86 space as well as numerous EC vendors. We're really in the connection business, doing as much of the lifting as possible to put together products, support, and services that enable embedded device manufacturers (ODMs and OEMs) to get to market quickly with specialized firmware.

Today our primary product platforms are *Embedded BIOS®*, *Embedded UEFI™*, *Embedded ECCORE™*, *StrongFrame™ Technology*, and *Firmbase® Technology*. These elements provide a unified coverage of multiple firmware cores across a matrix of silicon vendors.

Of all our products, *Firmbase® Technology* comes closest to a real operating system in terms of functionality. *Firmbase® Technology* is General Software's operating environment that runs 32-bit portable executable applications within the insulating protection of System Management Mode (SMM), a third mode of the *X86* architecture that was originally developed by Intel to support Advanced Power Management (APM). These applications are written once, yet scale across chipsets and board designs because *Firmbase® Technology* abstracts the hardware, presenting an interface that includes objects familiar to *Windows NT* kernel developers like threads, timers, interrupts, deferred procedure calls, mutexes, events, spinlocks, file objects, device objects, a full I/O system, a dispatcher that manages IRQL, a trusted computing base, a high availability system, and a full C library—all in 80KB of ROM.

Firmbase® Technology includes a full USB stack with class drivers, a main USBD driver, and host controller drivers; a multi-homed TCP/IP stack with HTTP server, telnet server, SNMP agent, TFTP client, and SMTP (email) client; file system drivers, and device drivers. Architected interfaces through the I/O system provide an easy means to build on these components quickly using the C library and kernel functionality as glue.

When launched by a platform like *Embedded BIOS®* on an embedded target, firmware applications can be running within 500ms (half a second) of power on, and run with less overhead than a larger, more general purpose OS.

Although it is actually a real operating system in its own right (having memory management, scheduling, and an I/O system), *Firmbase® Technology* is not intended for use as an operating system by itself; it's a platform that can be used to develop applications at the firmware layer—adding value that complements operating systems like Windows and Linux, to name the major ones. Adding value at the firmware layer is a good thing for hardware developers looking to save material costs associated with the Bill of Materials (BOM) by virtualizing those components with SMM by writing a firmware application to do so. And, *Firmbase® Technology* creates a great opportunity for hardware manufacturers to differentiate their products by making them more capable then could be achieved with more limited mechanisms, like hardware watchdog timers for example.

Q. **As a vendor in the x86 architecture space, how do your products interact with the dominant Microsoft Windows XPe and CE product families?**

A. *Embedded BIOS®* with *StrongFrame™ Technology* is fundamentally a BIOS that can be configured by the embedded developer to build "perfect fit firmware" for a device. Sometimes the device needs to boot *Windows XP Embedded*, sometimes *Windows CE* or other Microsoft operating systems.

Single Board Computer (SBC) manufacturers need to be able to sell boards that can be integrated into special equipment by OEM customers; often these applications will use *Windows Embedded XP* or *Windows CE*. *Embedded BIOS®* has the ability to boot these operating systems from disk like a desktop BIOS, but can also load operating systems such as *Windows CE* from Flash or from just an NK.BIN file on any mass storage device in the system, allowing the developer to use a more generic *Windows CE* adaptation that leaves the hosting for a given chipset combination to the BIOS. Using this built-in *Windows CE* loader, an external one is eliminated, saving time and space.

Developers building their own hardware can configure *Embedded BIOS®* to load these operating systems directly, without going through a typical BIOS BBS configuration (although this is available if needed) and make the system boot directly and more quickly, a hot topic today.

Embedded BIOS® has UDMA (Ultra DMA) support built right into its disk services, whereas most desktop BIOSes provide basic disk I/O support. While it represents a larger lift in the BIOS core, it means that these Microsoft Windows operating systems load from disk at the full UDMA transfer rates supported by the chipset/drive combination, offering substantial boot time improvements.

Firmbase® Technology is also closely tied to *Windows XP Embedded*. Its High Availability (HA) Subsystem with High Availability Monitor runs concurrently and independently, in the insulating protection of System Management Mode, running from the first 500ms from power-on until the system powers off, even when the OS is still booting, or is running, crashed, or missing. HA Monitor watches the operating system's health using developer-defined policies to allow the firmware to take actions when Windows blue-screens, black-screens, or otherwise stops responding, or when applications don't load or might not be responding. Corrective actions include SNMP alerting over the network, and reprovisioning the OS components with a Provisioning application also supported by *Firmbase® Technology*.

Q. **And what about *Linux?* Is there any particular hook between General Software and developers who will be deploying Linux in an embedded application?**

A. With *Linux* gaining market share in embedded applications, we see lots of new design starts with all the different Linux distributions, and of course there is a migration from 2.4 kernels to 2.6 kernels that is largely complete. Even so, we continue to support customers on both platforms (2.6 requires ACPI support at the BIOS level, whereas 2.4 does not), because there are so many customers with differing runtime requirements.

As with *Windows*, our High Availability Monitor (HA Monitor) provides the ability to monitor *Linux* internals to ensure that it is operating properly. Panics, or general non-responsiveness, as well as failures to run specific processes can give rise to developer-defined actions that can involve alerting, reprovisioning, or other activities.

Q. **Quick booting time is a real headache for many embedded devices, especially those running *Windows* or *Linux*. What products do you offer that help with quick booting time?**

A. Surprisingly, though *Embedded BIOS*® has so many embedded features, Quick Boot remains the number one issue for developers of embedded gear. That's because boot time can be a gate to *market adoption* of a device. For example, I have a Media Center PC at home. I've got it recording my favorite TV shows, and hosting all my media, which is great. However, turning it on just isn't the same experience that turning on a legacy TV is—from my wife's perspective, the TV has to "boot", which means it's going to be technical and may need human intervention to start working, not to mention it's not going to be ready for a while. So, the "TV" doesn't get used so much when I'm not around—which is clearly an adoption problem.

If we think about consumer electronics like Media Centers or any embedded device, like a medical cockpit showing the patient's vital signs, the way we think about mainstream devices today—like telephones, toasters, or gym bags, we can see just how far away we are from building devices that people will readily adopt. Nobody today expects to have to "boot" their gym bag, or "start" their toaster—it's about the toast, not the toaster itself.

Quick Boot is about eliminating the idea of booting altogether. The user presses the ON button, and the device starts doing what it is supposed to do. There is no sneaky "waking up from a deep sleep" such as S3 or S4, which gives rise to complexity that gives the impression that the device was off and immediately comes on after the user presses the ON button. That complexity fools the user, which causes them to do things with the device that give rise to failure—like unplugging a device in S3 sleep mode, for example.

Embedded BIOS® is known pretty well in the embedded world for its ability to boot in less than one second on most of our roughly 80 chipset platforms. In some cases, the

time is a lot less—we have a demo of POSTing to a serial-based console in 85ms (less than 1/10th of a second) on a VIA CLE266-based PCI motherboard commonly available in computer stores. It does this by eliminating the work that a desktop BIOS would do that might be unnecessary in the context of the embedded device, and also by performing much of the work in parallel to eliminate waiting.

Quick Boot is changing the way developers think about how their devices should work in the hands of users. Years ago, developers were thinking splash screens with animation and sound (the equivalent of the Windows "Cylon-Eye" animation) was a pretty good idea while POST proceeded, bolstering the user's patience while the OS loaded. Today *Embedded BIOS*® POSTs far quicker than it would take to display the splash screen, making the splash screen the tall poll in terms of turn-on time for the user. And, with advanced UDMA features in disk support, the OS loads so quickly, there is little need for splash screens today, even though we continue to provide multimedia POST for some customers.

Q. **What about security? As more and more devices are connected, security is more of a problem. What do your products do to help a designer "embed" security into his product?**

A. Security in targeted devices (versus general computing products like desktop PCs) is becoming more important for their acceptance in the market, but it doesn't revolve around the traditional passwords that might be supplied to get into the setup screens of a desktop PC. Security in devices such as Voting Machines and Gaming platforms (like slot machines) centers around preventing hacks that cause them to perform differently for the user.

For example, an electronic slot machine might be captured by an attacker for a brief period, perhaps long enough to copy the system's hard disk with OS, applications, and data. Then, from the comfort of a private lab or in a thousand private labs scattered over the internet in a distributed approach, the attacker(s) run the application on simple desktop PCs, using debugging tools to analyze it for weak or defective operation. Later, when the system is installed, users in-the-know walk up to the device and defeat its weaknesses using normal system operation. What is needed here is to prevent the application from running without genuine hardware.

A second attack can be more straightforward—by somehow bypassing the physical security of the installation with human factors like distraction, the attacker replaces the hard drive or application with another one that simulates the operation of the real application, but with different results. Here, the hardware needs to not run unless the application can certify itself on a regular basis, without being vulnerable to replay attacks.

Both of these scenarios can be addressed with a cryptographic challenge handshake that takes place between the firmware within a device (such as General Software's *Firmbase*® *Technology,* running in SMM), and the application. This is exactly what General Software's Boot Security firmware application does for devices like the ones we discussed.

A more linear "chain of trust" idea is supported by organizations like the Trusted Computing Group's specifications, which solve similar problems which are not so dynamic. The notion here is that the system's configuration stays the same, so that once components are measured by the Core Root of Trust for Measurement (CRTM), they don't have to be re-measured. The arrow of trust is also forward-looking, so it doesn't go both ways. TCG is an important standardization for security in IT systems and information appliances, where it is gaining rapid adoption. As a part of supporting IT platforms as well as targeted devices, we have TPM support available in both our UEFI and BIOS products.

Q. **BIOS products have become more and more complex to the designer over time. Tell us about your *StrongFrame* initiative. What particular vertical applications or needs does it address and how?**

A. *StrongFrame™ Technology* is a build platform with the right structure and module interfaces that makes it possible to support multiple cores within the same framework. Firmware cores, such as BIOS, UEFI, EC, FlexGo, UMPC, and MID coexist as options within a single build environment. Project definitions include a project file, which selects the core for the project, along with the board support package, chipset package, CPU package, SIO package, and other components pulled together from sources like *Firmbase® Technology* (i.e., High Availability, Provisioning, Boot Security, and networking.)

So, once you've done the board support package for the BIOS, the work can be leveraged for UEFI, as well as the other cores. The real power of this is that a hardware manufacturer can switch between cores quickly, without a huge investment in learning about BIOS or UEFI or other proprietary technologies—one umbrella technology harnesses this complexity and allows the developer to control it with one development environment.

StrongFrame™ Technology also manages the complexities that arise when a BIOS or UEFI core vendor wants to release a new version of the same basic core, to fix bugs, optimize performance, or make feature enhancements to support new industry standards. The complexity comes about because the typical BIOS or UEFI development tree (such as TianoCore) place the board support code within the core directory structure, rather than as peers. Those build environments also don't support the notion that one project describing an already-shipped firmware build shouldn't stick with the old modules, while new projects should go ahead and use the latest module versions commercially available. By versioning project files, as well as the core and other modules, *StrongFrame™ Technology* makes all of the modules available to developers within one build tree, so that developers don't have to start new development trees for each new project or new version of a project.

Q. **Thank you for this interview.**

INTOTO: SECURITY FOR EMBEDDED NETWORKING

23 October 2007: Security for Embedded Networking

INTERVIEWEE. SATHYAN IYENGAR, CEO
TEL. 408-844-0480
EMAIL. info@intoto.com
COMPANY. INTOTO, INC.
WEB. http://www.intoto.com/

Q. **First of all, tell us a little bit about yourself and Intoto.**

A. For the past 15 years I've had various management, operations, business development and engineering roles in the networking, embedded systems and computer industry. I have a master's degree in Computer Science from the Indian Institute of Technology in New Delhi, India, and a B.E degree in mechanical engineering. I co-founded Intoto in 1998. Intoto is headquartered in Santa Clara and we're the leading provider of security software for enterprise and carrier networking equipment and multi-service gateways. We deliver high performance security and gateway software products to the OEMs building various datacom and telecom products including routers, switches, multi-service edge routers, security appliances and business gateways.

Q. **Intoto, obviously is not an RTOS company, so please tell us how your product offerings are beneficial to an OEM developer selecting an OS and related products for his product. What RTOSes / GPOSes do your products support?**

A. Intoto provides production ready security and gateway software products to OEMs and equipment manufacturers. Our unique *iGateway* architecture is scalable and extensible to span from the low-end to high end of the market, including security/convergence solutions for the residential gateway, SOHO, SMB/SME, enterprise, and carrier/service provider market segments. Our solutions are processor and operating system agnostic and can be easily ported to a new architecture. Our base development is done on *Linux* OS and can be easily ported to other RTOSs including *VxWorks*, *Nucleus*, etc. In addition, we support multi-core processors extensively and have dataplane (fast path) and controlplane implementations for security products to achieve high performance. We enable OEM customers with production ready solutions helping them achieve accelerated time-to-market and time-to-money.

Q. **What sorts of applications and products do you see the greatest need for better security? What do you mean by "security?"**

A. Any business having Desktops, Laptops, VOIP phones, Mobiles and any device connecting to the Internet and Intranet require protection from attacks such as phishing, exploits, Bots, Viruses and other threats. Any business providing content to the Internet community requires server protection from attacks and threats.

Network security mainly falls into two categories – Security from threats (Threat Management) and data security (Integrity and Encryption) for traffic on the wire.

Our focus is on delivering unified threat management for network security, which includes firewall/VPN, ID&P (intrusion detection and prevention), and gateway anti-virus functions. Other UTM capabilities often include content filtering, anti-spam, and anti-spyware. Going forward, more applications are being added to UTM appliances such as VoIP security and SSL-VPN. I think the twin drivers in this market will be the application of security technology to an ever-broader range of new applications (such as VoIP), and the constant need for security products that deliver higher performance at a lower cost. In short, ever expanding functionality and/or performance at an ever-

decreasing price. Which is why, for example, we have invested in developing a 16gig firewall (new performance benchmark) for multi-core general-purpose processors (low price point).

Q. **Does Intoto play only in the network infrastructure space? Or are your technologies relevant to other embedded areas that need security?**

A. Absolutely. Intoto mainly provides solutions in the network infrastructure space. Intoto's security products such as *Firewall*, *VPN*, and *Intrusion Prevention* have been integrated into Multi-service Edge Routers, Switches, Security Appliances, and Media Gateways targeting the enterprise and carrier infrastructure market. In addition, some of our technologies are applicable for mobile products and desktop products as well.

Q. **Many developers struggle with "buy" vs. "build." Tell us briefly why you think "buying" the Intoto solution is better than "building" one for one's self and own application.**

A. Two simple reasons: faster time-to-market and lower development costs. We're also a safe bet because our technology has been proven and vetted by top networking equipment customers and the industry's leading certification houses. We've done the development design work to ensure our software is portable to any processor, hardware or OS. Furthermore, we meet the most rigorous security standards for ever-changing threat environments and we deliver industry-leading performance.

Also, Intoto owns the IP of the product it licenses to the market and hence there are no GPL related issues. Intoto products are well supported and Intoto provides updates and enhancements to the product.

Q. **Intoto's security products often sit on top of *Linux*, yet *Linux* is an "open source" / GPL'd product. What are the advantages of your products over alternative "open source" security solutions?**

A. Open source solutions are not supported, or in some cases supported, but not to the extent an equipment manufacturer would like. Every customer has some additional requirements and Intoto is in the position to provide features in a timely manner working with the customer. In the case of open source, box vendors need to go through this process themselves and may not have experience in the technology. Also, equipment vendors may need to depend on a community for some critical problem fixes that may not be available in a timely manner. Open source may be free, but it has hidden costs that we believe are more than the cost of licensing software from companies like Intoto.

Customers come to Intoto for the features, functionality, scalability, high-performance and flexibility that we offer in our products. Intoto owns all the technology it licenses to its customers and does not include any GPL components.

Q. **Are your own products "open source"? Why or why not?**

A. Intoto security products are not open source. We invest heavily in R&D efforts to build and enhance the products and keep them current with the market requirements. We also bring value to our products by tuning them for performance and providing features for different deployments.

Q. **So how does an engineer purchase your products? What are the available purchasing models?**

A. Intoto licenses its products directly to the OEMs and equipment manufacturers. Customers pay a license fee for access to the software (binary or source code) and annual support and maintenance (optional). In addition, customers also pay a royalty per unit for every box that ships with Intoto software. We work closely with our customers to arrive at mutually beneficial business models.

MENTOR GRAPHICS – TECHNICAL CRITERIA FOR SELECTING AN RTOS

23 OCTOBER 2007: TECHNICAL CRITERIA FOR SELECTING AN RTOS

INTERVIEWEE. DOUG PHILLIPS, PRODUCT MARKETING MANAGER
TEL. 251-208-3400
EMAIL. douglas_phillips@mentor.com
COMPANY. MENTOR GRAPHICS, INC.
WEB. http://www.mentor.com/

Q. **First of all, tell us a little bit about yourself and Mentor Graphics, especially the company's RTOS offerings in brief.**

A. I have been working with the *Nucleus OS* for over eight years. I started with porting the OS to different CPU's. From there I moved into network protocol development and then ultimately managed an engineering team working on USB development. For over the past year I have moved into product marketing where I manage and market the *Nucleus OS* product line.

Mentor Graphics is traditionally viewed as an EDA company, where it is at the top of its class. However, for many years, Mentor Graphics has also provided tools to help the embedded software developer get their product to market more quickly. The goal of Mentor Graphics is to help developers of electronics from idea conception to final distribution. This includes software to develop hardware components as well as software to develop software components.

Focusing on the software development cycle, Mentor Graphics has three main products. The *EDGE Developer Suite* includes IDE, Debugger, Profiler, Simulator, Compiler, and JTAG connection. The *Nucleus OS* is a full featured real time OS that has been in deployment since the early nineties. Beyond the kernel, it includes networking, file systems, GUI development, and USB. The final product is *Inflexion* which provides application level software to help differentiate and deploy a GUI based product with ease.

Q. **What, in your view, would be the top three, most important technical reasons in evaluating a potential RTOS choice? Can you put them in rank order?**

A. As an engineer, it is easy to get caught up comparing features of one RTOS to another and how those features are implemented. The goal of which is to determine which RTOS appears to be the "Perfect" RTOS, or at least the best of the breed. This can lead to an endless cycle of feature comparisons, discussions with the RTOS vendors, and arguments with your peers.

I do agree that there are three areas of focus, maybe less technical than you would think. The first area is to understand what you are developing and if the RTOS can meet these needs. These needs are going to be specific to the development project at hand and cannot be prioritized by someone outside your development team. Such needs encompass the footprint, responsiveness, development environment, and the availability of the technology required to complete this project. If the RTOS does not fit your specific needs, throw it out and start looking at the next.

Since a company cannot sustain itself by creating one great product, usually a family of products is planned for. Even if a family is not planned, another development project will follow the current effort. This is why the second area of focus surrounds future products. Will the RTOS you are reviewing meet your future needs? Again, we look at the technical features that can only be prioritized by the development group. However, you must also take into consideration the availability of required technology in the future. Does the

RTOS vendor support this required technology, or do they plan to support it in the future? If an RTOS meets your current needs, but will not meet your future needs, it must be moved down the list of preferred RTOS's.

Lastly, you need to know if you can afford the RTOS. Now, I know you asked for technical reasons, so lets not talk about royalty vs. royalty free, instead focus on in-house expertise. Do you have the engineering staff that can address any technical deficiencies or issues? Does the RTOS vendor provide this support? Often times development efforts discover bugs in the RTOS, or require changes to the RTOS. Addressing these issues can either cost money up front, or throughout the development process.

Q. **We hear a lot about "hard" real-time vs. "soft" real-time. What are the technical criteria by which an engineer can judge whether an RTOS is "hard" or "soft"? Is this an important criterion in your view?**

A. With the constant improvement of hardware processing speeds, the line between hard and soft real-time has been blurred. A hard real-time system is usually looked upon as a system where, if a response is not supplied in a certain timeframe, a death will occur. This is the extreme requirement, but is one of the few cases that clearly mandate a hard real-time system.

Let's take an example of how hardware blurs the line between hard and soft real-time. If I put a 3 GHz processor in my deliberator, will Vista serve as a real-time OS? It pulls up notepad quickly when I need it. This may be a bit extreme, but shows how hardware can compensate for a system that is not real-time to make it appear as real-time.

So, let's get to the nuts and bolts of it. To be hard real-time, the system must be predictable. More than that, it must not have the ability to interrupt or delay the application. Note that I did not say, "it must not interrupt or delay the application" as if it retains the ability to do so, it is not a predictable system. This means the RTOS cannot disable interrupts as that can delay responsiveness. System calls into the RTOS must have a predictable amount for the execution time and must execute in a timely manner. This is usually described in microseconds, rarely in milliseconds.

Q. **OK, let's allow for a "shameless vendor plug." In your mind, what are the "technical criteria" by which Mentor's own RTOS (*Nucleus*) outperform its competitors?**

A. There is no one differentiator that makes *Nucleus* the number one operating system; instead it is a series of features that, when combined, allow *Nucleus* to stand out above the rest. In the RTOS world, many vendors tout a small footprint; we at Mentor Graphics are no different. We have kept in mind the cost of memory with every development effort that occurs. We have a very modular system where each module has the ability to strip out unused components.

The *Nucleus OS* has a fast response time. The operating system should be there to assist the application, this includes providing the application as much processing time as possible. By providing a good response time, the developer can choose a less powerful processor and still achieve what is required from the system.

Most importantly, the *Nucleus* RTOS is an operating system. Beyond the kernel, the *Nucleus OS* has over 53 networking protocols, a certified USB software stack with a wide availability of class drivers, a GUI framework that takes the developer from simple polygons to a windowing environment to 3D animated menus, and multiple file systems including support for FAT, ISO9660 (CD-ROM), Flash, and Power-loss safety. Whether the developer is producing a pacemaker, which requires only the kernel, or a smart phone that requires everything, Mentor Graphics has the technology you need today and is continuing to add to the OS so that we have the technology you need tomorrow.

Q. **Tools support. Tell me about the advantages (or disadvantages) of the related developmental tools that I get with some commercial RTOS. Some**

companies have closely integrated tool chains, whereas others seem to sell just an RTOS. How or why is this important?

A. Tools are a necessary part of development. With the right tools, development is comfortable, debugging a snap, and profiling is complete. With the wrong tools, more time is spent fighting the tools than fighting bugs. The benefit of getting tools from the RTOS vendor touches on both compatibility and convenience.

Though the hope is there to be able to compile software with any compiler, the fact is that compilers parse source code differently. A warning on one compiler is an error on another. Furthermore, with all the focus on creating the most optimized output, the potential is there for errant output. Move on to a debugger parsing the debug information from an unknown compiler and more issues can arise. Purchasing the tools from your RTOS vendor provides a piece of mind that guarantees that the software you purchases will work with the tools you have purchased.

It is always convenient to be able to purchase all of your software needs from one vendor. You will deal with a single salesman, a single customer support department, furthermore, you may be able to get a discount off the components since you are making a bigger order.

However, buying tools from the RTOS vendor should only occur if those are the right tools for the job. If the tools are not powerful enough, or overpriced, then they won't work for your development. The RTOS vendor should not be tying their RTOS to their tools; they should only be providing the tools as a source of compatibility and convenience.

Q. **Specifically, what is Mentor's *"EDGE Developers' Suite"*? How is this unique as a tools / RTOS offering?**

A. The *EDGE developer suite* covers many areas of development. It is based upon the Eclipse platform, which also provides the means of extending the tools even further.

The most basic component is the integrated development environment (IDE). This takes on many of the Eclipse primitives to provide not only an editor for development, but also a framework for building projects. Though we provide a compiler with the tools, any command line compiler can be integrated into this framework.

Once the developer moves past the initial implementation details, it is often best to prototype the final application. Within the Eclipse environment, the *EDGE Developer Suite* provides a simulator for this purpose. It allows for running the application on the desktop, where hardware is not a factor. However, it provides much more than just simple simulation as it also allows the developer to model the final application visually and hook that model back into the software. In other words, the modeled keystrokes will enter the application at the same point the hardware-implemented keypad does.

When the developer is ready to move to hardware, the *EDGE Developer Suite* provides a debugger interface in the same Eclipse environment. The debugger has all the features expected from a standard debugger. It also has some powerful scripting abilities, in the form of codelets that are only limited by your imagination. Furthermore, the debugger doesn't require the developer to use our IDE or compilers.

Lastly, once the application is up and running on hardware, it is time to optimize the system. For this purpose, we provide a profiler. Again, this exists in the Eclipse environment. The application can be profiled through a serial connection or an Ethernet connection. The data retrieved is graphed and can be reviewed to determine where bottlenecks are in the system. The profiler is also extendable so that it can track any data points the developer would like to add.

The goal of the *EDGE Developer Suite* is to provide known compatibility with the *Nucleus OS* and convenience to the user. Since it is based on the Eclipse framework, we are also able to reduce the price of the tools to something that is more palatable for development teams on a budget.

Q. **Most RTOSes are modular, yet almost all tout their "footprint" as a relevant metric in comparing them to each other. Since what goes in vs. what is out of the RTOS is variable, how useful is memory footprint as a tool in evaluating an RTOS? What do you recommend in trying to sort out these apples to oranges comparisons in terms of memory?**

A. As I mentioned earlier, comparing feature to feature between RTOSes can send you into a misguide. Footprint is a perfect example. If an RTOS vendor wanted to advertise how small its footprint is, it will most likely reduce the RTOS to its minimalist form. At this point there may be only a scheduler available. On the networking side you may end up with enough software for only router capabilities. This is not a useful RTOS in most situations.

To make reason from a footprint number, you must first qualify what is provided in that footprint. This leads back to what you need in the system. If you plan to use queues, be sure this was included, if you don't plan to use mailboxes, then don't focus on this. This may sound elementary, but it's amazing how quickly you can start comparing one RTOS to another and forget that you are trying to match an RTOS to your requirements.

I can think of two best-case scenarios when determining whether the footprint of an RTOS is right for you. In the first case, get a detailed listing of what the sizes are for each of the components in the RTOS. This should be a very specific list. How big is the queuing module? How much RAM will be taken up with each queue I create? Etc. If you have a general idea of your end product, you can get a general idea of how big the system will be on completion.

The second case if great if you are untrusting of the information that the vendor provides. Get an evaluation of the RTOS. Creating a quick sample application can allow you to go into the MAP file and pull out sizes for each of the components. Most likely they will have sample applications already available for you to link with.

The most important thing to remember is to go into and feature evaluation with a goal for your system. If your system needs to stay under 500KB, you will find quite a few RTOS's that match this criteria. Throw out the rest and move to the next requirement of your system. Don't compare RTOSes to each other just for comparison's sake.

Q. **"Multicore" and "virtualization" are big buzzwords nowadays in embedded and real-time systems. How important do you think they are from a technical point of view? Are they important to designers whose application "today" might not need "multicore" and/or "virtualization?" If so, how? If no, why not?**

A. Interesting question. Let's start with multicore. Is it important, absolutely? Multicore provides a proven solution to two of the primary issues that is facing many device manufacturers, particularly those who build portable electronics: performance and power consumption. The multicore solution uses two simple approaches to address performance as well as reducing power consumption. The first is division of labor and the second is specialization of labor. Adding cores brings more processing power to bear on the performance problem. It can also address the power consumption issue. If a problem can be subdivided so that several lower frequency, lower voltage cores can work on solving the problem together, then the total power used is less then using a unicore approach. Furthermore, since unicore processors are selected based on the maximum or peak load, their power consumption is not a function of their through put but of there peak performance. However, since that peak processing power is only needed infrequently or for short periods, the bulk of the high-frequency processing cycles occur in the idle loop. Using the multicore approach for peak processing and then halting execution of a number of cores during non-peak performance is another way to reduce power consumption.

From the specialization of labor perspective, homogeneous multicore systems have the advantage of being able to utilize specialty cores such as DSPs or Multimedia cores that can do in a few cycles that which may take an application processor hundreds of cycles to complete. Their power reduction is gained by doing things much more efficiently i.e., increasing the output/watt over an application processor.

From a textbook perspective, "virtualization" is the ability to run multiple types of operating systems on a single core. Also know as virtual machines, the virtualization process ensures not only the ability to execute multiple operating systems but also implies a level of compartmentalization between operating systems and or application spaces. This is useful on unicore systems because one can mix desktop or non-real-time type of operating systems such as Linux with real-time operating systems such as the *Nucleus OS* to ensure that the system as a whole meets its real-time commitments. The interaction and sharing of system resource is governed by what is called a *hypervisor*. It is similar to a master scheduler and arbitrates and controls which virtual machines execute and when. Traditionally, one sees the virtualizaion/hypervisor scenario on unicore devices where there are either the need for mixed OS environments such as the ability to reuse legacy code for one OS while migrating new development to a new OS that is better suited to on going development. Another senerio might involve a situation where security concerns that are best mitigated through the use of a virtual machine.

With the massive convergence and increase in connectivity we are seeing today it is difficult to predict whether or not someone will need the benefits of virtualization tomorrow. I do think that the ability to "virtualize" ones environment is less important if one is migrating from a unicore to a multicore processor. Multicore hardware can provide an "actual machine" as opposed to a "virtual machine" on a unicore setup. That being said, the hypervisor may be the key to providing a solution to mixed heterogeneous/homogeneous multicore systems. Currently, an SMP-based operating system can only perform load balancing on homogeneous systems. I personally believe that the multicore systems of the future will comprise large numbers of both heterogeneous and homogeneous cores combined with multiple types of operating systems that are geared to not only hardware to maximize performance but also minimize power consumption.

Q. **How important are software add-ons, such as networking protocol stacks, drivers, GUI development, etc., in the selection of an RTOS? Are there any parameters you can suggest on selecting these?**

A. I'm sorry, I'm a bit of a purest and this question bothers me. An RTOS without networking, GUI, file system, etc., is just a kernel. The kernel is only the base component of the operating system; it in no way makes up the operating system.

The importance of these components is again based on the system being developed. If I am developing a pacemaker, perhaps the availability of a USB stack is not so important. However, if you know of a medical station on the roadmap, USB becomes of key importance.

If more than just a kernel is required out of an RTOS, it is best to get those components from a single vendor. This allows you to work with a single salesperson, allow you to work with a single customer support department, and provides confidence that all the software components will work together seamlessly. Hmmm, this sounds vaguely familiar.

Now I know that no RTOS vendor can provide every piece of software you may need for your project. At that point you need to look at building the component yourself, or finding a pre-made component. Key to the RTOS vendor is not just the RTOS they provide, but the ecosystem of pre-integrated software partners they have. Before you purchase the RTOS be sure that all the software components are available from either them or a trusted partner of theirs.

Q. **Specifically, what is Mentor's "*Inflexion*"? Is this a GUI solution only for mobile handsets and phones? Or is it (or will it become) a more broadly relevant GUI solution?**

A. Mentor Graphics' *Inflexion Platform* is more than a GUI solution, it is a software framework designed to make it much easier for manufacturers of consumer devices to deliver more compelling products in less time. It is built around a new menu-driven UI paradigm, aimed specifically at addressing the interface requirements of products such as mobile phones, portable medical devices, personal media players, TV set-top boxes and in-car entertainment systems.

The *Inflexion Platform's* key benefits are as follows: you can put together completely new interface designs (or modify existing ones) without making any changes to your software stack; you can easily exploit the full potential of whatever hardware you target (including 3D acceleration where present); and you can develop applications as presentation-independent 'data engines' that are far easier to reuse and iteratively enhance across products.

Q. **Thank you for this interview.**

MICRIUM *µC/OS-II*: BENEFITS OF A COMMERCIAL RTOS & UNIQUE LICENSING

18 October 2007: MICRIUM *µC/OS-II*: BENEFITS OF A COMMERCIAL RTOS & UNIQUE LICENSING

INTERVIEWEE. JEAN LABROSSE
PRESIDENT
TEL. 954-217-2036
EMAIL. Jean.Labrosse@micrium.com
COMPANY. MICRIUM
WEB. **http://www.micrium.com/**

Q. **First of all, tell us a little bit about yourself and Micrium.**

A. I am Jean Labrosse, Micrium's founder, president, and I regularly speak at the Embedded Systems Conferences. I hold a MSEE degree, and have been designing embedded systems for over 25 years. I authored two books: *MicroC/OS-II, the Real-Time Kernal* and *Embedded Systems Building Blocks, Complete and Ready-to-Use Modules in C*. My articles are published in several technology magazines.

Micrium was founded in 1999 with a mission to shorten the time-to-market for all product development cycles by providing very high quality embedded software components in source-code form.

Micrium offers a full portfolio of embedded software components including *µC/TCP-IP* (an embedded TCP/IP stack), *µC/FS* (an embedded file system), µC/GUI (an embedded Graphical User Interface), *µC/USB* (a bulk and mass-storage device stack), *µC/CAN* (a CAN-bus framework), *µC/Probe* (a universal run-time data monitor) and more.

Micrium's real time operating system (RTOS) product, *µC/OS-II*, is provided in source form on a CD with my book. The *µC/OS-II RTOS* is licensed for use in thousands of commercial applications worldwide. Unlike most competitive products, *µC/OS-II* is certified for use in Avionics (DO178B Level A) and Medical (510(k)) applications.

Q. **Micrium seems to be an unusual company in that you release your source code on the Internet, for free, and seem to "trust" that honest companies and individuals will re-contact you for commercial deployments and licensing fees. Your "marketing philosophy" seems unique. Can you tell us about it in a nutshell?**

A. Micrium operates as a hybrid between an open source and fully commercial model. Embedded software components are increasingly complicated and expensive. Engineers appreciate being able to try advanced software before they commit to purchasing it. Once they see how quickly our software runs on their platform and the quality of the products they achieve, it is easy for them to justify purchase to their management. We offer a 45-day try-before-you-buy program on several of our products, which is a sufficient period to make a decision. Colleges and Universities can use *µC/OS-II* and *µC/TCP-IP* for free.

Q. **Drilling a bit deeper here, can you give us the details of how your RTOS, *µC/OS-II*, is licensed for commercial projects? Is it per developer, per project, per microprocessor? Is this an "open source" license, hybrid, or something else?**

A. Our software is licensed on a per-end-product basis. A single product license allows a user to embed a Micrium component into one end product, which can be manufactured in any quantity, for the life of that end product. It is a royalty-free license, and a separate license is required for each different model that embeds the Micrium component, even if

the end product is in the same family as a previously licensed product (i.e. a different model number or processor). If the customer considers it a separate product...so do we.

The license is specific to the processor used and the name and/or model of the end product. Examples include any single model of a TV, HDTV, VCR, DVD player, network switch, network router, etc. If different only by color it's considered the same end product. We also offer other licensing models such as a "product-family" for companies that manufacture a number of different products that are similar (e.g. washing machines, microwave ovens, MP3 players, etc.). The product family license allows a holder to make any number of different products within the identified family, in any quantity, for life, irrespective of the CPU used. A per-CPU-type license, which allows companies to manufacture different products around the same type of CPU, is also available.

Q. **How is your license different from that of Linux? And, what about QNX, which recently announced a "hybrid" model for their own RTOS, *Neutrino*?**

A. Micrium has used its 'hybrid' model for a long time, and finally began calling it by that name approximately two years ago. Our licensing model is commercial. Our clients purchase the 'rights' to use our software in their product.

Unlike QNX, Micrium's philosophy is to not only put all of the necessary pieces together for the developer, but also to tell them how and where everything fits. We provide application notes that indicate how a particular port works, how to run code on a large number of evaluation boards, and more. We don't "dump" code on a customer's computer and let them struggle to figure things out.

In comparison, the *Neutrino* download consists of more than 56,500 files--more than 8,000 source files with little documentation explaining how everything fits together. Providing source code is one thing; being able to understand it is something else.

Q. **Many developers these days use embedded Linux. It is "free" to download, has an open source license, and no royalties. In what "business" ways is *µC/OS-II* a better choice than *Linux*?**

A. *Linux* code itself is "free" but it's well known that Linux comes with a huge learning curve. Isn't that actually expensive? Companies such as MontaVista earn millions of dollars each year supporting *Linux*. Unless one is a *Linux* guru, it takes weeks and possibly months to get *Linux* up and running.

If a company must get a product to market quickly, there is no faster and less expensive solution than *µC/OS-II*. It is easy to have *µC/OS-II* up and running on most CPUs in anywhere between 10 minutes and just a few hours. Micrium ports *µC/OS-II* to nearly all popular CPUs and evaluation boards, simplifying the process even further.

Linux is also known to require substantial resources (faster CPU, more RAM, more Flash, I/Os, etc.) and these translate to potentially higher recurring costs for a product. While it's true that royalties are not paid to a software supplier, royalties (i.e. chips) are paid to a hardware supplier. Also, forget about using *Linux* on most single chip controllers (it just won't fit), which is a segment that account for a large portion of the high-volume market.

µC/OS-II can be used in safety critical application because it has been certified for avionics (DO178B Level A) and medical use (510(k)). It would literally cost millions of dollars to just attempt to do this with *Linux*--which means that it won't happen.

Studies indicate that cost of ownership of *Linux* is much higher than just about any commercial RTOS solution. The market is like a pyramid; *Linux* covers applications at the top of the pyramid and *µC/OS-II* at its base. There are a lot more applications that are ideal for *µC/OS-II* compared to those where *Linux* could be used.

Q. **In what technical ways does *µC/OS-II* differ from embedded Linux? Where are its technical advantages?**

A. *µC/OS-II* boots in milliseconds, *Linux* takes tens of seconds. *µC/OS-II* is a real-time operating system, *Linux* is not. *µC/OS-II* excels in applications where the application requires fast and deterministic response.

Whereas *µC/OS-II* runs on 8-, 16-, 32- and 64-bit processors and most DSPs, *Linux* requires at least a 32-bit CPU. *µC/OS-II* has been ported to well over 45 different CPU architectures. *µC/OS-II* requires between 6K and 24K bytes of code space whereas *Linux* requires hundreds of Kilobytes (assuming just the kernel).

Q. **Every RTOS seems to claim it is the smallest, "hardest" real-time, etc.. What factors do you see as technically superior in *µC/OS-II* vs. other commercial RTOS choices besides *Linux*?**

A. Most RTOS offer the same capabilities and functionality. There many factors that come into play that makes *µC/OS-II* a better choice for most applications:

- The fact that *µC/OS-II* is certified for avionics and medical use
- The large number of ports (8-, 16-, 32-, 64-bit and DSPs)
- The superior performance of *µC/OS-II*
- The quality and availability of the source code
- The quality of the documentation
- Its low cost
- The large number of application notes targeted to different CPU architectures
- How fast the user is up and running
- The reputation of the RTOS
- The quality of the support customers gets from Micrium
- The fact that you can try-before-you-buy
- Micrium's outstanding reputation
- The *µC/OS-II* book that describes its internals
- etc. etc.

Q. **Micrium offers a lot of additional software such as TCP/IP, file system, GUI, etc. - tell us about these software "add-on's" on a technical level.**

A. *µC/TCP-IP* operates under the same commercial strategy as *µC/OS-II*. The source code is available for evaluation. For all other products, the source code is only available with a license.

µC/FS is a FAT file system (we offer a FAT-free version also) for embedded applications, which can be used on any media with basic hardware access functions. *µC/FS* is a high performance library optimized for minimum memory consumption in RAM and ROM, high speed, and versatility. It is written in ANSI C and can be used on nearly any CPU.

µC/GUI is designed to provide an efficient, processor and LCD controller-independent graphical user interface (GUI) for applications that operate with a graphical LCD. It's compatible with single-task and multitask environments, with a proprietary operating system or with any commercial RTOS. *µC/GUI* is shipped as "C" source code. It may be adapted to any size physical and virtual display with any LCD controller and CPU.

µC/USB Device works on any embedded system with USB device controller. Ports for most common USB devices are available. It can be used with USB 1.1. or USB 2.0 devices.

µC/USB-Host uses a layered architecture as defined by USB 2.0 Specifications. It is modular in design and easily adapts to different CPU, RTOS (Real-Time Operating Systems), and USB Host Controllers.

µC/CAN is a CAN protocol framework, which enables easy and clean implementation of CAN communication paths. *µC/CAN* is a source code library optimized for speed, flexibility and size. High portability and clean documentation were key design goals.

µC/Modbus-S is a Modbus Slave (i.e. server) software module that provides the capability to make your embedded system communicate to a Modbus Master (i.e. client). *µC/Modbus-M* is a Modbus Master (i.e. client) software module that provides the capability to make your embedded system communicate to any number of Modbus Slaves. The modules are written in ANSI C and are highly portable. *µC/Modbus* supports both ASCII and RTU protocols and, RS-232C and RS-485.

Q. **What is the licensing arrangement on the TCP/IP stack, file system, GUI, etc. Are these also free to download, and does one pay only upon commercial deployment? Are there royalties?**

A. All products have the same licensing model with the exception of *µC/Probe*, which is licensed on a per-developer-computer basis. *µC/Probe* is a tool versus the other products, which are embedded software components (more on *µC/Probe* below).

Q. **One of the themes for this edition of the "Insiders' Guide: RTOS" is tools. You recently launched a new tool, *µC Probe*. This also won "best of show" at the *Freescale Technical Forum*. Tell us briefly about this tool.**

A. *µC/Probe* eliminates the need to stop an application to get system feedback. It saves considerable development time by visually allowing users to see the internals of a running embedded application. As a result, developers can ensure that the system is working properly or immediately identify system instabilities visible only when the system is live.

µC/Probe works with compilers, any 8-, 16-, 32- and 64-bit CPUs as well as DSPs and, can be used with any tool chain that generates an ELF/DWAFF or IEEE695 file, eliminating the need for custom programming or scripting. Data is displayed graphically on a PC running Microsoft Windows, and values can be numeric or shown as gauges, bar graphs, plots, graphs, LEDs, counters, pie charts and more. *µC/Probe* does not require users to write "any" code and can operate with or without a real time operating system (RTOS). The product interfaces with hardware targets via J-Tag, RS-232C, TCP/IP, USB and Renesas emulators through HEW Target Server for data collection.

Until *µC/Probe*, there has been no significant improvement in the way developers examine the run-time behavior of their system (LEDs, printf() statements, character displays, graphics displays, etc.). All of these prior techniques needed that the developer spent time and resource 'instrumenting' their code. With *µC/Probe,* there is no code instrumentation, no time spent setting up test code and temporary hardware to 'see' what's going on, etc. It takes seconds to examine/display/plot the run-time value of just about any variable or memory location in a system.

µC/Probe is a product that is as fundamental and universal as text editors are to writing code. *µC/Probe* can not only be used by developers but also marketing people, field service technicians and management to demonstrate the capabilities of their products.

A fully functional, 30-day limited version of *µC/Probe* is available from Micrium's website.

Q. **What is the interrelationship between your RTOS and *µC Probe*? Can one use one without the other?**

A. They are totally independent; one doesn't need an RTOS to use *µC/Probe*. *µC/Probe* works just as well with or without an RTOS. It works with virtually any CPU and just about any tool chain (compiler/assembler/linker).

Q. **You are known in the industry as one of the "guru's" of RTOS. What advice would you give to an engineer who is seeking to figure out which RTOS is best for his project? What criteria would you recommend he list to compare and contrast his available RTOS choices?**

A. There are many factors to consider when selecting an RTOS and well over 150 RTOS to choose from. Much depends on the goal, and the market segment. Examine recent RTOS surveys and contact the top 5-10 vendors to see what they have to offer. Here are some questions you might want to ask:

- Is it important that the RTOS be third party certified? Is it certified for: Avionics (DO178B Level), Medical (510(k)), Nuclear, IEC, SIL, etc.?

- Is the RTOS ported on the CPU you're planning on using?

- What is the scheduling policy of the RTOS? Most are preemptive.

- What interrupt model does it use? Unified or segmented?

- Is the source code available and is that important to you? If so: Ask for samples to see the quality, see that there is a description on how the code works, and note that needing 8,000 source files doesn't mean it's a better product!

- How's the quality of the documentation?

- How much does it cost?

- How is it licensed? Is it royalty-based or royalty-free?

- Get references from other customers

- Can you try the code before you have to buy it?

- How long has the vendor been in business and what is their reputation?

My most important piece of advice--Don't make the mistake of writing your own RTOS. While it may appear both simple and tempting, it takes months and even years to write a solid RTOS, document it, port it, etc.

Q. **Why do your customers decide on Micrium over the competition?**

A. While they appreciate the clean code that Micrium consistently offers, our try-before-you-buy model, and the wealth of documentation, what is consistently communicated is a need for the level of support that Micrium provides. Most companies come to us after using competitive products and running into a myriad of usability issues in the process. Once they arrive, we have an extremely impressive track record of keeping our customers satisfied, and providing additional products that they need.

Q. **Thank you for this interview.**

MICROSOFT .NET MICRO FRAMEWORK: A NEW APPROACH TO EMBEDDED SOFTWARE

23 October 2007 –.NET Micro Framework: A New Approach To Embedded Software

INTERVIEWEE. COLIN MILLER
PRODUCT UNIT MANAGER, .NET MICRO FRAMEWORK
TEL. (425) 703-4530
E-MAIL. colin.miller@microsoft.com
COMPANY. MICROSOFT CORP.
WEB. http://www.microsoft.com

Q: **First of all, tell us a little bit about yourself and Microsoft, especially the company's broad "embedded strategy."**

A: I'm the product unit manager for the *Microsoft .NET Micro Framework* and am responsible for driving Microsoft Corp.'s initiative to deliver a developer platform for extremely small devices. In this role, I manage business plan development, group partnerships and the overall strategic direction of the.*NET Micro Framework platform.*

Changing the paradigm of application development on small, resource-constrained devices is something that I'm very passionate about. The *.NET Micro Framework* opens embedded development to a greater number of developers, a lot of whom never considered making a device before because they didn't have the specialized skill set or access to proprietary tools. To me, it looks a lot like the early days of PCs, where anyone could have a great idea and bring it to market without a huge staff or a ton of venture capital. It's my hope that the *.NET Micro Framework* will touch off an explosion of innovation in small devices.

Microsoft's broad embedded strategy is to deliver adaptable and scalable platforms for connected devices that enable rich applications and services. Our Windows Embedded product family supports the extremely diverse needs of the embedded device market segment. An important part of this is making sure that embedded developers can use the same powerful tools and technologies to build applications for a broader range of devices. That way they can continue to apply the skills they already have and don't have to learn a new development environment, API or language just because they are moving to a smaller class of device.

The *.NET Micro Framework* is .NET for even smaller devices than what *Windows Embedded CE* and *.NET Compact Framework* can support. Providing a real .NET "feel" was an important design goal for the *.NET Micro Framework*, second only to the goal of making the smallest possible *.NET* runtime.

Q: **What is the *.NET Micro Framework*? What is its value proposition for "embedded systems" designers?**

A: The *.NET Micro Framework* is an implementation of the *.NET* common language runtime (CLR) that runs on small, low-cost devices constrained by such resources as processor speed, memory and battery power. It is not an operating system in the traditional sense, but a "bootable runtime" with some of the features traditionally provided by an operating system. It can run directly on hardware, not just on top of an existing operating system or kernel, though some of our customers are in fact using it with an operating system.

For embedded systems designers, the main value proposition of the *.NET Micro Framework* is increased developer productivity. That means you can lower development costs, or get to market faster, or provide more functionality for a given budget and

schedule — you can trade off among these benefits depending on your goals. Also, you can choose from a much larger pool of *.NET* developers rather than requiring developers with highly specialized skills.

Small-device development is very similar to where desktop development was about a decade ago. A lot of the programming is still done in C or even assembly language, and the tools can be primitive. The whole concept of managed code, which helps improve security and reliability, largely hasn't reached small devices. Our value proposition arises directly from Microsoft products and technologies, including *Visual Studio* and *.NET,* which were created to make it easier to develop software. With embedded processors becoming more powerful with the lower price and power consumption, it's now possible to bring these solutions to small-device development. As desktop development paradigms continue to advance, we will continue to look for ways to bring the same level of productivity to embedded development.

But that's not all that the *.NET Micro Framework* brings to the table. The *.NET Micro Framework* is essentially a virtual machine, providing a high level of abstraction from the hardware — not just the processor but also support chips such as graphics controllers and I/O ports. This makes it possible to use the same application code on a wide variety of embedded hardware. The *.NET Micro Framework* also offers a managed driver model, which means that as long as a device is connected by an interface supported by the platform (there are a number of these, including I²C, SPI, USB, serial and GPIO), you can write the actual device driver in C#. That gives you the productivity advantages of managed code and further improves hardware-independence.

Q: **What sort of "end applications" do you feel are ideal for the *.NET Micro Framework*? What sorts of embedded systems designers and what sorts of projects should be interested in this technology?**

A: We are seeing a lot of interest from home automation and sensor network developers, and we also feel the *.NET Micro Framework* is a good match for healthcare devices, informational and retail kiosks and point-of-sale devices, logistics systems, and low-cost consumer electronics. We've already seen the *.NET Micro Framework* used on *Microsoft TV* set-top boxes, and it's also the platform for many *Windows SideShow*-compatible devices. In fact, Microsoft offers a software development kit (SDK) based on the *.NET Micro Framework* specifically for SideShow. The one area where the *Micro Framework* by itself is not appropriate is applications that have very rigid real-time requirements. In many of those cases, the *.NET Micro Framework* can be paired with a real-time operating system (RTOS) to provide both a great applications development environment and the real-time needs of the application.

Getting away from specific verticals, I think the *.NET Micro Framework* is a great platform for inventing new kinds of devices. One of the biggest problems with innovation in the embedded space is that the projects take so long to get to market. The risk of starting a new project that won't make it into the market for two years is very high. But if you can cut your development time significantly, that allows you to take a few more chances with new applications of embedded technology, which dramatically increases the opportunities for innovation. I see a lot of potential in low-power wireless networking, for example — devices that will detect when they're in the proximity of another wireless device and exchange some data. The *.NET Micro Framework* actually has its roots in an earlier Microsoft initiative to build smart, connected devices, so you could say this idea is in our DNA.

Q: **How is the *.NET Micro Framework* positioned vis-a-vis Microsoft's other embedded products, such as *Windows XP Embedded, Windows CE* or *Windows Embedded for Point of Service?***

A: In a nutshell, *Windows Embedded CE* is a full operating system well suited to multifunction devices with rich user interfaces. It also has hard real-time capability. The *.NET Micro Framework* is a "bootable runtime" that provides a small-managed

environment suitable for single-purpose or limited-function devices. The *.NET Micro Framework* is the smallest embedded platform in the *Windows Embedded* family and can run on even smaller devices with fewer resources than *Windows Embedded CE*.

With the *.NET Micro Framework*, we re-imagined what *.NET* would look like on a device with only a few hundred kilobytes of memory. Our primary design goal was to provide the essential functions an extremely small device would need because we built it for small devices that we were building. The result is that the *.NET Micro Framework* is a useful foundation that developers can use to build their device-specific functionality. The goal of *Windows Embedded CE*, on the other hand, is to provide a rich user experience by bringing much of the Windows user interface and technology to portable and consumer electronics devices. These differences in goals drive the technical differences. For example, the *.NET Micro Framework* can run with or without a memory management unit (MMU), because per-unit cost is critical for extremely small devices. *Windows Embedded CE* supports many more processors than the *.NET Micro Framework*, which focuses on the most popular processor families for small devices.

A kernel-only configuration of *Windows Embedded CE*, offering only a Win32-derived API, has about the same footprint as the *.NET Micro Framework*. *Windows Embedded CE* also supports a managed environment, the *.NET Compact Framework*. By the time you add the *.NET Compact Framework* and the additional components needed to support it, *Windows Embedded CE* has a footprint about 40 times the size of the *.NET Micro Framework*.

Q: **How is the *.NET Micro Framework* like a traditional RTOS for small, resource-constrained devices? How is it different?**

A: The *.NET Micro Framework* is not a traditional RTOS, though it does have some operating system features. Because it's a managed environment, the system will sometimes need to perform garbage collection to reclaim memory that is no longer being used. That makes it nondeterministic, so it is not possible to guarantee interrupt response latency.

The *.NET Micro Framework* can run on top of a host operating system or kernel, which can be real time. This approach would let you write time-critical code to run directly on the underlying RTOS, while still taking advantage of the rapid development capabilities of the *.NET Micro Framework* for the rest of your application. Using a separate processor for the time-critical functions could be another fruitful approach.

Q: **What is the relationship between the product and the *C#* programming language? What advantages might *C#* bring to embedded systems?**

A: *C#* is the only programming language currently supported by the *.NET Micro Framework*. We may support other *.NET* languages, such as Visual Basic, in the future as demand warrants.

Because its syntax is similar to *C*, embedded developers should find *C#* comfortable. It's a fully modern object-oriented language, and it gives developers all the advantages of *.NET*, including managed execution, automatic memory management, and access to the *.NET* libraries, a subset of which is included as part of the *.NET Micro Framework*. They'll have a few new things to learn if they've never used *.NET*, but they'll enjoy the power and ease of developing with the *.NET Micro Framework*.

In addition, most experienced *.NET* developers have used *C#*. So our use of *C#* means they can get up to speed building *.NET Micro Framework* applications quickly.

Because the *.NET Micro Framework* is integrated with *Visual Studio*, developers can create a new *.NET Micro Framework* project from one of the templates we provide as easily as they can create any other type of *Visual Studio* project. The language and the libraries will be familiar to them. I encourage *C#* developers to download the SDK and

try their hand at embedded programming. You might just find yourself with a new marketable skill.

Q: **How is the *.NET Micro Framework* situated with respect to Microsoft's development tools such as *Visual Studio*?**

A: *.NET Micro Framework* support for embedded devices in *Visual Studio* goes far beyond editing code and building an application. The *.NET Micro Framework* supports full emulation of a device on a PC, allows deploying *.NET* assemblies to the device, and permits debugging the code running on the device — all from within *Visual Studio*.

Initial development can be done on the PC using an emulator, which allows application developers to write much of their code without a development board. The emulator can be configured by modifying an XML file (to change characteristics such as screen resolution) and can also be extended with custom components written in *C#*, so you can simulate I/O devices that your target hardware will have.

Q: **How much does it cost? What sorts of purchase models are available?**

A: Microsoft's strategy is to make development as affordable as possible by allowing device makers to defer licensing costs until their products ship, so the *.NET Micro Framework* SDK is available for download at no cost. The SDK requires *Visual Studio 2005* (Standard Edition or better) and includes all the deployment, debugging and device emulation features that developers need to develop applications for the *.NET Micro Framework*. There are no additional fees for developing *.NET Micro Framework* applications beyond the required *Visual Studio* license. Distributing devices containing the *.NET Micro Framework* runtime, however, requires a distribution agreement and a per-device licensing fee, which varies by expected volume.

Q: **Where specifically on the Microsoft Web site can a designer find more information? Are there free downloads or other educational content to help one investigate if this is a good solution for a particular embedded project?**

A: The *.NET Micro Framework* SDK is available for download at no cost. For more information on the *.NET Micro Framework*, please visit us on MSDN at http://msdn2.microsoft.com/en-us/embedded/bb267253.aspx

QUADROS SYSTEMS: ADVANTAGES OF A COMMERCIAL RTOS

19 November 2007: Advantages of a Commercial RTOS

INTERVIEWEE. STEPHEN E. MARTIN
V.P. SALES AND MARKETING
TEL. 781-210-2393
EMAIL. stephen.martin@quadros.com
COMPANY. QUADROS SYSTEMS, INC.
WEB. http://www.quadros.com/

Q. **First of all, tell us a little bit about yourself and your responsibilities at Quadros.**

A. I am vice president of sales and marketing at Quadros Systems, responsible for customer engagements, ensuring we have the products they need for successful embedded systems development.

Q. **As a commercial RTOS vendor, Quadros certainly has an opinion about the advantage commercial RTOSes have over "build your own." What do you see, in brief, as the primary advantages?**

A. For those who have never used a commercial operating system there is still some misunderstanding about the reasons to use a third-party RTOS. There are several important reasons to consider using one.

1. Minimizes complexity. Today's embedded processors offer more integrated functionality than ever before. The OS abstracts away the complexity of the processor and I/O. Your application interfaces to the HW via a powerful API that simplifies your application. It also makes it easy to plug in and support the necessary communications stacks and middleware (TCP/IP, USB, CAN, file systems, etc.)
2. Eases Development. The OS provides a solid infrastructure of rules and policies that provide consistency and repeatability.
3. Optimizes use of system resources; provides deterministic responsiveness. A preemptive, prioritized real-time OS
4. Improves product reliability, maintainability and quality

We still see some customers maintaining a legacy "house kernel." Although in the 32-bit space it is rare to see a customer decide to write a new kernel or scheduler. The systems are too complex, there are too many good kernels available and the cost to maintain a house kernel is too high. There is a definite trend of customers who are abandoning house kernels.

Q. **There are lots of RTOS choices out there from small, focused RTOSes to general purpose RTOSes to safety critical certified, DSP, and on and on. What is your take on the "technical criteria" for choosing an RTOS? How do you recommend a customer proceed from short list to RTOS decision in terms of technical criteria?**

A. The embedded systems industry is wide-ranging and highly diverse. Not every RTOS solution is going to be the best fit for a given application. On top of that, embedded engineers have their own preferences for what they are comfortable with. The result is a confusing throng of RTOS options.

First, you need to acknowledge that there are significant differences between RTOSes—they are not all created equal or with the features you will need for your project. In selecting an RTOS you need to look at technical considerations, availability of communications stacks and middleware, and support.

1. Technical criteria. At Quadros Systems we have a strong point of view that the RTOS should adapt to the needs of the application. In other words, you shouldn't have to bend and twist your application to fit the limitation of the operating system. Our approach is to offer a family of kernels with a common API that allows you to choose the processing model that best fits your needs—control, data plane, convergent (dataplane and control) or multicore. (In fact, the name Quadros denotes that we have four real-time kernels.) All of our kernels are small footprint, highly responsive, and are fully deterministic.
2. Stacks and middleware. Most customers today are looking to source their stacks and middleware from the RTOS supplier. This lowers their risk of having to integrate software products from different vendors into a single system. Quadros Systems does extensive integration and testing of all of our software and delivers it to you as a working project that you can then adapt and build on.
3. Licensing issues. Royalty-free, open source, deployment-only, royalty-based; there are a myriad of options being offered. How do you decide what is right for you. In the same way that a single RTOS solution cannot fit every application, a single licensing model is not right for every customer. At Quadros Systems we have a flexible model that allows us to customize a licensing solution to fit your needs.
4. Support. Our customers will tell you that we give the best support in the industry. At Quadros you are supported by experienced development engineers who can quickly get you up and running with your project.

Q. **Tell us briefly about Quadros' products. What are your specific strong suits?**

A. As I said in the previous answer, too many people in the industry believe there is no difference between RTOSes. Quadros Systems offers a number of distinct advantages to embedded developers.

1. An RTOS family that allows you to chose the processing model/configuration that best fits the needs of your application. No one else in the industry has an RTOS family with this scalability built in.
2. RTOS experience that is second to none (our team has been working with real-time kernels for more than 30 years. *RTXC Quadros* is a second-generation kernel that builds on years of use in thousands of applications.
3. Superior knowledge of processor architectures, which leads to extremely tight integration of RTOS to hardware.
4. A wealth of communications stacks and middleware that is integrated, running and tested on the target.
5. A new visual design tool that dramatically shortens the development process.
6. Outstanding, timely support.

Q. **Are there specific applications or industries that Quadros focuses on?**

A. The long history of the *RTXC* RTOS has given us experience in a wide range of industries. We support world-leading customers in industries such as telecom/datacom, consumer electronics, medical, aerospace, industrial control, automation, and test & measurement.

Q. **Many developers feel that choosing a commercial RTOS will "add complexity" to their design process as opposed to keeping it all in-house. Why do you feel that is a wrong conclusion?**

A. A commercial RTOS is almost always a better choice than an in-house kernel because it is proven, reliable, supported and scalable. An RTOS is also less expensive to maintain (for instance, how do you address bug resolution in a house kernel? What level of internal staffing is required?) There are obviously exceptions, but if you are looking for a responsive system that is easy to build and easy to maintain, it will be hard to beat a kernel family such as the *RTXC Quadros* RTOS.

Q. **Recently Quadros introduced *VisualRTXC*. Can you tell us a little bit about this?**

A. We created the *VisualRTXC* development tool to help engineers using *RTXC* shorten their software development process. It provides the efficiency of a high level design tool with a practical, real-world implementation that fits neatly into any software development lifecycle, focusing on three key areas: software architecture, detailed design and code development.

There are many sophisticated modeling packages (primarily UML-based) that take a major monetary and time commitment to learn, implement and maintain. *VisualRTXC* is intuitive, using dataflow diagrams and standard flow chart conventions, already familiar to any design engineer. Our belief is that a tool like *VisualRTXC* can be effectively used by any engineer on the team, not just by one or two system architects.

This powerful design tool allows the developer to move rapidly between design concepts and generated C code shaving weeks off of the typical, hand-coded development process.

Q. **How does *VisualRTXC* help with the migration to 32-bit architectures?**

A. Developers migrating from 8- and 16-bit architectures to a 32-bit are moving into a different world. We know from talking with many such teams that there are legitimate concerns about this transition. An intuitive, graphical environment such as *VisualRTXC* that is tightly coupled to the *RTXC* Quadros RTOS can shorten the learning curve and reduce coding errors. You can also create reusable code components that can be applied to future code migration on other projects.

Q. **Quadros today is primarily an RTOS company, not a tools company. What recommendations can you give a developer on how best to choose both his RTOS and his tools partners?**

A. We have always been tool-agnostic. Our belief is the developer should be free to use the development environment (compiler, debugger, linker/loader) that best fits his needs. As a result we have support for a wide variety of embedded tools: IAR Embedded Workbench, Freescale CodeWarrior, ARM RealView, Keil MDK, WindRiver Diab Compiler, Analog Devices VisualDSP++ and gnu.

Q. **Beyond the RTOS, Quadros offers communication stacks, file systems, development kits, and even GUI development tools. How do you recommend a developer analyze these sorts of software products vis-à-vis his RTOS vendor? How, and why are they important?**

A. Generally, most developers prefer to source these products from the RTOS supplier. This avoids the integration issues we discussed earlier. We have a best-in-class strategy to ensure that our customers are assured of a quality solution. This approach combines software products developed in house and third party products. For any module that we integrate, we pay close attention to issues such as footprint, data requirements, performance, source code availability and license model so that we can offer a consistent suite of product solutions to our customers.

Often we find there is one software area that the developer perceives as the highest risk to his project (currently this is USB communications.) That means we spend more time with the development team in advance of the product sale, specifying the various components and discussing implementation details.

ROWEBOTS: RTOS FOR DSP

31 October 2007: RTOS for DSP

INTERVIEWEE. KIM ROWE, CEO
TEL. +1 519 208 0189
EMAIL. QUESTIONS@ROWEBOTS.COM
COMPANY. ROWEBOTS RESEARCH INC.
WEB. http://www.rowebots.com/

Q. **First of all, tell us a little bit about yourself and your responsibilities at Rowebots.**

A. I've been a DSP engineer for 25 years and I've built all types of systems. I've also run DSP companies during that time and I have a deep appreciation for the business side of things as well. In a team environment we've helped design hundreds of applications from motor control and small speech applications, through large autonomous "robots". We have many market firsts over the years generally involving heterogeneous multiprocessors and tiny POSIX kernels. Today at RoweBots we focus on software and consulting for single chip DSPs and DSCs, and specialized multi core DSP and embedded operating systems.

Q. **Can you briefly outline your product lines?**

A. We have three general offerings: software **products** for DSP and embedded systems, design **services** and **training**. We find the combination of these elements best to serve our customers.

DSPnano RTOS, iZoom! IDE, and the *Unison RTOS* are our software products. All are open source.

The ***iZoom! C/C++ Integrated Development Environment*** *(IDE)* is very powerful and yet very low cost. It is Eclipse and GNU C/C++ based and runs on Linux and Windows. The idea behind this product was to offer great open source technology with specific add ons, testing and support at a price point where every engineering organization could put a copy on every desk. It is a general engineering tool for model development, data analysis, and application development that equals or exceeds the industry's best environments.

The ***DSPnano RTOS*** is unique because a DSP RTOS requires special features to effectively implement a DSP application. There are tens of thousands of DSP applications today from speech processing through intelligent vision systems, which run on DSPs and DSCs or collections of these. The *DSPnano RTOS* provides compete and seamless integration for the tools, real-time executive and I/O subsystems enabling developers to deliver these DSP applications quickly and easily based on fully functional POSIX standards. This approach allows our customers to achieve maximum effectiveness and minimum cost and time to market.

The ***Unison Embedded and DSP RTOS*** is in it's forth version. It offers support for uniprocessors; heterogeneous multicore solutions, which require shared memory and/or thin wire architectures and a small footprint. Transparent heterogeneous multiprocessing (THMP) with POSIX, a complete Eclipse based IDE, and complete I/O and DSP libraries are included. It is ideally suited to embedded applications, which require THMP, limited resources and POSIX libraries. It is fully optimized for DSP applications.

Our **training, design and consulting** services are DSP and embedded system focused as well. We have a team of very skilled and experienced people doing this work. The

training covers DSP as well as DSP system design using DSP RTOS solutions, FPGAs, various DSP chips and more. Other training courses cover embedded systems, RTOS fundamentals, various I/O systems and more.

Q. **Much of your efforts seem focused on RTOS for DSP. Why do you see this as a market opportunity? What sorts of applications or designs would profit the most from employing your technology?**

A. DSP RTOS is unique and it requires special features to be effectively implemented. There are tens of thousands of applications today from speech processing through intelligent vision systems, radar and sonar. If the DSP RTOS solutions is not integrated for maximum effectiveness and minimum cost and time to market for clients, then significant risk and expense is created for the project. That's what RoweBots does; create environments and reusable modules for effective DSP and embedded system delivery from both technical and business point of view.

Q. **What features does a DSP RTOS offer that makes it different?**

A. A DSP RTOS or even an RTOS is not always necessary on tiny and simple applications. Some new multicore architectures eliminate the RTOS completely by statically scheduling each function to one processor but these are the exception not the rule.

In all DSPs I/O is required for the primary data to be processed. In addition, some type of output is generally required for the processed data. Real systems need diagnostics and monitoring. Other parameters are often required from the environment: temperature, pressure, distance measurements, direction, velocity, and acceleration are often inputs into signal processing systems. With all this I/O and other varied requirements, multiple software modules are required to implement them effectively and in a maintainable fashion. A DSP RTOS is the best solution to the problem.

A DSP RTOS offers the following features and benefits:

- Fast and easy startup
  - Reduce time to market
  - Maximize profits and market share
- Single process with multiple POSIX threads:
  - Supports diverse parallel and pipeline implementations
  - Provides simple access to shared data.
  - Builds in modularity for lean product development
  - Large available programmer pool
  - Simplifies initialization
- POSIX synchronization and communication mechanisms
  - Well known and broadly used models
  - Large available programmer pool
- POSIX timers
  - Time based response with well known models
  - Large available programmer pool
- POSIX memory management with multiple pools and fixed size shared blocks
  - Deal with flash, RAM, EPROM and other memory types easily.
  - Build single processing pipelines with ease
  - Manage resources more effectively
- Open Source Model
  - Modify anything to suit your application or your processor
  - Benefit from community support
  - Easily port to a new FPGA processor with a custom instruction set
- Integrated POSIX Thread Safe I/O
  - Build systems quickly and easily
  - Support lean product development.
  - Tap large programmer pool

- Integrated DMA
  - Increase performance with minimal effort
- Integrated Thread Safe DSP Libraries
  - Faster development
  - Better performance with minimal context switching
- High DSP and DSC Performance
  - Elimination of expensive optimization
  - Deliver more features for the same price
  - Faster development

Today, the *DSPnano RTOS* is the premier solution on the market for DSPs and DSCs. Visit www.rowebots.com.

Q. **Many applications that use DSPs deploy in a heterogeneous multi-core environment. How do you address "multi-core?"**

A. Multicore is all about energy consumption and power dissipation. By using multiple cores, the total power consumed by a high performance chip is reduced substantially because the power consumption is proportional to the square of the clock frequency of the chip. Larger chips were simply not able to dissipate enough energy, therefore we get multicore to reduce power consumption and cool off the chip.

The cost of multicore is on the software side. Now we have to deal with multiple processors communicating to solve a problem. To date, two basic architectures have emerged; the traditional shared memory model with L1, L2 and L3 cashes, and bus connected multiprocessors using some type of grid or mesh or pipeline of cores.

On the issue of heterogeneous cores, all industry experts are pretty much agreed that heterogeneous is the only way to go in the medium and long term. The rational is simple. Processors can be specialized for specific tasks, TCP for example, and then the overall performance per square mm is better. It means extra costs and product trade offs to design special parts but the benefits are there if the segment is large enough.

Traditionally, low volume high margin DSP applications have deployed as heterogeneous multiprocessors. Then they added FPGA accelerators. Lower volume applications are moving towards high performance COTS parts (i.e. Quad core) with FPGA add ons today.

In the mid market, FPGA solutions with add on DSPs or COTS multicore processors are the most cost effective in general, but many are using multicore DSP solutions here as well. If the application is very parallel it favors FPGAs and if not, DSPs are the choice.

For the volume end of the market, now they are moving towards heterogeneous multicore solutions. Some of these are even used in the mid market. Some examples here are wimax base stations and HD TV encoders.

Now, to answer your question, we are modifying the ***Unison Embedded* and *DSP RTOS*** to deliver effective DSP and embedded solutions in these markets for low, medium and high volume. Unison offers transparent heterogeneous multicore support across thin wires and shared memory with complete I/O. DSP libraries are included. It is smaller and leaner than other solutions and as such offers resource savings for embedded applications. *Unison* also offers POSIX and a complete Eclipse based IDE.

Q. **Which DSPs do you support?**

A. The *DSPnano RTOS* offers a complete solution for single chip DSPs, DSCs and microcontrollers. It includes IDE, RTOS (POSIX), complete I/O solutions, DSP libraries, training and support. Its main claim to fame is its tiny size with POSIX compliance, DSP features and complete I/O.

DSPnano supports higher end Microchip processors like *dsPIC* and *PIC24, Pic32,* FPGAs like *NIOS* from Altera, Renesas microcontrollers *M16C/32C,* TI processors and ADI processors and Freescale CF. *PowerPC* is also in the mix.

Q. **What is the difference between your *DSPnano* and *Unison* products?**

A. *DSPnano* is intended for very tiny footprint systems where cost is a big issue. This is generally the territory for DSPs, DSCs and microcontrollers from 8 to 32 bits.

The *Unison DSP* and embedded RTOS are intended for bigger systems where transparent communication across thin wires and shared memory is necessary. It is intended for very embedded DSP systems using heterogeneous and/or multicore technologies.

With this said, the feature sets are very close in that they both use Eclipse development and POSIX standards throughout. Both integrate DSP libraries and thread safe I/O libraries. They share common startup routines and I/O modules. Porting from one to the other or even to Linux or Solaris is easy.

Q. **Does that mean that *DSPnano* and *Unison* share all their I/O modules and drivers?**

A. The easy answer is yes; however it is far more complex than that. RoweBots delivers DSP and embedded operating system environments to maximize effectiveness for our customers. We deliver these solutions on a variety of hardware, most of which is completely different from *DSPnano* to *Unison*.

DSPnano is targeted at very lean resource poor environments while Unison is less demanding in that way. For this reason, different I/O solutions are used. Because the hardware is different, generally based on system on a chip (SoC) solutions, then the drivers are completely different as well.

From the user point of view, the applications look very similar: almost identical.

As multicore versions of the current uniprocessors emerge, then Unison and *DSPnano* will in fact share the same drivers and potentially the same I/O. Unison will run on the multicore versions and *DSPnano* will run on the *unicore* versions.

Q. **What is the licensing arrangement on your products?**

A. Our licensing has several components. The first component is a per user cost for tools development which ranges from $29/user to $1299/user depending on the environment.

The second development component is an embedded development license per project, which ranges from $0 to $1999 depending on the user group and affiliation. For example, students don't pay for development licenses. Special non-commercial pricing is also available.

All deployment costs are separate and these licenses are royalty free, ranging from $499 through to $29 999 depending upon your needs. This allows developers to remove the software from their Bill of Materials (BOM) and consider software a sunk cost, improving margins.

Finally, to simplify purchasing, bundled products are offered which include: development seats, development licenses and deployment licenses. This is generally the most cost effective way to purchase and it typically reduces costs by 10% to 20%.

Q. **And how much does it all cost? What are your various models for purchase?**

A. Our *iZoom!* product provides Engineering companies a very high quality C/C++ environment complete with textbooks, examples and support on their desk for $29 for a single user. Volume costs are lower. Costs for a department, complete with in house training are under $5000. Without training those costs drop to a few hundred dollars.

DSPnano for a typical application would cost $6000 for a royalty free license and development. If training was included the cost would increase to $10 000.

Unison DSP RTOS is packaged per application without a fixed price. It generally includes porting to dedicated hardware, tool integration and training. Pricing starts at $19 999 for development, integration and a royalty free license.

Our typical consulting rates are $400-$800 per day for DSP specialists.

Typical course costs range from $999 to $2400 per student per week for individuals and $6999 and up for in house courses depending on the volume.

Of course, volume discounts apply to all products.

Q. **Why should an engineer buy from RoweBots and not one of your competitors?**

A. First, I'd say that if our product were not the very best solution for your problem and one of our competitor's products is, you should buy their product. This type of evaluation is typically not so clear though particularly when short, medium and long-term benefits are considered.

Second, I'd re-iterate that our lines of products and services are focused on specific applications and requirements. If you are in our target market, ours will be the best product for your needs or it will not be on the market. You can go across each product or service that we offer and see this principle at work.

Third, I'd point out that we started in this business in 1987 and we've just celebrated our 20th year in business. We are here for the long term and our products and services reflect that long-term commitment.

Forth, we're an open source software company. Other companies may or may not subscribe to this model but we do. Open source reduces costs and risks for our customers. It is now proven in the market place and we can be sure that we can deliver the best products in our markets using this approach.

Fifth, we have great values. We're committed to our customers success and building a better world. We'll do everything we can to help our customers succeed. Profit is always secondary to these other goals.

Finally, I would add that we're just fun to deal with. We're international and have international attitudes, approaches and values. Lets have some fun and build a better world together.

Q. **Thank you for this interview.**

Interviews: The RTOS Tools / Connection

Many of the real experts on RTOSes are the vendors themselves. We interviewed many leading vendors and tried to tie each to an important "theme" to help those selecting an RTOS choice. Here are interviews relating to the RTOS / Tools Connection.

- DDC-I: EMBEDDED DEVELOPMENT TOOLS AND THE RTOS CONNECTION
- IAR SYSTEMS: THE RTOS TOOLS CONNECTION
- SEGGER: THE RTOS / TOOLS CONNECTION
- TELELOGIC: PROJECT MANAGEMENT FOR EMBEDDED AND REAL-TIME

DDC-I: EMBEDDED DEVELOPMENT TOOLS AND THE RTOS CONNECTION

8 November 2007: Embedded Development Tools and the RTOS Connection

INTERVIEWEE. OLE OEST, CTO
TEL. 602 386 4360
EMAIL. ooest@ddci.com
COMPANY. DDC-I
WEB. http://www.ddci.com/

Q. **First of all, tell us a little bit about yourself and your responsibilities at DDC-I.**

A. I'm one of the founders of DDC-I and have worked for the company since it was formed in 1985. I have a Ph. D. in Computer Science in the area of formal methods for software development. DDC-I's *Ada compiler system*, developed in the 1980s, was the fist large-scale development using formal methods. The system was very successful, and the methods used to develop it laid the foundation for DDC-I's future new information assurance/software assurance product line.

Q. **DDC-I is not an RTOS company. This second edition of our Guide has a focus on "beyond the RTOS." How does DDC-I take a developer "beyond" the RTOS?**

A. DDC-I's software development tools work with an RTOS as well as without. When the application developer is looking for the smallest footprint and fastest execution speed, he turns to one of DDC-I's bare board solutions, which allows the user to use the bare board kernel we provide and only use the services needed for the application. However, if the developer needs a full-featured RTOS, our tools will also target the RTOS.

The DDC-I solution also allows developers to use the Java programming language without an RTOS. This is unique in that an RTOS is typically required for the Java Virtual Machine to function. Our run-time system allows our *Scorpion JVM* to run on a bare board without an RTOS, thereby saving cost and memory space.

Q. **Your company has a focus on "safety critical" software, as do certain RTOS vendors like Green Hills or LynuxWorks. How do you define the requirements of "safety critical?" How does your product complement the work of RTOS companies in the safety-critical realm?**

A. DDC-I's compiler systems are optimized for developing safety critical applications. Safety critical applications must exhibit traceability from the executable machine code to the source text. DDC-I's compiler systems generally do not insert any extra code into the executable, which cannot be traced back to the source text of the application.

Safety Critical applications must also be predictable. With DDC-I's tools, even applications written in real-time Java (RTSJ) are predictable due to DDC-I's unique and patented garbage collection technology. Full traceability from executable to source requires *Safety Critical Java*, which DDC-I will introduce shortly.

Since we support Java and Ada (RTOS vendors do not), we can offer this functionality for the RTOS while at the same time maintain the traceability and determinism needed for safety critical applications.

Q. **Tell us please about your efforts in Java. How does DDC-I help to make Java ready for real-time and embedded systems?**

A. Our *Scorpion Java* product is a true hard-real-time solution for Java programmers.

One of the biggest issues with Java is the need for garbage collection. In the traditional RTSJ implementation, the programmer must specify which threads will be real time and which will not. Then, they must program each thread accordingly. Even if they do it right, the system can become non-deterministic if the garbage collection process gets interrupted frequently enough to cause a memory leak issue or lack of resources. Also, establishing communication between real-time and non-real-time threads without destroying the real-time behavior is tricky and error prone.

Our *Scorpion* solution solves these issues by allowing all threads to be real-time threads, thereby enabling them to communicate with each other freely while guaranteeing real-time performance. This is achieved through a patented deterministic garbage collection process in which the time required for collection is allocated to each and every thread allocating memory, thereby making the time needed for garbage collection bounded and predictable.

DDC-I's tool suite also supports mixed language development, which allows developers to compile and debug a mixture of Ada, C, C++, Java and FORTRAN in a single application. These languages can make calls to each other, thereby allowing for tighter integration and more code re-use. The debugger can also debug all these languages in a single session (from source text to machine code with co-relationships depicted), thereby making code reuse for embedded developers a real possibility.

Q. **What is *OpenArbor*? How does it relate to *Eclipse*? And what is its value to a developer who may (or may not) be using a commercial RTOS?**

A. *OpenArbor* is DDC-I's Eclipse-based framework, which combines our mixed-language debugger with compiler plug-ins for Java, Ada, ANSI C/C++, and FORTRAN. Our *Open Arbor* environment and compiler plug-ins can target our own bare board solutions as well RTOSes from companies such as Wind River and LynuxWorks.

Q. **Tell us about *SCORE*. Is this an RTOS product? Does it create RTOSes, work with them, what?**

A. *SCORE-Ada* is an Ada run-time system and compiler. It can work with and run atop an RTOS, or be used as a bare board solution using a run-time system for a specific processor.

SCORE compiler systems with similar characteristics are also available for C, Embedded C++, and FORTRAN.

Q. **Finally, tell us a little about how DDC-I engages with customers. What models do you have for customer engagement and product purchase?**

A. DDC-I uses direct sales to engage customers. We have field sales people as well as field application engineers to help customers choose the best solution for their application. We sell either a perpetual license for the development seat or a yearly subscription for the development seat. For customers that purchase the perpetual license, we offer a yearly support contract that provides real-time user support as well as maintenance releases.

With the exception of the Java products, there are no run-time production royalties. The customer gets an unlimited right to use the run-time for the cost of the development seat. We do charge run-time royalties for Java.

Q. **Thank you for this interview.**

IAR SYSTEMS: THE RTOS TOOLS CONNECTION

19 November 2007: The RTOS Tools Connection

INTERVIEWEE. LOTTA FRIMANSON
MANAGER – RTOS & MIDDLEWARE
TEL. +46 18 167800
EMAIL. lotta.frimanson@iar.com
COMPANY. IAR SYSTEMS
WEB. http://www.iar.com/

Q. **First of all, tell us a little bit about yourself and your responsibilities at IAR.**

A. I am the product manager of *IAR PowerPac*. I have a Master of Science in Engineering Physics from Uppsala University, and have worked at IAR Systems as a product manager since 1999. Prior to this I have 10 years of experience from embedded systems programming working for Biacore as a software developer and at Styrex as a consultant.

Q. **IAR Systems is known as an "embedded tools" company, yet you recently launched your *IAR PowerPac RTOS*. What is the strategy here?**

A. *IAR PowerPac* is a complement to our C++ compiler - *IAR Embedded Workbench*. *IAR PowerPac* is an add-on RTOS and middleware family that is very well pre-integrated with other IAR Systems products with the ambition to provide a more complete solution throughout the whole development chain

Q. **Since our guide is primarily about RTOSes, can you give us a brief sketch of the features and benefits of IAR Power Pac? What is unique about it as an RTOS?**

A. IAR Systems offers one-stop-shopping for embedded development tools—RTOS and middleware components integrate seamlessly with *IAR Embedded Workbench* and provide a complete development environment even for the most complex applications.

IAR PowerPac is a fully featured real-time operating system (RTOS) combined with a high performance file system. *IAR PowerPac* is tightly integrated with *IAR Embedded Workbench* and comes with sample projects and board support packages from different manufacturers.

IAR PowerPac USB device stack and *IAR PowerPac TCP/IP* protocol stack are additional options.

Unique things with *IAR PowerPac a*re:

1. Low-risk business model with its site-license structure.
2. Pre-integrated solution for *IAR Embedded Workbench* customers
3. Easy to get started
4. Small footprint and RTOS debug awareness

Q. **There are many RTOS companies, but only a few RTOS companies that are tools companies as well (Green Hills Software comes to mind, as does Wind River Systems). What do you see as the advantages in unifying the choice of RTOS with the tools provider?**

A. 1. One-stop-shopping for our customers

2. Support of the complete solution

3. Pre-integrated solution throughout the complete development chain

Q. **What is your business model for selling *IAR PowerPAC RTOS*? Is it different from the established RTOS vendors?**

A. *IAR PowerPac* is licensed in a similar way to that of *IAR Embedded Workbench*. In addition to a single-user license, with an option to include source code, IAR Systems is also introducing the concept of a group license suitable for development teams of up to 20 developers, and a site license for an unlimited number of developers per site. In addition to the license fee there are no royalties or fees connected to projects, end-user products/product families or production volume. This makes investing in *IAR PowerPac* a truly low-risk licensing model, since all associated licensing costs are known in advance.

Q. **Do you think developers should look at tools first, and then choose their RTOS? Do you recommend a different way of thinking about embedded development vs. the path from hardware to RTOS and then to tools? Why?**

A. When the applications get more and more complex and the need for external communication increase, customers can save a lot of the time and money by finding a solution where tools, RTOS and middleware already are tested and verified together.

This means that developers should look at all the components that are needed first and then try to find a complete offering, rather than buying bits and pieces. If you can buy the product from the same vendor this will of course not only save you some time, you will also get support from one place.

Q. **You recently announced a TCP/IP stack for *ARM* for *PowerPAC RTOS*. Tell us about not just about this but your view about "middleware" and how it relates to the RTOS choice.**

A. Many customers identify the need for an RTOS, when they start to work with solutions that require external communication. The most common communication stacks today are USB and TCP/IP. The differentiator between different RTOS's is in many cases its integration with middleware and building tools.

With the TCP/IP stack module, *IAR PowerPac* includes everything necessary to add a fully working TCP/IP interface to an *ARM* device. *IAR PowerPac TCP/IP* stack is runtime configurable and designed to work on any *ARM*-powered embedded system with an Ethernet controller.

The communication stack is supplied with ready-to-run projects for supported ARM-powered microcontrollers. Supported protocols include: TCP; IP; UDP; Telnet; ARP; DHCP (client); ICMP; and TFTP. The stack is highly optimized to ensure both minimum memory consumption and high-speed operation, and is connected via a standard BSD socket interface. It features zero data copy for fast performance, and the number of connections is limited only by the available memory. Zero compile time configuration means that the code can be compiled into a library for use in future projects.

The TCP/IP stack also benefits from the unique per seat business model that is a key feature of *IAR PowerPac*.

Q. **Thank you for this interview**

SEGGER: THE RTOS / TOOLS CONNECTION

25 October 2007: The RTOS Tools Connection

INTERVIEWEE. ROBERT TEUFEL
CHIEF OPERATING OFFICER
TEL. +49 2103 28780
EMAIL. info@segger.com
COMPANY. SEGGER MICROCONTROLLER SYSTEME GMBH
WEB. http://www.segger.com/

Q. **First of all, tell us a little bit about yourself and your position at SEGGER.**

A. With almost 20 years of experience in Embedded Computing, I joined Segger as COO recently with responsibilities in partner relations and business creation among other things. My background includes embedded programming, competition analysis for 32-bit architectures and tool partner management. Before joining Segger, I managed very successfully all tool partners for NXP Semiconductors.

Q. **SEGGER sells an RTOS, but the company also has a robust focus on development tools. Can you run us briefly through your products?**

A. It is true that our company sells wide variety of middleware for the deeply embedded market space. There are 5 major software packages and a couple hardware products. All software works with our Embedded OS, embOS but can also be used with any other OS or if needed as stand-alone entity. Probably the best-known product is our Graphic Software and GUI, emWin. It enables fast graphics with minimal requirements to memory. In combination with embOS, customers can create eye-catching LCD graphic demos with a small fraction of memory needed compared to embedded Linux or any sort of MS Windows OS. embOS and emWin support exists for a large number of embedded architectures.

The emFile File System can be used with all kinds of memory cards, integrated Flash memory on microcontrollers, it supports NAND Flash as well as NOR Flash and the customer list includes big names in the memory card area. For USB applications we offer emUSB, a small and highly efficient USB device stack. It supports HID, Bulk transfer, Mass Storage device and CDC. Segger's portfolio has recently been expanded to include embOS-IP, a TCP/IP stack which has been ported to several microcontrollers with embedded EMAC already and will be expanded to cover most standard Ethernet interfaces that are used in the market within the next quarter. Common to all our software solutions is the "no royalties" concept. Customers operating in a price sensitive market can not afford to pay a percentage of every product to the manufacturer of an OS or any other software used in their product.

Q. **Like many other RTOS vendors, you sell related software such as TCP/IP stacks, USB stacks, file systems, etc. What is different or unique about your software offerings or approach?**

A. Our focus has always been to offer compact and fast code, specific for Embedded Microcontrollers. Software from Segger is actually software developed by the Segger-team. Our team is able to deliver best in class support because the experts are not employees of another company, they are part of the Segger team.

Q. **There is this whole concept of "middleware" nowadays in embedded systems. How do you define "middleware?" How does it make the job of embedded designers easier?**

A. Middleware can be described as software that acts as an intermediary between software applications so that they can exchange data. An OS can use a USB stack to talk to a PC. The wide variety of software products sold by Segger covers many areas needed in today's embedded systems.

Q. **Unlike other RTOS vendors, you also sell many hardware development tools such as your ARM J-tag Emulator, flash programmers, or starter boards. Why is the company focused on hardware tools as well?**

A. That's a funny story, glad you ask. It all started out as a promotion for our software. The focus is still on software tools but our team generated such a high performance J-TAG emulator called J-Link, that it became best in class with download speeds up to 720 Kbytes/sec. This superior download speed was made possible by using our own USB device stack. The intent of the promotion was to get in contact to more customers with hardware tools that cost a lot less than software packages. After a while, J-Link and its companion J-Trace grew into a decent business all by itself. With specific solutions for many semiconductor vendors, J-Link is sold by a number of companies as OEM product. J-Trace is still relatively new and offers trace if the microcontroller itself supports this feature.

Segger offers also Flash Programmers, called Flasher for different architectures. The Flasher ARM is the latest product of the Flasher family. These devices provide a robust low cost programming solution that can be used on the test floor where PCs are not so common. Flashers can also be remotely controlled and used for Production programming in larger quantities.

Q. **There are quite a few RTOS companies ranging from the very small "niche" players to Microsoft or Green Hills or embedded Linux vendors like Wind River. What do you think differentiates SEGGER from the pack?**

A. Segger offers a very small, yet highly efficient OS. With highly optimizing compilers the kernel can be as small as 1.5 KB for a 32-bit architecture. Task overhead is also minimized as well as SRAM usage. Microsoft and *eLinux* as examples offer complete packages that require any Mbytes of memory, thus can usually not be used in combination with single chip solutions. Using *embOS*, *embOS-IP*, *emWin*, *emFile* and *emUSB* all together in an application still provides the option to run the system on a single chip microcontroller. Low cost systems with compact and highly optimized code, which is what makes Segger offerings different!

Q. **Unlike many vendors, you have an online store (http://www.segger-us.com/). Selling embedded software, however, usually seems to be a "hands on" process. What is your reasoning behind the online store? What does this bring to developers?**

A. An online store brings transparency in pricing. We do not hide our prices, they can be downloaded, compared and there are no sudden surprises. Still selling embedded software is a hands-on process. If a customer wants to order hardware, he can do so with a credit card or other online payment options but if he wants to purchase a software package from Segger, he needs to contact a Segger office directly, either via e-mail or via phone. There is no option to buy the OS by providing a credit card number. Through this direct interface, we avoid misunderstandings, which software requires what components and how do they interact is discussed in detail with the customer. Having excellent support through our US office together with the online store has been extremely successful so far.

In a nutshell, the developer gets the best of both worlds, easy access to pricing and downloads of trial software and at the same time the kind of technical support that shortens his time to market.

Q. **Thank you for this interview.**

TELELOGIC: PROJECT MANAGEMENT FOR EMBEDDED AND REAL-TIME

24 October 2007: Project Management for Embedded and Real-time

INTERVIEWEE. MICHAEL LESTER
DIRECTOR OF PRODUCT MARKETING
TEL. 949-830-8022
EMAIL. info@telelogic.com
COMPANY. TELELOGIC
WEB. http://www.telelogic.com/

Q. **First of all, tell us a little bit about yourself and your responsibilities at Telelogic.**

A. I am responsible for helping organizations solve problems they are facing which can be addressed with Telelogic's product portfolio and requirements management product lines. In that role, it's my responsibility to understand the needs of the market and our customers in order to have successful products, as well as share the examples of successful customers with those that are maturing or introducing new methods for doing product planning. Prior to joining Telelogic I have been in product management roles at Pacific Edge Software and Artemis International, and technical or management positions at Buildnet and Franklin Covey.

Q. **Telelogic is obviously not an RTOS company. Tell us quickly why and how Telelogic is important to companies in the electronics industry, especially those managing large multi-person "teams."**

A. Telelogic has focused on solutions for the entire lifecycle of product development. Our portfolio covers every stakeholder – from the executive, customer or business unit manager all the way through product planner, designer, engineer or tester. By bringing all of these different roles together, they all benefit. It's important for the person managing a large team or project to have visibility the 'why' behind the project's scope. A lot has been written about the importance of project schedules, resource utilization and finances of a project, but only managing those aspects of product development often leads to well executed projects that don't necessarily deliver what was requested. Telelogic feels that it's important to select the right scope to maximize the value delivered to your customer and to your organization, AND to deliver it efficiently. The end result is a well-built product that is valued by your customer.

Q. **Which Telelogic products are most applicable to the embedded systems and/or electronic design world? Can you briefly summarize each?**

A. Well, the upfront part of the process is often the most neglected. Organizations may spend time on detailed design, but very often the part of the process prior to that, where we ask 'Why?" if skipped or not connected. *Telelogic Focal Point* facilitates the gathering of that information and the decision-making that helps organizations and product managers make informed decisions about scope and requirements with an eye towards the value they deliver to the customer and just as importantly, the value delivered back to the company.

In addition to delivering value through *Focal Point*, embedded teams that are constantly driven to improve quality and do more with less, while still demonstrating compliance to standards and regulations will find *Telelogic Rhapsody* to be valuable. *Telelogic Rhapsody* offers a model-driven development environment through which you can create graphical models of the system that can be simulated and tested, while generating the code that is directly deployable to the end product.

Q. **This year we are trying to go "beyond the RTOS." Why do you think project management, and thereby software to manage complex projects, is important to the industry? Why can't people just keep going with the ad hoc systems that are so prevalent?**

A. Project management is the consistent method by which all of the various development practices – requirements, design, development, test, change management, documentation, etc. coordinate their activities. The trick is bringing them all together to achieve a common goal... that's the responsibility of good project management. When project teams grow or project complexity is increased either as a result of scope, limited budgets or resources, or constrained timelines, the pressure is increased on the team to communicate more efficiently. In order for that to happen, it is critical that everyone is working from the same information and they understand the 'Why' behind that information. The "Why" piece is critical because without it, everyone will have a different interpretation of the information. We think of this as value-driven project management.

That is why a solution, like *Focal Point*, that brings all of that information together with the "Why" included, and is accessible easily through a web browser for all of the various stakeholders has been so well received. Everyone from executives wondering about status of projects all the way through testers trying to validate a product's compliance to requirements can access *Focal Point*.

Q. **Recently, the company announced *Focal Point 6.1* with Microsoft *Project* integration. Please tell us about this announcement.**

A. Yes, *Focal Point 6.1* introduced integration with *Microsoft Project*, allowing a product manager to easily pass the requirements, high-level timelines, and resource or skill needs into a *Microsoft Project* plan. This kicks off the project quickly, in a format that project managers and teams already consume. With the integration, as the project team updates tasks and activities with duration, scope and status, the product manager gets updates in *Focal Point* of the overall project's duration, scope and status.

Our goal was to make life easier for the product and project managers by automating the reporting and visibility activities, which allows them both to focus on their core responsibilities.

Q. **So what you are saying is that embedded software development - especially that in large complex projects and "teams" - can really benefit by using software that organizes the *project*?**

A. Definitely. That complexity is difficult to manage and it becomes so overwhelming that product and project managers are spending more time doing the mundane, tactical activities of communicating status, managing activity updates and providing reports to stakeholders that they can't actually lead their products or projects. Research shows that most products are failing because of internal problems where teams don't have the right understanding of the customer's needs and the value to the customer, because those in the decision-making roles do not have the time to understand how things should be organized.

Q. **Having looked at the *Focal Point* Demo online, it seems like this is a very business oriented product in which managers are able to trade off costs and benefits of product features, while various other stakeholders (e.g., customers, engineers, marketers) are able to cooperatively input their opinions. How does *Focal Point* help engineers who deal with the "real world" of what is technically possible interact with managers or markets that deal with the "customer world" of what requirements might be wanted?**

A. Yes, there is definitely a business side to the use of *Focal Point* because that is where the vision and understanding of the business or market problem originates. But, as you point out, there are requirements that come from the technical world as well that often times are viewed as constraints by the business side. *Focal Point* brings the important

information from the 'customer world' and balances it against the technical achievability, but also allows a product manager to introduce technical requirements alongside those from the business so that the business side learns about the platform requirements that exist and need to be developed to support their needs. All of that is done in one place so that the value of these can be balanced, evaluated, assessed and decisions made from a full picture of the market requirements and technical possibilities.

Q. **If you were an engineer or manager at a company considering an RTOS and all sorts of other interrelated technologies for their product design, how would you go about convincing others to formalize project management? Can you give us some bullet points to take to higher management?**

A. Well, I'm sure they have some aspect of project management, but I think the form it takes varies. That is where the critical issues need to be raised from engineers. They need to ask the questions about where the requirements come from and how are the decisions made that decide the scope of the product releases and projects. We find that all of the critical activities are usually happening in an organization, they just don't work together and share the information they use. As a result, the same decision is revisited and something is lost each time the decision gets farther from the customer. I think if teams explore the number of decisions that are made between the customer's input and the actual development they'll find they can save a considerable amount of time and money by centralizing all of that information so nothing is lost and decisions do not need to be revisited.

Q. **Thank you for your interview.**

Interviews: Open Source Linux & Beyond

Many of the real experts on RTOSes are the vendors themselves. We interviewed many leading vendors and tried to tie each to an important "theme" to help those making an RTOS choice. Here are interviews relating to Linux and other open source RTOSes.

- eCosCentric: A Commercial Open Source RTOS, eCosPro
- LynuxWorks: *Linux* for Embedded and Real-time Systems
- Micro Digital Inc.: RTOS Licensing Models
- MontaVista: Linux for Embedded and Real-time Systems
- QNX: New Licensing Models for Real-time Systems
- TimeSys: LinuxLink as a Model for Linux Development
- Wind River Systems: Linux for Embedded and Real-time Systems

ECOSCENTRIC: A COMMERCIAL OPEN SOURCE RTOS, ECOSPRO

30 October 2007: A Commercial Open Source RTOS, eCosPro

INTERVIEWEE. PAUL BESKEEN
CHAIRMAN AND DIRECTOR OF BUSINESS DEVELOPMENT
TEL. +44 1223 245 571
EMAIL. info@ecoscentric.com
COMPANY. ECOSCENTRIC
WEB. http://www.ecoscentric.com/

Q. **First of all, tell us a little bit about yourself and your responsibilities at eCosCentric.**

A. My career spans over 20 years in the engineering and technical management of embedded operating systems and related software. I started at Perihelion Software, a company specializing in parallel and embedded operating systems. The ten-year stint at Perihelion culminated with my appointment as Technical Director (CTO) of the company. I subsequently worked for Virtuality Inc, a developer of virtual reality hardware and software, as manager of its Operating Systems group. In 1996 I joined Cygnus Solutions as Director of Engineering, tasked with building and heading up a team to create an open source operating system designed specifically for the deeply embedded space – the *eCos RTOS*. I continued in this role after the merger of Cygnus Solutions with Red Hat. I am one of the founders and chairman of eCosCentric. As part of its management team I have responsibilities in marketing and business development.

Q. ***Embedded Linux* obviously gets all the buzz as an "Open" RTOS. Give us your elevator pitch about why we should care about *eCos* instead of *Linux*.**

A. In many ways *eCos* and *Linux* are complementary as they provide solutions to different segments of the embedded space. *eCos* was specifically designed as an open source RTOS for the deeply embedded market. *Linux* in contrast was developed as a general purpose open source operating system that has successfully been used in higher end embedded devices. *Linux* memory requirements are typically measured in megabytes. In comparison *eCos* is an extremely scalable embedded operating system that, depending on the required functionality, scales from a few kilobytes to a few hundred. The memory budget for an *eCos* based system is therefore very much lower than any *Linux* system, enabling lower unit costs to be achieved. Indeed, its footprint is capable of fitting within the confines of SoC designs that have only limited amounts of on-chip memory available. As product retail prices are typically three times that of the BOM cost, this can be critical for price sensitive, high volume designs.

Much of *eCos*'s scalability is due to its configurable nature. The configuration technology built-in to *eCos* enables developers to specify the required functionality and characteristics of the operating system. This effectively creates an application specific version that is ideally suited to the requirements of a particular device. *eCos* is therefore suitable for a wide range of device types, from designs that might have traditionally employed a "roll your own" OS approach, through to the more sophisticated real-time applications that require features such as networking, file systems and so forth.

In comparison with *Linux*, *eCos* is a far less complex system, enabling individual developers to more easily comprehend its implementation, and to port, debug and customize it to meet their requirements. *eCos* implements a classic real-time multi-threaded architecture with priority based scheduling. This is a very efficient execution model that delivers deterministic response times, minimal interrupt latencies, and low overhead context switches. *Linux* by contrast employs a more sophisticated and

heavyweight approach, but does provide virtual memory, paging and multiple processes with memory protection. For deeply embedded real-time designs the former rather than the latter functionality is usually a more suitable choice. *Linux* however excels where it can leverage its more extensive feature set in devices such as cell phones and PDAs.

In conclusion, both systems benefit from their open source development approach. For developers they provide the ultimate in flexibility and freedom, whilst from a commercial perspective they are attractive due to the lack of royalty and license fees. Both are backed up by commercial distributions and support services. Where they differ is in the embedded market segments that they service. *eCos* is more suited to embedded devices that require instant startup, real-time performance, and have more challenging RAM/ROM budgets and processor performance targets. *Linux* is more suitable for higher-end devices that are not too restricted in machine resources, and that require more sophisticated functionality.

Q. **How does *eCosPro* compare to commercial RTOSes like *ThreadX, VxWorks, and Neutrino* on a technical level?**

A. Technically *eCosPro* provides a broadly similar set of features and functionality at the kernel level. It includes a multithreaded kernel with priority based scheduling, a rich set of synchronization primitives, priority inversion protection, timer and interrupt handling, and so forth. By design the kernel never needs to disable interrupts during normal operation, and therefore delivers best in class interrupt latency performance. *eCos* doesn't provide memory protection as the kernel and application are directly linked together, but this does enable smaller resource footprints to be achieved. The configuration technology provides a finer grained level of control and scalability than other RTOSs, with built-in validation and dependency checking in the configuration tool to ensure a self-consistent application specific operating system is generated.

In common with most commercial RTOSs, *eCosPro* is a portable system and has been ported to virtually all major 32bit processors, and many other less common ones. In addition *eCosPro* includes the *RedBoot* bootloader that is also layered on *eCosPro's* hardware abstraction layer. Thus both RTOS and bootloader / software debug agent functionality result from a single porting exercise.

In addition to the kernel, *eCosPro* includes a range of additional features including C and C++ runtime libraries, *Posix* and *uITRON APIs*, file systems, and networking and so forth – none of which are extra cost options. It supports development on *Windows* and *Linux* hosts, and includes a full set of compiler tools and *Eclipse*-based IDE.

Q. **What advantages does a developer get from the fact that *eCos* is "open source?"**

A. The general benefits of open source development are widely understood and fully applicable to *eCos*. Having the source code at hand ensures that developers enjoy maximum control, flexibility and understanding over all aspects of their particular embedded design. If required, the code can be inspected to better understand how it functions, it can be debugged and traced if you suspect a problem, it can be modified and extended if the functionality doesn't meet your exact needs. Another advantage arising from availability of source code is that the core software is well understood by many engineers and has been thoroughly vetted for efficiency, performance, and reliability - "Given enough eyeballs, all bugs are shallow."

Developers also benefit from the commercial freedoms that open source provides, these include perpetual access and rights to the codebase with no vendor lock-in, no restrictions on your IP, and no imposition of royalties, royalty audits, or license fees. *eCos* is licensed under a modified version of the well-known GPL license that allows the use of non-GPL licensed code to be linked with *eCos*. This means that your application, additions to the system, and any third party middleware, remain proprietary and do not have to be licensed under the GPL.

When developers wish to evaluate *eCos*, there is no permission to seek, or legal paperwork to process. The source, documentation and tools can be downloaded and evaluated to get a thorough understanding of the system before committing to any development.

The above advantages are backed up by a range of services provided by eCosCentric including: training, support, contract development and porting, and the *eCosPro* commercial distribution of *eCos*. This approach provides a "best of both worlds" combination of guaranteed availability of commercial support and services, whilst maintaining the benefits and freedoms of open source.

Q. **How does *eCosPro* compare to *FreeRTOS* (freertos.org)? Isn't that also an "open source" RTOS?**

A. FreeRTOS is a basic runtime kernel that only provides a limited set of features: binary semaphores and mutexes, queue handling, and task management. *eCosPro* can be configured down to similar level of functionality, if memory footprint is critical, but in contrast is also scalable to much greater levels of functionality.

If a project needs to be expanded to cover more features such as device support, file systems, and so forth, then *eCos* can readily accommodate this. Time invested in familiarizing yourself with *eCos* can also be applied to subsequent, more complex, embedded applications.

Q. ***eCos* can be downloaded off the web for free. Can you explain what the benefits to a developer are of going with *eCosPro*?**

A. *eCosPro* delivers enhanced levels of stability, features, and technical support compared to the free download. We believe that this provides the most efficient and lowest risk approach to the development of *eCos* based products. The use of *eCosPro* helps ensure that time to market and quality are not compromised by reliability problems, support issues, or lack of particular features and functionality. *eCosPro* also serves as the foundation for delivery of other eCosCentric services and middleware.

An *eCosPro* release integrates all the software functionality necessary to immediately start development of embedded applications. This includes an IDE, compiler toolset and utilities, configuration, profiling and other tools, as well as *RedBoot* bootloader firmware for the target hardware.

Each release of *eCosPro* is run through comprehensive set of QA and test procedures. Our automated test infrastructure typically runs over 21,000 tests on the target hardware. This helps ensure the stability and reliability of the runtime and tools that your product development will depend upon.

eCosPro extends the standard *eCos* functionality with a number of exclusive tools and runtime features. On the host side these include Eclipse-based IDE, profiling and code coverage, and memory allocation analysis and debug. On the runtime side these include a wealth of additional and extended functionality such as the C++/STL library, FAT file system with long filename, multibyte character set and removable media support, enhanced JFFS2 flash file system, and so forth.

eCosPro supports a number of platforms and processor variants that are not available in the public release. Device support varies by platform but typically includes a mix of serial, Ethernet, flash, RTC, watchdog, I2C and SPI drivers.

Perhaps most importantly *eCosPro* comes with comprehensive technical support. This provides a responsive, guaranteed source of advice, and resolution of any issues that a development team might encounter. eCosCentric was founded by the creators of *eCos* and no better source of *eCos* specific expertise exists.

Q. **How much does *eCosPro* cost? What is your business model?**

A. eCosCentric employs a hybrid open source business model. We sell a range of services and products based on the *eCos RTOS*. These include support, contract development and porting, training, and consultancy. These are generally provided under fixed price contracts, the cost of which varies depending on the level of engineering involved, required support period, and so forth. We retain the copyright on some of our extensions and additions to *eCosPro*, and restrict its redistribution to third parties.

It is also important to eCosCentric that the *eCos* project as a whole continues to prosper and grow. We are an integral part of the *eCos* open source community, supporting various hosting services, contributing code, and providing employment to the majority of the *eCos* maintainers.

Q. **Your company also sells middleware such as GUI's, JVM's, databases, etc. Is this middleware also "open source?" What middleware or peripheral software is available for *eCosPro* and how is it sold?**

A. The middleware comes from eCosCentric and a number of middleware partners. It is licensed under a range of different licenses, but in all cases the source code is included or available. Middleware available direct from eCosCentric includes:

eCosPro-C/PEG	Graphics user interface library (C based API)
eCosPro-PEG+	Graphics user interface library (C++ based API)
USBware	USB Host, Device and OTG stacks
CEE-J	Embedded Java virtual machine
EXtremeDB	Embedded in-memory database
OS Changer	Porting aid to migrate existing applications
eCosPro-CAN	Controller Area Network device API and drivers
CANopen	CiA standards based CAN protocol stack
eCosPro-SecureShell	SSH-2 compatible Secure SHell daemon
eCosPro-MMFS	High performance multimedia file system

Q. **What is the size of the current *eCos* user base compared to other embedded operating systems?**

A. Due to its open source nature, the exact number of users is hard to gauge precisely. Some of the best indicators of relative market share are provided by the yearly IDC and CMP embedded surveys. These show sustained growth for *eCos*, with the 2007 surveys both placing *eCos* at around 5-6% usage in the market. The operating systems sector is pretty diverse and fragmented, but that places *eCos* roughly on a par with *Integrity*, *QNX*, and *ThreadX*.

LYNUXWORKS: *LINUX* FOR EMBEDDED AND REAL-TIME SYSTEMS

25 October 2007: *Linux* for Embedded and Real-time Systems

INTERVIEWEE. ROBERT DAY
VP, MARKETING
TEL. (408) 979 3900
EMAIL. rday@lnxw.com
COMPANY. LYNUXWORKS, INC
WEB. http://www.lynuxworks.com/

Q. **First of all, tell us a little bit about yourself and your position at LynuxWorks.**

A. I graduated with a degree in computer science and became a software developer early in my career developing SCADA systems. Twenty years ago I joined Microtec Research and got involved with embedded tools developing an early version of the *XRAY debugger*. I have worked with many different real time operating systems and tools throughout my career and I now enjoy planning and promoting the different solutions that we offer at LynuxWorks.

I am particularly interested in open standards, and have been the chairman of the embedded workgroup of the Eclipse foundation for the last three years. I carefully watch the impact that both open-source and open-standards have on the embedded world, and how they can be brought into the products here at LynuxWorks.

Q. **Can you tell us in a nutshell what is LynuxWorks's "*Linux* Strategy?" What is your value proposition for embedded and real-time designers?**

A. LynuxWorks offers a number of complementary options to developers who are considering using an embedded *Linux*/UNIX operating system. Our embedded *BlueCat Linux* uses the standard *Linux* distribution, and adds embedded features, utilities, tools and support to make it an easy-to-embed *Linux* version. We believe that most *Linux* users do not want something that deviates too far from the standard *Linux* distributions, as this moves away from the openness and vendor neutrality that makes using *Linux* appealing.

As you will see from later examples, LynuxWorks doesn't just provide a *Linux* operating system; we provide a family of compatible open-standards based operating systems that offer a choice of features and performance to meet different embedded systems needs.

Q. **The embedded and real-time *Linux* community is growing, no doubt about it. How does LynuxWorks distinguish itself from other *Linux* competitors such as MontaVista or Wind River as a provider of *Linux* for real-time / embedded?**

A. LynuxWorks is unique in that we provide a family of compatible open-standards based operating systems allowing the users to select which OS meets their needs best. The most significant feature of our products is the compatible APIs across the operating systems, enabling migration across operating systems and offering maximum code re-use.

We offer *Linux* developers four options:
1) *BlueCat Linux* – available on most major embedded platforms
2) *LynxOS* with its unique *Linux* Application Binary Interface (ABI). This allows users to use the hard real-time *LynxOS* operating system, but run *Linux* applications on it un-modified.
3) *LynxOS-SE* time/space partitioned hard real-time operating system, which also utilizes the *Linux* ABI.

4) *Linux* as a guest operating system under the *LynxSecure* real-time embedded hypervisor. This allows *Linux* and *Linux* Applications to be run in a para-virtualised separation system, alongside other operating systems, giving maximum protection and security when using *Linux*.

Q. **LynuxWorks is somewhat unusual in offering both a "hard" RTOS (*LynxOS*) as well as a commercial *Linux*. What is your strategy on the relationship between the two, especially for potential developers using one or both?**

A. The *Linux* ABI provided with *LynxOS* enables the most seamless application migration and re-use over and between the *LynxOS* and *BlueCat* families. Add to this a common set of development tools based on the open-standard Eclipse framework, and a dedicated embedded support and services department, and LynuxWorks has a truly unique and inviting set of solutions for building complex embedded systems.

The open-standards based POSIX API is the native interface for developers using *LynxOS*. This means that code written to run on *LynxOS* is also reusable across other POSIX operating systems. *Linux* itself is close to POSIX compliant, which means that POSIX applications can be easily migrated to *Linux* and vice versa. This close link between POSIX and *Linux* is also what makes the *Linux* ABI for *LynxOS* very efficient, as there is a direct mapping between *Linux* calls and POSIX calls. This means that most *Linux* applications run faster on *LynxOS* using the ABI than on top of *Linux* itself.

Q. **In the discussions before this interview, you indicated that LynuxWorks has some really exciting angles on "multicore" and/or "virtualization" and the company's RTOS strategy. Can you share some details with us?**

A. We see the CPU market moving towards both multi-core processors and virtualized processors, and our products take advantage of these new architectures to help boost performance in complex real-time systems. Our 5th generation of the *LynxOS* RTOS now supports symmetric multi-processing (SMP) across multiple cores. This allows the OS to decide how to best utilize the processors available power and allocate the processes in the system accordingly.

The *LynxSecure* embedded hypervisor takes advantage of the latest hardware virtualization technology to help run fully virtualized or para-virtualised guest operating systems like *Linux* in an efficient way. In recent tests, we have had multiple copies of *Linux* running on a single Intel VT based processor para-virtualised in their own partitions, without notable degradation in performance. We are now looking at having fully virtualized guest operating systems such as Windows available, and again using hardware and software virtualization techniques to allow fast performance.

Q. **You recently announced a partnership with FPGA maker Xilinx. Tell us about *Linux* and/or *LynxOS* on FPGAs.**

A. We have ported our *BlueCat Linux* across both of Xilinx embedded processor families (PowerPC and MicroBlaze), giving Xilinx customers a consistent set of operating systems regardless of which processor is embedded in the FPGA. The MicroBlaze processor is available with or without an MMU, so we introduced *BlueCat*-ME for this architecture, which is configurable based on the presence of an MMU in the design.

Because an FPGA is configurable hardware, we made *BlueCat* easily configurable too, matching the selected hardware design. Xilinx provides an embedded development kit (EDK) that allows the hardware to be easily configured; we then added configuration options to the EDK to enable *BlueCat* to be easily tailored to meet the new hardware requirements. *LynxOS* is also available on the Xilinx PPC embedded processor, giving Xilinx customers the same *Linux* to *LynxOS* migration options as stated above.

Q. **Is it possible using your solution to employ *Linux* in "hard" real-time systems such as those for military or safety-critical? What factors might**

encourage a developer to consider *Linux* for those markets, or would your recommend he stays solely with *LynxOS*?

A. For safety critical systems, a certified and certifiable operating system with software partitioning is generally required. As there is a cost per line of code to safety certifies software, *Linux* has been generally considered too large to be commercially viable. Our *LynxOS*-178 operating system is time/space partitioned, has a POSIX conformant API, and is the only operating system to be awarded a Reusable Software Component (RSC) from the FAA, allowing the OS to be seen as "pre-certified" and therefore reducing the cost of system certification.

For security systems, our *LynxSecure* separation kernel allows *Linux* to be run in its own secure partition as a guest OS, and hence allows the use of *Linux* in these types of applications. There is still a cost per line of code certified, but the *Linux* part (in its own partition) does not necessarily have to go through those high costs of certification. So, *Linux* is more viable and more widely used in military systems than safety-critical avionics systems.

Q. **LynuxWorks is known as a major RTOS vendor in the military and safety critical market. Your website has many impressive design wins in those areas. Yet *Linux* is deployed in the telecom infrastructure vertical as well as in the mobile phone areas. Does LynuxWorks offer solutions for non-Military / non-Safety Critical areas?**

A. The LynuxWorks family of operating systems is widely deployed across the embedded market. The open standards, reliability, safety and security features of our operating systems meet the needs of the military and avionics markets very well, especially for complex, connected real-time systems. These same features are also a very good fit for other markets such as telecommunications, medical, transportation and industrial control. Having a broad range of embedded operating systems from *Linux* through to hard real-time POSIX allows our customers to use our products in a very wide and diverse set of embedded devices.

Many embedded devices used outside of military and aerospace now have a requirement for safety and security, and having a pedigree with military and commercial avionics (especially in certified systems) goes a long way in demonstrating the suitability of our products in other markets. The need for security of information and communications is now proving to be a requirement in the financial, medical and industrial control markets, and the ability to run virtualized guest operating systems such as *Linux* or Windows in a secure system using *LynxSecure* is very appealing.

Q. **Thank you for your interview**

MICRO DIGITAL INC.: RTOS LICENSING MODELS

9 November 2007: RTOS Licensing Models

INTERVIEWEE. RALPH MOORE
PRESIDENT
TEL. TELEPHONE CONTACT
EMAIL. info@smxrtos.com
COMPANY. MICRO DIGITAL INC.
WEB. http://www.smxrtos.com/

Q. **First of all, tell us a little bit about yourself and your responsibilities at Micro Digital.**

A. A partner and I founded Micro Digital in 1975 as a consulting company. Our objective was to design microprocessor based systems for customers. Our first project used the 8080 in a gas pipeline metering system. In 1988, I decided to go into the RTOS business. *smx* was released and first sold in 1989. Today, I am the President and chief salesman of the company and my son, David, is Director of Development. Between us, we handle marketing.

Q. **What are Micro Digital's offerings in the RTOS and embedded software space? Please give us a very brief run-down of the products.**

A. We offer the *SMX RTOS* consisting of the smx hard real time multitasking kernel, TCP/IP stack and networking protocols, FAT file system and flash file system, USB host, device, OTG stacks with class and function drivers, and other products. We focus on *ARM*, *ColdFire*, and *x86* processors and we offer out-of-the box solutions that include BSP code and drivers.

Q. **We touched base briefly before the interview, and you indicated that you wanted to explore "licensing" issues in terms of the RTOS choice. For those who don't know, can you explain briefly the basic models for licensing an RTOS?**

A. The traditional model is the royalty model where there is a moderate up-front charge per development seat, then a royalty for every unit shipped. Microsoft, Wind River, QNX, and others still use this model. The dominant model, today, for embedded software is the no-royalty, one-product license. For this, there is a relatively large up-front charge for one developed product and no royalty per unit shipped. My feeling is that the royalty model makes more sense. However, by the time that I and contemporary RTOS vendors (Accelerated Technology, A.T. Barret, U.S. Software, and others) entered the market we needed a way to beat entrenched competition, so we independently decided to follow the lead of Kadak and we each offered no-royalty pricing with full source code.

Q. **What is your opinion of the "royalty free" movement? Most - if not almost all - RTOSes these days are "royalty free." What do you think of that?**

A. I think the royalty model makes more sense because what the customer pays is proportionate to the value of the software to him – i.e. the more he sells of his product, the more he pays. However, most customers do not want to keep track of shipments and royalties. Hence, even among royalty companies, royalty buyouts are common. With the royalty-free method, we obviously cannot sell one license to a customer to use forever. We would soon be out of business. Hence, our competitors and we limit the license to one developed product.

Defining one product is difficult. If there are two versions, one with an industrial hardened case and the other with a normal case, and everything else is identical, are there two products or one? If some versions have more inputs, but everything else is identical, is there just one product? I think you can see that defining what is one product leads to a lot of negotiating. I had one prospect argue with me that their refrigeration and heating units were all one product because they all moved heat from one place to another! What do you do if there is single "platform" used in diverse products such as parking systems and data acquisition systems? Many large companies like to do this to keep costs down. What do you do if your products are part of a technology used in diverse moderate-cost products of which only 10 or 20 are made per year?

Q. **Micro Digital is a representative RTOS vendor. Can you explain for us your own licensing and purchase models?**

A. We offer the one product royalty-free license as well as other licenses. About 2 years ago, I decided to introduce a 50,000-unit production limit on our least expensive one product license. Most true embedded systems have lifetime productions of less than 50,000 units, so this is no problem for them. Products with greater than 50,000 lifetime productions tend to be consumer or quasi-consumer products and I think they should pay more. Most people agree with this. There is a scale of percentage increases for higher limits, up to 50% increase for no limit. Some people have not liked this plan, but generally, it has been accepted well. I find that it allows me to be more liberal on the one-product definition. Many of our competitors deal with this problem by sharpening the one product definition when they find out that large production quantities are involved. Needless to say, they don't make many friends this way.

When one product does not fit, then customers often want to negotiate license buyouts so they can develop many products with one license. I would prefer not to do this, because it is very hard to have a stable cash flow if you must start every sale from scratch. However, the customer is always right. Hence, we offer multi-product licenses. However, as with the single product license we cannot sell one license forever after to each customer or we will soon be out of business. Hence, we and other RTOS vendors limit multi-product licenses to one processor. However, this too can be tricky to define. Is it the same processor if it just has more memory? What if it has an additional peripheral? One processor is not good enough for many customers – they want to use an *ARM7* for low-end products and an *ARM9* for high-end products.

We have addressed this range of requirements with a range of licenses: Product Line (2:1), Product Family (3:1), and site license (4:1). Each, in succession, is more liberal. For example, the product line is limited to one processor; the product family allows two processors; and the site license allows 3 processors. This is a new licensing plan and it is too early to tell how well it will work. I like it because it provides a smooth range of prices to customers and precise definitions of each license. it provides

Q. **Microsoft argues that royalty free is actually not beneficial to developers as it puts 100% of the risk and cost on the early design phase. Microsoft argues instead that "royalties" (as it charges for XPe and CE) create a "shared success" model in which the RTOS vendor is motivated to ensure that the developer actually ships real products quickly. What is your response to that?**

A. The main statement is correct. In general, 1/3 of all projects never make it to market. Another 1/3 are marginal successes due to being late to market, too expensive, not meeting the market's requirements, etc. Hence, only 1/3 of the projects pay the bills for royalty companies. Thus, the RTOS vendor who charges royalties is definitely sharing risk with the customer and is motivated to help that customer succeed. Many times, small companies, startups, and others have asked me if we will offer them a royalty. I tell them it is possible, but if they later want to do a buyout, it will cost 3X our list prices. They don't like this, and so far I have not had any takers.

There is a dark side to royalties, also. I think the royalty RTOS companies tend to provide better support to the well-funded, well-staffed projects that look like big winners. It is sort of like betting on a horse race. The small guys and probable losers get neglected. One reason for this is that there often is not enough money in the up-front payment to provide much support. It is ironic that you cite Microsoft, because Microsoft does not provide good RTOS support – it charges $200-300 per "incident" and it relies on newsgroups to provide most support – no better than Open Source.

The other side of the coin – to say that the no-royalty companies provide poor support is false. Micro Digital provides very good support and most of our no-royalty competitors provide at least adequate support. In general, the no-royalty RTOS vendors are small and the programmers who wrote the code provide support. Obviously, this is better than providing support with go-betweens. We are motivated to provide good support because happy customers come back for more products.

Q. ***Linux*, in contrast to Microsoft, is both "open source" / GPL licensed and "royalty free." What is your take on the "open source" license such as the GPL?**

A. I think the whole Open Source movement is based upon false premises and is harmful In as much as *smx* was focused on the x86, when *Linux* appeared, it probably caused us more harm than it did our competitors. In general, I think it has suppressed growth of RTOS companies and of innovation in the RTOS market. I have seen studies that show that the using *Linux* on large projects actually costs much more than using the most expensive commercial RTOSs. Yet, people persist in thinking that it is free.

I am starting to see a little bit of sense returning to the market. Some people I talk to have tried Open Source, found it wanting, and have returned to the commercial fold. However, may others sill consider it to be the best solution. Many engineers seem to think that they work for free and the man-months necessary to learn *Linux* and provide their own support matter little. If these people were paying their own salaries, what decision would they make? I think many of the engineers who choose Open Source have the same motivation as the ones who chose to design in-house RTOSs, prior to *Linux*. Designing an in-house RTOS is no longer politically correct, but picking Open Source is.

Q. **QNX recently introduced a dual-source license with non-commercial uses having full access to source and uninhibited use of the RTOS. Commercial users must license. What is your opinion of this "hybrid" open source sort of license?**

A. This "hybrid" is not a hybrid at all – it is practiced by most Open Source projects and to me is another hypocritical action in the Open Source community. Code is either free or it is not free. How can Open Source project leaders use the free work of community members then turn around and sell it for a profit? This the moral high ground? Of course, QNX is a commercial company and they have a different motivation to introduce this licensing. Many other commercial companies have done the same thing. I think QNX's objective is to get their RTOS in greater use and hopefully that will lead to more sales. I do not know whether this is working for them. I tend to find that the people who want free lunches will never pay for them, so I have so far avoided offering *smx* this way.

Q. **Have you considered much of the legal issues in terms of licenses? We hear, for example, that many are worried that Linux contains commercial products illegally introduced into it, and/or that Linux might force software code to be "open sourced" itself. What is your take on the legalisms around RTOS licensing?**

A. This is the reason that many attorneys and corporate leads consider GPL to be pollution. Many big companies practically insist that we swear on a stack of Bibles that there is no GPL pollution in our code – they do not want to end up in messy law suites. I understand that the *Linux* Community has allowed commercial code to be linked to the *Linux* kernel,

without requiring it to be Open Source, but the more radical members want to end this practice as of January 2008. Our position has been to steer clear of *Linux*. I refuse to allow any of our code to be linked with the *Linux* kernel.

Q. **Many vendors offer a "per seat" license, a "per project" license, and even a "per processor" license. Do you think any of these are especially advantageous? What factors would you recommend a developer pay attention to in terms of potential license "gotchas?"**

A. Some of the compiler companies have introduced per seat RTOS licenses, probably because this is how they sell their other software. Customers are allowed to do any number of projects without paying more. I think these vendors will find that this is a poor business plan – there is not enough money to fund supporting new processors, to provide good support, and to keep an RTOS team together. After all, those of us who have been in the RTOS business much longer are not getting fabulously wealthy. There are good reasons why we charge as we do. I expect to see these plans fade away, in time. I have not studied any RTOS per seat licenses, but I would think that the purchaser should expect to write his own BSP code and drivers and should not expect much support. Also, as with chip vendor software, the code may not be well written or well debugged. Probably the programmers who wrote it have already been moved off to other projects – that is the problem with something that is not financially viable.

The per project and per processor licenses are akin to what we offer.

Q. **Thank you for this interview.**

MONTAVISTA: *LINUX* FOR EMBEDDED AND REAL-TIME SYSTEMS

14 December 2007: *Linux* for Embedded and Real-time Systems

INTERVIEWEE. JAMES READY
CTO AND FOUNDER
TEL. (408) 572-7830
EMAIL. jim@mvista.com
COMPANY. MONTAVISTA SOFTWARE
WEB. **http://www.mvista.com/**

Q. **First of all, tell us a little bit about yourself and your position at MontaVista.**

A. I am the founder and CTO of MontaVista Software.

Q. **Can you tell us in a nutshell what is MontaVista's "*Linux* Strategy?" What is your value proposition for embedded and real-time designers?**

A. Our customers are in the business of developing and delivering innovative embedded devices. *Linux* is rapidly becoming the operating system of choice for any embedded application. MontaVista delivers the following industry leading, proven and quantifiable benefits to customers of embedded *Linux* designs:

- faster time-to-market
- reduced cost
- lower risk

We deliver these benefits by providing proven, high-quality platforms, including commercial *Linux* OSes, advanced *Linux*-focused development tools and expert *Linux* professional services.

Q. **Let's start with some very basic questions. The kind a Dilbert-like manager might ask of an engineer who is reading this Guide, and investigating *Linux* as a possible RTOS for his new embedded project. First of all, why should someone pay MontaVista for *Linux*, when *Linux* is "free?"**

A. First and foremost, the *Linux* community defines "free" in the context of *Linux* as free as in freedom of speech, not free as in free beer.

When a MontaVista customer pays for MontaVista *Linux*, he is not paying for the intellectual property embodied in the *Linux* code – as customers of proprietary embedded OSes do. They pay for what MontaVista adds to the code, including hardware enablement, integration of code from several open source projects with each other, integrated development tools, legal protection, and software hardening. MontaVista invests heavily in productizing the *Linux* code into a consistent platform customers can reliably base their designs on. Customers also receive professional support and ongoing software maintenance for the platform they are using in their designs.

MontaVista customers therefore are freed from having to invest in integration, testing, and the ongoing sustaining engineering of the core platform. This frees resources to focus on developing value-add device capabilities.

MontaVista customers therefore pay us for getting to market faster, at lower cost and reduced risk

Q. **What precisely does the developer "get" from MontaVista? A hardened *Linux* distribution? A true real-time *Linux*? Product support during**

development? Longevity? What are the bullet points vis-à-vis *Linux* that a developer gets by going with MontaVista?

A. At the highest level, MontaVista *Linux* consists of a proven, commercial quality *Linux* platform for our customer's target processor architecture and an integrated development environment for efficient cross development for the customer's target environment.

MontaVista *Linux* quality is achieved by including thousands of bug fixes not provided in the original open source code bases, rigorous testing in MontaVista's QA test framework with over 35,000 tests and extensive documentation and ongoing sustaining engineering (bug fixes, security patches) for released platforms.

MontaVista's development environment *DevRocket* is comprised of Eclipse plug-ins optimized for both system and application development in a cross-development environment. *DevRocket* is based on open source tool infrastructure, and provides an easy-to-use interface for advanced analysis tools, including memory leak detection, memory analysis, system tracing, and performance profiling.

Q. **Tools support for "free" *Linux* of the Kernel.org variety is not entirely of commercial quality. MontaVista offers some development tools, including your *DevRocket IDE*. What sorts of development tools do you offer that might be superior to the "free" or "open source" tools available for *Linux*?**

A. As mentioned above, MontaVista *DevRocket* leverages open source tool infrastructure to provide *Linux* tools for *Linux* developers, as opposed to proprietary RTOS tools ported to *Linux*. *DevRocket's* value add is to make these tools available through an intuitive user interface instead of a providing only a difficult to learn and interpret command line interface.

Q. **What is *MontaVista TestDrive*? What are its benefits to developers looking at embedded *Linux*?**

A. *MontaVista TestDrive* is an innovative, virtual evaluation environment that allows to our customers to quickly determine if MontaVista *Linux* addresses the requirements of their project and for them to get an opportunity to experience our development environment.

Setting up an actual evaluation environment is often a lengthy and laborious effort, yet customers are faced with ever-decreasing project cycles and less resources to complete their projects.

TestDrive provides a complete environment of MontaVista *Linux* available over the Internet to our customers, including the distribution running on virtualized target hardware and our cross-development environment *DevRocket*. This web-based evaluation environment is set up in minutes, thus making it fast and easy for our customers to evaluate our products.

Q. **What about services? We have heard that often times the real "sale" involved in a hardened *Linux* is a sale of a stream of services. What are the services offered by MontaVista for *Linux*? How does one commit to pay for them?**

A. MontaVista has a very experienced, *Linux*-savvy Professional Services team. Our service offerings span the range from *Linux* hardware porting and device driver development, to middleware and application integration and development. These services are delivered in a traditional SOW-based quotation system.

Q. **The embedded and real-time *Linux* community is growing, no doubt about it. How does MontaVista distinguish itself from other *Linux* competitors such as Wind River or TimeSys as a provider of *Linux* for real-time / embedded?**

A. MontaVista's differentiation lies in:

- **Proven:** Recent independent analyst data (from Embedded Market Forecasters, 2007) shows that customers using MontaVista's *Linux* are faster to market, require less resources, and benefit from lower total cost of development compared to other *Linux* options, including both non-commercial *Linux* and *Linux* from Wind River.
- **Quality:** MontaVista invests heavily in delivering a high quality *Linux* distribution. Our competitors pass through the original *Linux* source code – sometimes referred to as "pristine," leaving it up to their customers to fix bugs that hinder progress in their projects.

Linux focus: MontaVista is entirely *Linux*-focused. All our investments are directed to make our customers more successful in using *Linux* in their projects. We do not have to make a delicate balance between investing in proprietary legacy OSes and open source *Linux*. This allows us to work more closely with the open source community and to drive further community-based enhancements on behalf of our customers.

Q. ***Linux* licensing is complex because the GPL is such a different animal than standard commercial licensing. Because there have been concerns about *Linux* licensing both in terms of commercial products illegally brought "in" to *Linux* and in terms of upwards "GPL contamination" of *Linux*-based products, how does MontaVista help its customers with *Linux* licensing issues?**

A. MontaVista has the industry's most experienced open source and *Linux* legal team. Our team guides customers through the legal in and outs of developing with *Linux* and other open source licensed software. Our team has worked successfully with the legal teams of the world's largest companies to enable them to make a sound legal and business decision to employ *Linux* in their products.

Q. **MontaVista has recently been promoting *Linux* for mobile devices. What vertical markets do you see the most potential for *Linux* expansion and why?**

A. That word "recently" is misleading. Actually, MontaVista has been promoting *Linux* for mobile devices for several years. Our product *MontaVista MobiLinux* just announced version 5.0.

As shown by multiple analyst reports, *Linux* is already the market share leader in operating system technology across many embedded markets. *Linux* is forecasted to rapidly gain additional market share from legacy proprietary RTOSes such as *VxWorks*. Within just a few years, *Linux* is predicted to power over 80% of all new embedded applications. We see strong growth and increased *Linux* adoption virtually across all segments of the embedded market, including consumer devices, telecommunications, data communications, medical, industrial, automotive and defense applications.

Q. **How would you compare and contrast the experience of downloading *Linux* from Kernel.org and working with MontaVista on a *Linux* project?**

A. Every time development groups download *Linux* from the many repositories on the net, compile the sources, declare "We have *Linux* up and running," and then commence their application development based on that *Linux*, they are putting their company at great risk. The engineering team has made the fatal assumption that *Linux*, in its raw, open source form, is a fully tested, production-ready, maintained and supported operating system ready to use in a real product. That is a fatal assumption. The key developers of *Linux* have made it very clear publicly that the goal of the core *Linux* development team is cutting-edge, fast-moving R&D.

"The kernel.org community is no longer delivering a quality commercial product, it delivers cutting edge technology and depends on commercial distributions to clean it up and make it into a commercial product" – Andrew Morton (2.6 kernel maintainer) at Linux Foundation Summit 2007.

Pure unbridled leading edge software development, with the functions of integration, stabilization, testing and quality assurance and maintenance left for others to implement. Only after those processes are completed, by someone somewhere, is *Linux* then a viable production-ready, fully tested operating system.

So the message is simple: Unless you're comfortable in forgoing the necessary work it takes to produce a commercial-quality production-ready *Linux* and just having a "download and go" strategy, with all its attendant risks, the do-it-yourself approach puts you in the *Linux* OS business, with all its costs in time and people and schedule risks. At that point, we believe the wise business decision is to buy rather than build. After all, is your company really in the *Linux* OS business?

Q. **Thank you for this interview.**

QNX: NEW LICENSING MODELS FOR REAL-TIME SYSTEMS

25 October 2007: New Licensing Models for Real-time Systems

INTERVIEWEE. SEBASTIEN MARINEAU-MES
VICE PRESIDENT, ENGINEERING
TEL. 613-591-0931
EMAIL. INFO@QNX.COM
COMPANY. QNX SOFTWARE SYSTEMS
WEB. http://www.qnx.com/

Q. **First of all, tell us a little bit about yourself and your responsibilities at QNX Software Systems.**

A. I've recently been named vice president of engineering at QNX Software Systems, where I'm responsible for all R&D activities, including software development and quality assurance. Before becoming VP, I was director of QNX's operating system group, where I oversaw the development of OS technologies, networking stacks, multimedia middleware, and, most recently, the new *QNX hybrid software model*. In 2002, I became the first project leader for the Eclipse C/C++ development tools (CDT) project. I also led the team that created QNX Momentics, the first Eclipse-based IDE for embedded development

Q. **For those of our readers who do not know, can you give us a quick run-down of QNX's products for embedded and real-time systems?**

A. QNX offers a large portfolio of OS technology, development tools, middleware, and professional services for the embedded developer. These include the *QNX Neutrino RTOS*, the *QNX Momentics development suite*, and *QNX Aviage middleware family*, which allows developers to quickly integrate multimedia, 2D/3D graphics, or voice-activated interfaces into their embedded products. QNX also offers an award-winning suite of debugging tools and multiprocessing technologies for multi-core systems.

Q. **On September 12th, QNX made a radical change to its licensing model. Can you summarize that change for us?**

A. For the first time, any developer can enjoy quick, hassle-free access to QNX OS source code. Developers can also modify and distribute that code under commercial-friendly licensing terms. In fact, QNX has introduced three licenses: one for commercial users, one for noncommercial users, and one for QNX technology partners. For instance, a student or hobbyist developer can choose the noncommercial license, which provides free, perpetual access to QNX development tools and OS technologies, including OS source code.

These licenses form part of QNX's new *hybrid software model*. We created the model with a simple goal in mind: to combine the benefits of proprietary software (quality product management, published roadmaps, professional support) with key benefits of open source software (easier customization, faster bug fixes, community support).

Q. **Licensing, royalties, and development models are all very important to any engineer selecting an RTOS. How does QNX's new model compare and contrast with that of *Linux*?**

A. If you're a developer, the new QNX model looks and feels like an open source project: You can access the source code, download bug fixes or new features in real time, and participate in the development process — whether by providing input, posting bug

reports, or offering up code modifications. From that perspective, it is very similar to *Linux* and other open source projects.

The big difference is in the licensing: Rather than publish its source code under a reciprocal license like the GPL, QNX has opted for a licensing model that offers flexible terms for creating derivative works. So if you modify QNX source code and ship it in a device, you can either keep your modifications proprietary or donate them back to the QNX community — the choice is yours. With *Linux*, you would have publish your code modifications and make them available to everyone, including your competitors.

Also, as a commercial RTOS vendor, QNX offers a product roadmap for embedded developers and tests its software in an ISO9001-2000 certified environment. QNX also invests heavily in maintaining the "cleanliness" of its source code. As a result, embedded device manufacturers don't have to waste time establishing the provenance of source code or determining whether it infringes on someone else's patents.

Q. **Can you compare and contrast it specifically with the models of vendors like MontaVista or Wind River, that essentially re-package *Linux* and sell a stream of *Linux*-related services?**

A. We focus strongly on innovation — on delivering unique technologies that give our customers a competitive edge. This is a very different approach from repackaging a *Linux* distribution. For example, our adaptive partitioning technology offers a combination of security, realtime determinism, and ease of software integration that no other OS, including *Linux*, can offer.

At the same time, our published source code and new, transparent development model offer the key benefits of an open source OS like *Linux*. So in addition to leveraging our unique technologies, developers can also contribute to the evolution of those technologies or modify those technologies for their own purposes.

Q. **For the commercial developer, does this person continue to pay run-time royalties if he uses the *QNX Neutrino RTOS*? What purchase models exist for working with QNX?**

A. Customers continue to pay runtime royalties if they ship QNX OS technology or middleware in a commercial device. The new model doesn't change that. Nonetheless, royalty arrangements are flexible and depend on several factors. For instance, if a device manufacturer knows that their product will ship in large volumes, they will sometimes negotiate an "advanced" royalty payment with us before shipping the product. That way, they can reduce total costs.

Because we sell a royalty-bearing RTOS, our success is intimately tied to our customers' success. Thus, we're highly motivated to help customers get to market in the shortest time possible.

Q. **If we read between the lines about QNX's new model, it sounds like what a commercial user gets is basically early access to source code. How is this model different than just downloading a free demo? Is the big difference the availability of source code?**

A. We aren't simply publishing our source code; we're also making our development process transparent, for everyone to see. From now on, QNX will develop its products out in the open and allow the QNX community to participate in the process. Put simply, customers can now participate in developing the software foundation that their devices rely on.

Moreover, our technology partners and noncommercial users now have quick, easy access to the full binary version of our *QNX Momentics development suite*, which includes state-

of-the-art visualization tools and the *QNX Neutrino RTOS*. This isn't a crippled demo or eval version, but the real thing.

Q. **QNX plans to "grow" a community around its new model. What is your vision for the QNX community? How do you think this can change the "development model" used today to deploy and enhance real-time OSes?**

A. The new QNX model will definitely put pressure on other RTOS vendors to follow suit. The only question is, how long will they resist before finally giving in? We've seen this pattern before: QNX was the first RTOS vendor to offer an Eclipse-based development environment, which our competitors criticized until they realized that Eclipse was a much better way to go. In the meantime, QNX got a huge headstart on its Eclipse tooling.

Likewise, QNX now has a major headstart on a development model destined to achieve widespread adoption. Developers today cut their teeth on open source OSes like *Linux*, and that has changed their expectations about source code availability and developer collaboration. RTOS vendors must either satisfy these expectations or lose market share.

From the community perpsective, one of the biggest benefits of a hybrid software model is that the vendor is no longer the bottleneck. For instance, if some community members wish to target a market segment or processor architecture QNX doesn't currently support, no problem: They can spin their own version of QNX. Just as important, they don't have to enter into complex source code licensing or negotiations to get started — though they will, of course, need to negotiate royalty payments if they commercialize the product.

Q. **How do you see the new QNX community helping developers? Most embedded designs are unique, or close to it. How can one developer really help another?**

A. Device-specific issues tend to occur at the driver level. Most QNX customers use our off-the-shelf drivers and focus on higher-level development, where they all work with the same development tools, OS kernel, file systems, and APIs. So if a netcoms developer has a question about, say, symmetric multiprocessing, the developer who provides the answer could easily work in the medical or industrial control market.

In fact, we see an increasing amount of crossover between our various market segments. Case in point: The same graphics software that QNX originally developed for the industrial automation and casino-gaming markets is now popular in automotive navigation systems. So we fully expect developers from different markets to help one another on a regular basis.

Q. **What is "Foundry27?" Are there any novel Web-centric ways that QNX will change the RTOS industry?**

A. The *Foundry27* web portal is "command central" for QNX software development. It provides source code repositories, discussion forums, wikis, blogs, technical articles, and a variety of other developer resources. It even includes a community bazaar that allows developers to post applications and tools of interest to other community members.

Foundry27 is turning RTOS development inside out. The source code "drops" you see on *Foundry27* are, in fact, real source repositories. What used to be a closed internal development process is now open and publicly accessible on the web — everyone can see the software evolve in real time. That's a completely new way to develop and enhance an RTOS, and, judging by our customers' enthusiasm for the model, other vendors will inevitably try to emulate it.

Q. **Where can developers to find out more about your new hybrid model?**

A. I recommend they read Lawrence Rosen's white paper on the *QNX hybrid software model*, at http://www.qnx.com/download/feature.html. Rosen is the former legal counsel to the Open Source Initiative (INI) and he does a great job of explaining what the *hybrid software model* is and isn't.

Q. **Thank you for this interview.**

TIMESYS: LINUXLINK AS A MODEL FOR LINUX DEVELOPMENT

25 October 2007: LinuxLink as a Model for Linux Development

INTERVIEWEE. GENE SALLY
PRODUCT MANAGER
TEL. 1-412-232-3250
EMAIL. info@timesys.com
COMPANY. TIMESYS CORPORATION
WEB. **http://www.timesys.com/**

Q. **First of all, tell us a little bit about yourself and your position at TimeSys.**

A. I am Product Manager for TimeSys. I have been working with *Linux* for 10 years, 7 of those in an embedded environment with TimeSys. I also co-host *LinuxLink Radio* (http://www.linuxlinkradio.com/), a popular podcast in the world of embedded *Linux*.

Q. **Can you tell us in a nutshell what is TimeSys's "*Linux* Strategy?" What is your value proposition for embedded and real-time designers?**

A. The short answer is that TimeSys concentrates on helping engineers using *Linux* to be successful in their embedded projects.

TimeSys' strategy involves enabling embedded *Linux* developers by supplying them with the *Linux* kernel, root file system components and tools appropriate for the customer's target board and environment. Our users are looking to avoid the pitfalls of configuring a development environment, porting a kernel and creating the *Linux* distribution that runs on the device. Many of these users are new to *Linux*, but a substantial portion have solved this problem in projects past and don't want this process to take away time and resources from the project itself.

All embedded *Linux* projects require some level of customization, with something as simple as configuring the kernel or root file system, or as complex as building custom device drivers. Furthermore, the rapid turn-around required in the embedded space means that using an external consultant or traditional embedded *Linux* services vendor introduces latency in the schedule that can result in a company losing their first-mover advantage. *LinuxLink* provides the right starting point so subscribers can be productive faster, without the encumbrances (financial and schedule) of an external company. For example, each *LinuxLink* subscription contains a cross-compiler that's ready to use for both glibc and uclibc libraries. Pair that with pre-compiled root file system components for both libraries and subscribers can assemble a custom root file system using a combination of components from *LinuxLink* and other open source projects to create a custom platform in as little as a day.

By providing the components designed for customization, the user maintains control over this critical aspect of their project and avoids the time consuming and costly process of engaging in a professional services contract.

Real time on *Linux* is another great proof-case of our *Linux* strategy. *LinuxLink* subscribers are kept up-to-date with the latest code ready for this quickly evolving technology. Customers working on cutting-edge, next-generation devices need the most recent technology to meet their product requirements. With *LinuxLink* our subscribers can start their projects using latest kernels and real time patches ready for their target processor and keep using that kernel throughout their project.

Q. ***Linux* for embedded systems has generally been either (a) downloads from kernel.org or other "free" distributions or (b) hardened *Linux* distributions**

plus services from companies like MontaVista or Wind River. *LinuxLink* is a very different model. Can you compare and contrast it with the others?

A. Let's take this one thing at a time. One of the first things those approaching an embedded *Linux* project realize is that the kernel is one part of a larger project. The contents on kernel.org contain just a kernel, which is source code, and isn't useful unless it can run on a target board. To do that, most users need a cross-compiler environment and that tool must usually be built from source, frequently including patches located at third party sites or posted on mailing lists in order to work correctly. In addition, the kernel itself won't run unless the user supplies a root file system, which requires locating the right source and patches and getting the packages cross-compiled.

Traditional embedded *Linux* software vendors hide this complexity from users through a professional services contract. Customers pay for (and wait for) the services company to find the right pieces and put them together, in effect ceding control of their project to an entity with incentives and priorities that can, and frequently do, vary from that of their customers. For example, a traditional embedded *Linux* services company recently pre-announced support for *Linux* on a year-old kernel version. This strategy fits that of the vendor, but what about their customers?

LinuxLink helps customers succeed by providing an up-to-date *Linux* kernel, components to build the distribution and the tools necessary to put everything together along with expert help and advice. Customers are then in control of their project with much higher productivity, because of the ready-to-use components supplied through *LinuxLink*. Remember, all embedded *Linux* projects require customization; *LinuxLink* is designed from the ground up to make this as easy as possible.

Q. ***LinuxLink* seems to be chip-centric. If a developer's target chip is not supported by *LinuxLink* what should he do? Or what if he is working in the Telecom infrastructure vertical where usually *Linux* is being deployed on third party boards - does *LinuxLink* apply to those situations?**

A. This is the exact use case we had in mind when designing LinuxLink. The vast majority of embedded deployments happen on custom boards. By using a LinuxLink for a processor, the user is free to perform the small amount of customization necessary to get Linux running on a custom board, at a fraction of the cost.

Traditional embedded companies, who have built their models around providing services, try to make this step seem as difficult and mysterious as possible. TimeSys doesn't want subscribers to pay for things they could easily do themselves. Most boards contain a "bootloader" (which is separate from *Linux*) that can easily run *Linux* stored on a flash device as part of the board. *LinuxLink* users configure their kernel by picking the right options from an easy to use configuration utility and frequently don't write or modify a single line of code. Should things be more complex, TimeSys is ready to provide support.

Q. **How much does *LinuxLink* cost? What is the model by which it is purchased?**

A. People interested in purchasing LinuxLink subscriptions can contact sales@timesys.com for pricing information.

Q. **Who has ownership of the *Linux* that is created via *LinuxLink*? If a developer "customizes" his own *Linux* - is he required to give this back to the community via GPL?**

A. Ownership isn't quite the right concept. The GPL states that GPL licensed code must be made available to customers, as part of the product delivery or available for download. Users who use GPL code to create something new still maintain ownership of their work; however, the GPL license means they must supply the source to their customers.

Much of the GPL confusion stems from a misunderstanding of when something becomes GPL. For instance, just running code on a *Linux* kernel does not make it GPL, nor does

packaging code with a *Linux* system make it GPL. However, if you take code and combine it with yours to make something new, the GPL covers that combined work. Furthermore, many software components typically used in a "combination" scenario are licensed under a variant of the GPL (the so-called LGPL or Lesser GPL) that means if you depend on an LGPL page to supply certain functionality, that dependence isn't combination and your code can remain proprietary.

Customers who create their own code can license it as they see fit. When you create a work "from scratch", even if you used GPL tools in the process, that code is yours and how you choose to license it is up to you. If the code depends on certain projects to run, for example a C library, those projects are licensed as LGPL. This makes sense, because when a person created something with GPL and that's used to create something else, that original author should retain control over how that code is used.

Q. **What about longevity? Many embedded products last for years, and the deployed *Linux* may be increasingly antiquated vs. kernel.org *Linux*. How does TimeSys help with product longevity and support issues?**

A. TimeSys knows that most customers want to start with the latest kernel, but then hesitate to move to a newer version in the middle of their project. TimeSys, by keeping current with changes to the *Linux* kernel and projects, lets users decide when to update. Again, most users are very conservative when it comes to updating the kernel or user space packages, and others are more open to making changes. TimeSys' position is that we supply the latest versions (while maintaining the older versions as well!) and let the user decide when to update.

Q. ***Linux* licensing is complex because the GPL is such a different animal than standard commercial licensing. Because there have been concerns about *Linux* licensing both in terms of commercial products illegally brought "in" to *Linux* and in terms of upwards "GPL contamination" of *Linux*-based products, how does TimeSys help its customers with *Linux* licensing issues?**

A. Licensing is a complex issue subject to differing interpretations based on the locality of the user and vendor. This is true for any type of license: GPL or otherwise. TimeSys does not pretend that a quick one-size-fits-all answer exists and encourages customer to work with their legal council and technical staff to find the best answer.

Q. **Thank you for your interview**

WIND RIVER SYSTEMS: LINUX FOR EMBEDDED AND REAL-TIME SYSTEMS

25 October 2007: Linux for Embedded and Real-time Systems

INTERVIEWEE. JASON PETTIT
PRODUCT LINE MANAGER, LINUX PLATFORMS
TEL. 510-749-2709
EMAIL. jason.pettit@windriver.com
COMPANY. WIND RIVER SYSTEMS
WEB. http://www.windriver.com/

Q. **First of all, tell us a little bit about yourself and your position at Wind River.**

A. I'm the Product Line Manager for *Linux* Platforms at Wind River. I'm responsible for our embedded *Linux* distribution and Real-time offerings. Prior to joining Wind River, I worked at a variety of high-performance computing companies and have been involved with *Linux* technology since 1998.

Q. **Wind River is known as a market leader in many ways in the embedded and real-time systems, but we would like to focus this brief interview on your "embedded *Linux*" offerings. Can you tell us in a nutshell what is Wind River's "*Linux* Strategy?" What is your value proposition for embedded and real-time designers?**

A. We made the decision to enter the *Linux* market several years ago due to overwhelming customer demand and because it was a natural extension of Wind River's product line. Our strategy is to provide embedded device developers with a commercial-grade *Linux* distribution, optimized for specific embedded markets and to address the wide variety of architectures present in the embedded market better than any other provider. In addition, our comprehensive tools strategy and experience in *VxWorks* allowed us to provide the same complete end-to-end solution to Linux developers as we have delivered to *VxWorks* users over the years.

Our value proposition is based on addressing the needs of a variety of markets including networking, automotive, mobile handset, digital video, consumer devices, and aerospace and defense to name a few. We enhance *Linux* development by offering development, debug, and analysis tools with our Eclipse-based Wind River Workbench suite. Finally, we cover the entire embedded lifecycle from board bring-up with Wind River On-chip debugging and in-field diagnostics with Wind River Device Management.

To summarize our value proposition is providing Linux, tools, support, and services to embedded developers.

Q. **Let's start with some very basic questions. The kind a Dilbert-like manager might ask of an engineer who is reading this Guide, and investigating *Linux* as a possible RTOS for his new embedded project. First of all, why should someone pay Wind River for *Linux*, when *Linux* is "free?"**

A. That's probably one of the biggest misconceptions in open source. Yes, the source code for the *Linux* kernel is free. What is *Linux*? Is it just the kernel? *Linux* is complex. A minimal distribution is made up of 100's of packages, patches, and extensions that need to be integrated and tested together. That integration effort is not trivial and embedded engineers need to understand how much additional time and resources this integration will add to their total project development. How much time will be spent creating a *Linux* distribution versus spending valuable resources on device differentiation? And whom do you turn to when you get stuck?

Wind River Linux reduces project risk, increases developer productivity, is fully supported, and allows developers to focus on creating unique features instead of integrating *Linux* packages.

In the end Dilbert's pointy haired boss and Dilbert share a common need - they both want their company to be successful. I believe the combination of *Wind River Linux Platforms*, support, and services increase the probability of success.

Q. **What precisely does the developer "get" from Wind River? A hardened *Linux* distribution? A true real-time *Linux*? Product support during development? Longevity? What are the bullet points vis-à-vis *Linux* that a developer gets by going with Wind River?**

A. The answer to all of your questions is a resounding YES!

A few proof-points are:
- Wind River will soon release our 6th update to *Wind River Linux*, our *Linux* targets embedded deployments and is commercially tested and supported.

- Wind River offers real-time choice. *Wind River Linux* includes PREEMPT_RT extensions and real-time safe drivers on validated hardware. We also offer *Wind River Real-Time Core* as an add-on to *Wind River Linux*, which provides hard real-time interrupt latency performance.

- Wind River support is a global organization, providing 24/7 customer support.

- Wind River has a well-defined standard product lifecycle with extended support options to provide long-term support.

- Wind River Professional Services can help development teams get up and running quickly or implement new technology needed for a particular application. Our services organization is very flexible and has deep *Linux* expertise.

- Wind River offers a complete set of tools for *Linux* development including:
  * Command-line tools
  * Wind River's Eclipse-based Workbench IDE
  * Wind River On-chip Debugging, a set of advanced JTAG tools
  * Wind River Device Management

Wind River has established the deepest and broadest partner ecosystem in the embedded device industry. Our partnerships with world-class technology companies enhance the value of our products through their seamless fit with the standards-based solutions of other industry leaders.

Q. **Wind River is known in the industry for your robust tools. What sorts of development tools do you offer that might be superior to the "free" or "open source" tools available for *Linux*?**

A. Wind River provides a platform developer seat of *Wind River Workbench* as part of our *Linux* platform offerings. *Workbench* is an Eclipse-based collection of tools that assists developers who are building devices with *Wind River Linux*. *Workbench* offers the only end-to-end, open standards based collection of tools for device software design, development, debugging, test, and management. *Workbench* enables organizations to standardize on a common environment for device software development, enabling developers to be more efficient. Among the many benefits, *Workbench* offers:

- Best-in-class capability at each phase of the development process, including hardware bring-up, firmware development, application software development, advanced visualization, system diagnostics, and test.
- Broad availability to support increased standardization across projects
- Multiple-target OS support, including support for *VxWorks* and *Wind River Linux*
- Target processor support for *ARM*, *ColdFire*, *IA/Pentium*, *MIPS*, *PowerPC*, Renesas *SuperH*, and *XScale* processors
- Plug-in architecture that enables additional target OS, target processor, and target connection support to be added
- An extensible framework, based on Eclipse, to seamlessly integrate third-party and in-house plug-ins for total customization and scalability

Workbench addresses the challenges developers and project teams face by increasing productivity, enabling collaboration between hardware and software developers, and meeting diverse development needs across an enterprise.

Q. **We see flashy ads on your website for "On-chip Debugging" via *Wind River Workbench for Linux*. What is this? Why is this so great?**

A. *Workbench On-Chip Debugging (OCD)* provides standards-based JTAG tools that revolutionize the way companies integrate their hardware, firmware, and software development processes. *OCD* provides a full suite of products designed for today's complex 32-/64-bit architectures, enabling customers to quickly bring up and diagnose hardware, debug firmware, and expedite the development of OS and middleware on their target hardware platform. Based on industry standards (Eclipse) and patent-pending hardware diagnostic capabilities, *Wind River on-chip debugging* combines hardware debugging with an open, end-to-end development suite to reduce costs and complexity.

Q. **What is Wind River *Real-time Core for Linux*?**

A. *Wind River Real-Time Core* is as extension to *Wind River Linux* that enables guaranteed real-time response for applications ranging from single-core feature phones and high-bandwidth IP communications to robotics and industrial control.

Real-Time Core is implemented as a real-time executive that coexists with the *Wind River Linux* kernel. *Real-Time Core* executes *Linux* as the lowest priority task and when no *Real-Time Core* applications are executing *Linux* operates normally. *Real-Time Core* applications are written in C using standard POSIX system calls, the *Real-Time Core* scheduler preempts *Linux* and provides traditional RTOS response times to *Real-Time Core* applications when executed.

In our experience, *Real-Time Core* provides lower worst-case latency and more predictable response times than PREEMPT_RT, allowing developers who might otherwise be concerned that *Linux* is unable to meet their real-time requirements to choose *Linux*.

Q. **What about services. We have heard that often times the real "sale" involved in a hardened *Linux* is a sale of a stream of services. What are the services offered by *Wind River for Linux*?**

A. As part of our comprehensive solutions, Wind River offers a *Linux Services Practice*, with focused offerings that help our customers meet strict market deadlines while keeping development costs down. Our experienced team delivers device software expertise that solves key development challenges and directly contributes to our customers' success. Backed by our commercial-grade project methodology, *Wind River Professional Services* include device design, *Linux BSP* and driver optimization,

software system and middleware integration, and legacy application and infrastructure migration.

Q. **The embedded and real-time Linux community is growing, no doubt about it. How does Wind River distinguish itself from other *Linux* competitors such as MontaVista or TimeSys as a provider of *Linux* for real-time / embedded?**

A. In a nutshell, the competitive advantages of Wind River can be summarized in a few key points:

- Stability – both from a corporate and technical standpoint. Having been in business for over 25 years, Wind River offers trusted financial longevity. And Wind River's *Linux* platform releases are quality tested and validated to ensure that our customers develop devices on a stable *Linux* platform.
- Complete toolset – from hardware bring-up to development, debugging, testing, and analysis; no other vendor offers such a complete end-to-end tools offering.
- Broad Hardware support – with a wide offering of Board Support Packages (BSPs) across all major architectures.
- The *Wind River Linux build system* – which allows developers to organize, store, and manage different parts of the development system and thus more easily understand what parts of the build may be responsible for performance issues, bugs, or defects.
- Unique value-added capabilities, such as *Real-Time Core* and Advanced Networking Technologies.
- Award winning Technical Support – to ensure that our customers have a vendor that will assist their development efforts and address issues that may arise.

Q. ***Linux* licensing is complex because the GPL is such a different animal than standard commercial licensing. Because there have been concerns about *Linux* licensing both in terms of commercial products illegally brought "in" to Linux and in terms of upwards "GPL contamination" of *Linux*-based products, how does Wind River help its customers with *Linux* licensing issues?**

A. A couple of things here, first the *Linux* community is very good at catching intellectual property violations and has continued to respond quickly to correct any code that is claimed to infringe on proprietary code. Second, for properly written user-space applications, the threat of GPL contamination is low. That said, open-source licensing issues can be complex and Wind River helps our customers in a number of ways.

Wind River has developed an intellectual property review process to ensure that our distribution does not improperly included proprietary code, and Wind River identifies the prevailing license on every package included in *Wind River Linux*. Going back to the "Dilbert-like manager" question asking why should someone pay for "free" software? The assurance of knowing that you are purchasing a product that has undergone careful scrutiny is a huge value to our customers' in peace-of-mind, time, and legal review savings.

Q. **Wind River has recently been promoting *Linux for mobile devices*. What vertical markets do you see the most potential for *Linux* expansion and why?**

A. Mobile: The mobile market has a huge opportunity for *Linux* expansion. *Linux* is increasingly being used on phones and is an attractive, cheaper, faster alternative to in-house software creation.

Automotive: The automotive industry is increasingly using *Linux* for in-car devices. Again, using *Linux* is much cheaper and faster than making software.

Network Equipment: Network equipment also has an opportunity for *Linux* expansion. *Linux* is increasingly being used and offers a cheaper, faster alternative to in-house software creation.

Q. Thank you for this interview.

Insiders' Guide Embedded RTOS: Interviews: Virtualization, Etc.

Interviews: Virtualization, Multicore, Hypervisors

Many of the real experts on RTOSes are the vendors themselves. We interviewed many leading vendors and tried to tie each to an important "theme" to help those making an RTOS choice. Here are interviews relating to virtualization, multi-core, and/or hypervisors.

- Green Hills Software: Advanced RTOS Issues, MultiCore & Virtualization
- Open Kernel Labs: OKL4 and Microkernel Technology
- RadiSys: OS-9 and Embedded Virtualization
- Real-Time Systems GmbH: Hypervisors for Real-time Systems
- TenAsys®: Virtualization and Real-Time for Windows
- Trango Virtual Processors: Secure Virtualization
- VirtualLogix: Virtualization for the "Connected World"

GREEN HILLS SOFTWARE: ADVANCED RTOS ISSUES, MULTICORE & VIRTUALIZATION

29 October 2007: Advanced RTOS Issues, Multicore & Virtualization

INTERVIEWEE. DAVID KLEIDERMACHER
VP ENGINEERING & CHIEF TECHNOLOGY OFFICER
TEL. 805.965.6044
EMAIL. bfrench@ghs.com
COMPANY. GREEN HILLS SOFTWARE
WEB. http://www.ghs.com/

Q. **First of all, tell us a little bit about yourself and your responsibilities at Green Hills Software**

A. I have been at Green Hills Software for 16 years, currently as VP Engineering and CTO. I have a split internal and external focus. Internally, I'm responsible for the engineering department and its product development and direction; externally, I interface with sales and customers to both communicate our technology vision as well as incorporate the proceeds from those interfaces into our technology strategy.

Q. **Green Hills is one of the oldest and most prestigious vendors of RTOSes and embedded tools. Can you give us just a quick overview to your products?**

A. Our corporate mission is to enable software developers to achieve total reliability, absolute security, maximum performance, the lowest manufacturing and development costs, and the fastest time to market for their electronic products. To this end, we have a broad product line, including integrated development environments (IDE), operating systems, compilers, hardware debugging devices, virtualization technologies, static analysis tools, and more. Our product brands include the *MULTI IDE*, *INTEGRITY* real-time operating system; *Green Hills Compilers*, *Green Hills Probe*, *DoubleCheck static analyzer*, and *INTEGRITY PC* secure operating system and virtualization technology.

Q. **When we discussed setting this interview up, Green Hills indicated that you have been doing a lot of work lately on "virtualization" and "multicore." Can you define what you mean by each?**

A. I recently did a presentation at Embedded Systems Conference entitled "Virtualization Demystified", attempting to categorize, explain, and justify the various types of computer virtualization technologies. Unfortunately, the term "virtualization" is often misused and misunderstood. One important type of virtualization technology is what I refer to as "virtual prototypes", which refers to a high speed software simulation environment that improves engineering productivity. A second important type of virtualization technology is run-time virtual machine software that enables one or more "guest" operating systems (and their applications), such as *Linux* and *Windows*, to run on a single computer.

Multicore is more straightforward, referring to a single-chip computer that has more than one processing core. Multicore architectures can be further classified in two dimensions: homogeneous vs. heterogeneous, and tightly vs. loosely coupled. Developing an electronic product based on a multicore processor brings with it numerous challenges not typically found in unicore designs. Multicore accounts for 10% (and steadily growing) of embedded systems designs; the need to educate and train software developers on how to deal with these challenges is becoming increasingly urgent.

Q. **How does "multicore" relate to "virtualization?" And vice-versa? Are these the same thing? Opposite sides of the same coin? Or interrelated software and hardware technologies?**

A. Both technologies are being driven, in part, by a desire to improve performance efficiency. However, there are many other independent factors influencing the move to each technology. Lots of people care about multicore but have no current interest in virtualization. And vice-versa. It is important to note, however, the synergy that makes the combination of these technologies more compelling than each taken alone: performance improvements made possible by multicore increase the usability of virtualization (which incurs some performance penalty).

Q. **What products or product initiatives does Green Hills have vis-a-vis virtualization and/or multi-core?**

Green Hills has been a provider of "virtual prototypes" for a very long time, but most recently we released a new version that uses dynamic binary translation to dramatically speed up simulation. With this technology, engineers can run full system simulations on a modern PC and approach (or in some cases exceed) the speed of the target embedded computer itself.

On the run-time virtualization front, Green Hills' "*Padded Cell*" technology enables guest operating systems to run in securely partitioned cells on top of the *INTEGRITY* real-time operating system. *Padded Cell* is a key component of our *INTEGRITY PC* product which is aimed at mobile, desktop, and server computing environments that have a requirement for general purpose "guest" execution alongside security and/or real-time critical components. A key aspect of this is *INTEGRITY PC's* ability to partition the guest environments away from the critical components in a provably secure manner. There are many fascinating usage scenarios that our customers see for this type of technology, including improving the size, weight, and power of military and intelligence IT computers and networks, combining head-unit and rear-seat office applications in an automobile, improving the security of *SCADA* and other critical computer infrastructure, improved security for medical and financial information systems, etc. Our pedigree for achieving the highest levels of safety and security in critical embedded systems is opening doors in an IT world eager for a change to the fail-first, patch-later status quo.

There are three major areas of concern for multicore developers, and Green Hills uses a holistic approach of products and services to address all of them: how to architect and develop multicore software; how to debug and tune multicore software; and how to properly manage the run-time execution of multicore software. The key challenge for multicore developers is the need to learn new techniques. The multicore developers' toolbox includes chip-level synchronized run control, application-level debugging for heavily threaded applications, system-level event analyzers, multicore trace collection and analysis tools, static and run-time defect detection, and operating system features such as fault-tolerant IPC, SMP, and POSIX multiprocessing APIs. Many of these tools are often foreign to software developers working with multicore systems for the first time and hence require serious activation energy. Green Hills has had a comprehensive focus on multicore software development productivity since multicore began to come into vogue at the beginning of this millennium. Instead of throwing a new toolbox over the fence, Green Hills multicore experts, via technical support, training, and consulting, help software developers accelerate up and over the learning curve.

Q. **Green Hills is known for your focus on high-end military and safety critical applications. Do you see these as areas for the first deployments in embedded systems of virtualization and/or multicore technologies?**

A. Green Hills customers run the gamut of industries, including consumer, automotive, telecom, medical, industrial, financial, and military/aerospace. Multicore is a horizontal technology, achieving adoption across all of these industries.

Virtual prototypes have the most profound impact on projects using custom hardware with long manufacturing lead times. Yet most any significant development project can benefit from the increased convenience and productivity enabled by simulation.

Run-time virtualization has received tremendous press in the IT infrastructure world, particularly as a method for improved server utilization. Our focus on secure virtualization is aimed at industries in which the value of information and assets managed by computer systems is high – such as military/intelligence, medical, homeland security, and financial – and this is certainly where much of our customer uptake has been. However, we're seeing compelling new usage scenarios across a wide range of industries. I mentioned the automotive example earlier. Another one in telecom involves collapsing control and data plane applications onto a single blade, running critical real-time applications alongside management applications on a virtualized Linux. As in the server world, virtualization has the potential to dramatically reduce size, weight, power, and cost for telecom infrastructure. And some of these telecom applications need a virtualization system that can provide guaranteed availability and real-time response of the critical applications regardless of what the guest environments are doing. That is where we excel over traditional IT virtualization solutions.

Q. **Many hardware people who are bringing us "multicore" say that it will take really efficient software to take advantage of multicore in order to realize its technology promise. Do you agree with that?**

A. Some have argued that we need better tools to increase application concurrency. I don't feel this is a major issue in mainstream computing applications. As software has increased in complexity, the coarse-grained parallelism enabled by threads and processes is found bountifully in typical embedded and even desktop applications. These threads map well to homogeneous multicores, enabling performance gains without requiring significant retooling or rewriting of code. Heterogeneous multicores typically require custom APIs regardless. Of course there are niches where parallelization tools and new languages and language extensions for concurrency make sense. The bigger requirement for efficiency, in my opinion, is to employ the proper debug and analysis tools to ensure optimum utilization. For example, performance analysis may show bottlenecks that can be broken by using core affinity assignments in an SMP system.

Q. **There are other "virtualization" companies such as VMWare or in the embedded space VirtualLogix or QNX Software. What does Green Hills bring to the party that is unique in terms of "virtualization?"**

A. I covered this earlier, but to reiterate – our unique capability has two major components: first, using the safety and security-certified *INTEGRITY* kernel, we can provide the highest levels of assured separation between virtualized and native/critical environments. Secondly, as *INTEGRITY* itself is a full-featured operating system with 10 years of successful application software developed and deployed, the *INTEGRITY PC* solution enables developers to develop and deploy real-time and secure applications that can fulfill critical functions that cannot be trusted to the guest portions of the system. This hybrid approach has many applications over and above the typical "hypervisor" solutions out there.

Q. **What are some other areas for improvement in the multicore ecosystem?**

A. Another area where there is much room for improvement is in multicore-related standards. We need better standards in the APIs between loosely coupled and heterogeneous processing elements. We need better hardware debugging interface standards. We need to improve the POSIX operating system standards to include the notions of core affinity and application-level (not just thread-level) scheduling. For many of these standards, we need big iron (i.e. IBM, Freescale, Intel, TI, ARM) to cooperate in order to achieve widespread adoption.

Q. **Thank you for this interview.**

OPEN KERNEL LABS: OKL4 AND MICROKERNEL TECHNOLOGY

23 October 2007: OKL4 and Microkernel Technology

INTERVIEWEE. ROB MCCAMMON
VICE PRESIDENT, PRODUCT MANAGEMENT
TEL. 312-924-1023
EMAIL. robm@ok-labs.com
COMPANY. OPEN KERNEL LABS
WEB. **http://www.ok-labs.com/**

Q. **First of all, tell us a little bit about yourself and Open Kernel Labs.**

A. I have spent my entire 23-year career working in the embedded systems arena. The first several years of my career were spent developing embedded systems and the most recent 10 years have been spent managing commercial products that make it easier to develop embedded systems. Most recently I spent 7 years at Wind River prior to joining OK Labs. At OK Labs I lead the development and implementation of a product strategy that results in the delivery of unique capabilities, which help embedded system developers build trustworthy systems in the face of ever increasing complexity.

OK Labs product offering has its origin in the foresight of our CTO Dr. Gernot Heiser, who saw the need several years ago for new techniques and innovative technology to overcome the challenges to reliability and security posed by increasing software complexity.

Several years ago Dr. Heiser initiated a program of academic research and microkernel technology development focused on creating new and better methods for the development of embedded system software. OK Labs products represent the incorporation of results from that R&D program in a commercially supplied system software platform for trustworthy embedded systems. The *OKL4* system software platform combines the capabilities of a hypervisor and a lightweight embedded operating system kernel in a single, micro-kernel based software solution that offers both trustworthy virtualization and secure decomposition of embedded system software.

Along the way Dr. Heiser also assembled a large and highly capable team of expert kernel hackers who now make up the OK Labs engineering staff.

As an innovator, we are committed to providing an evolutionary path to dramatic improvement in embedded systems software. We are enabled by our uniquely structured collaboration with leading embedded systems software research organization, National ICT Australia, to commercialize research results and rapidly bring to market the new capabilities required to enable the transition to this improved architecture.

Q. **Open Kernel Labs is not an RTOS company, yet your flagship product, *OKL4*, is closely related to RTOSes, and especially to the "emerging needs" of embedded software. How so?**

A. Technically *OKL4*, which is constructed using cutting edge microkernel technology, provides an execution environment for embedded systems software applications. As a result, *OKL4* can correctly be characterized as an operating system. However, the specific characteristics of *OKL4* make it most valuable when used to provide trustworthy virtualization and secure decomposition alongside one or more additional embedded or application operating systems.

The primary challenge facing embedded systems developers today is improving their ability to deliver increasingly complex systems, in less time, and at lower cost, while

satisfying the reliability and security requirements of their market, application, and product. In this context a better description of *OKL4* is a system software platform for trustworthy embedded systems that combines the capabilities of a hypervisor and a lightweight embedded operating system in a single micro-kernel based software solution.

OKL4 provides developers with trustworthy virtualization and secure decomposition capabilities. These capabilities are often used in a way that is complimentary to other operating systems. For example, the right combination of *Linux* or an RTOS running within an *OKL4* provided virtual machine, with select software components running in lightweight *OKL4* components, results in a more reliable and secure embedded system.

Q. **How is *OKL4* related to "virtualization?" How do you define "virtualization" and what are its benefits?**

A. I define system virtualization: as technology that supports execution of computer program code, from applications to entire operating systems, in a software-controlled environment. Such a Virtual Machine (VM) environment abstracts available system resources (memory, storage, CPU core(s), I/O, etc.) and presents them in a regular fashion, such that "guest" software cannot distinguish VM-based execution from that which runs on bare physical hardware. The bottom line is that virtualization allows multiple operating system instances and other execution environments to run on a shared processor.

There are several benefits for virtualization in embedded systems, which are best, demonstrated through use cases. Common use cases include; combining different and complimentary environments on the same processor (*Linux* and an RTOS for example); consolidating applications from a multiple processor environment on a single processor to reduce cost; and running legacy applications on a legacy OS alongside new applications developed for a new OS.

One of the primary uses of *OKL4* is as a platform for trustworthy virtualization. Several characteristics of *OKL4* contribute to its suitability for use in this way. First, when using *OKL4* as a hypervisor all guest operating systems are de-privileged and run entirely at user level. Second, because of its microkernel architecture the amount of *OKL4* software running in privileged mode has been minimized. Third, all device drivers also run at user level further reducing the amount of SW in privileged mode. Fourth, the source code for *OKL4* is freely available for inspection in order to make it a more trustworthy component of the system. Finally, every virtual machine exists within a separate protected domain. Many other virtualization solutions do not have these characteristics and thus do not provide trustworthy virtualization.

In addition, since *OKL4* is microkernel based system software, and not just a hypervisor, it allows the user to do things that a simple hypervisor does not. Specifically, *OKL4* provides a complimentary lightweight execution environment that offers a more trustworthy option for critical components than a full featured OS because it has been optimized to offer a much smaller trusted computing base.

Q. **Let us get this straight. *OKL4* allows for different OSes to run on the same processor, correct? Is this the same, different, or just related to the whole buzz about "multicore?"**

A. Yes, *OKL4* does provide a trustworthy way for different OSes to run on the same processor. You can think of *OKL4* as taking control of the physical processor and its resources and providing a number of virtual processors to higher-level software including operating systems. Each of these virtual processors runs entirely at user level and in separate protected domain. This virtualization applies whether the underlying processor is a single core implementation or a multicore implementation. Virtualization, when applied to a multicore processor, allows the number of virtual processors used to implement the software system to be decoupled from the number of physical cores in the actual processor. At the same time, the design principles applied to partition an

application across multiple processors apply whether those processors are physical cores or virtual processors managed by a hypervisor.

Virtualization using *OKL4* and multicore both support a multiprocessing system architecture. Virtualization provides multiple virtual processors using a single physical processor. Multicore provides multiple physical processors in a single physical package. The benefits of using virtualization, rather than simple physical multiprocessing are reduced hardware cost and increased design flexibility. Virtualization and multicore are not mutually exclusive since the benefits of virtualization apply to its use on both single and multicore processors.

Q. **Other products allow for one RTOS to run on one processor core, and another (say, for example, Linux) to run on another. They thus use "multicore" to achieve greater system stability (via the RTOS) plus enhanced features (via the *GPOS*, or *Linux*). How is your solution different or better vs. that sort of a solution?**

A. When you dedicate a physical processor core to an RTOS and another processor core to *Linux* the processing resources available to each OS are static and fixed by the physical resources of each processor. If, for example you use a dual core processor with identical cores then the MIPS available to the RTOS environment and the *Linux* environment are the same. If you need 2/3 of the available MIPS for the RTOS and 1/3 for *Linux* it is not possible to partition the physical resource in that way. If you use *OKL4* to virtualize the processor instead, you can flexibly partition the physically available resources amongst the virtual processors. Trustworthy virtualization offers the same benefits of incorporating an RTOS and Linux into the system but with more flexibility. Also, since *OKL4* runs the RTOS and Linux at user level the amount of privileged mode software in the system is reduced making it much easier to insure the overall reliability and trustworthiness of that software .

Q. **You mention "trusted computing" or "security" on your website. What does *OKL4* do for embedded security? How does it do it?**

A. One of the benefits of *OKL4* is that it provides lightweight security components that support applications with a minimal trusted computing base.

The trusted computing base (TCB) of a computer system is the set of all hardware, firmware, and software components that are critical to its security. Bugs occurring inside the TCB might jeopardize the security properties of the entire system.

Reducing the size of the TCB, which reduces the likelihood of bugs within the TCB, improves the overall security of a computer system. All code that runs in the privileged mode of the underlying processor is part of the TCB; however, the TCB almost always includes other software as well. For example, in a Linux system, any daemon running as root would be part of the TCB.

The TCB is always relative to a particular program because different programs depend on different things. The TCB of a system generally refers to the total combination of TCBs from all of its programs. Developers looking to provide more secure environments for particular programs should strive to place those programs in an environment with a small TCB.

OKL4 offers exactly that capability. It allows such programs to exist in a small TCB execution environment alongside a full operating system environment with a much larger TCB.

Q. **Which chip architectures do you support? *ARM*? *x86*? *MIPS*? Any others? Are you closely related to certain hardware architectures? Why or why not?**

A. *OKL4* does not require any specific hardware features and is not limited to support of any specific hardware architecture as a result. *OKL4* supports *ARM*, *MIPS*, and *x86* architectures today. Support for securely isolated and protected domains does require

the processor to have an MMU. The implementation of *OKL4* for a particular hardware architecture is optimized to utilize the features of that architecture to boost performance.

Q. **What application areas or verticals do you feel are particularly suited for this approach? Wireless? Consumer? Others? Why?**

A. The benefits to the user of *OKL4* are ultimately in the form of increased reliability, increased security, and reduced development effort for complex systems. These benefits apply to the development of embedded systems in just about any vertical.

To date the trustworthy virtualization and the secure componentization capabilities of *OKL4* have been of particular interest to developers in Mobile Handheld, Consumer Electronics, Network Equipment, Point of Sale, and Industrial/Medical applications.

Another way to describe systems well suited for the *OKL4* approach is to look for one or more of the following characteristics: handling of digital media with restricted rights, handling of sensitive personal information, payment applications, devices with safety requirements, systems combining more and less trusted subsystems, and any system where the increasing complexity of the software is threatening its reliability or straining the teams ability to move the product forward in a rapid and agile manner.

Q. ***OKL4* is "open source." Why is this important to designers who just want to get their design done, and don't have a political agenda vis-a-vis open source / good or bad?**

A. *OKL4* is dual-licensed and available under either an open source license or a proprietary use license. The *OKL4* download that is made freely available at ok-labs.com is provided under the open source license. This makes it easy to experiment with *OKL4* and to use it in evaluation, research, academic, and open source development contexts. The open source distribution of *OKL4* also contributes significantly to its trustworthiness as a software component that falls within the trusted computing base by making the *OKL4* source code freely available for inspection and review.

The proprietary use license available from Open Kernel Labs removes the requirement of the open source license that the user publish source code for their software. This form of license will be desirable for the majority of commercial product deployments involving *OKL4*.

Q. **How much does it cost? Can you explain your business model - what do developers purchase, and how? What are the various models?**

A. How much it costs to use *OKL4* depends on the nature and scope of its use. The shared risk business model will be familiar to anyone who has had experience licensing operating systems and other target system runtime software components in the past.

The use of *OKL4* requires a development license under the proprietary use model in order to develop a commercial product. The development license entitles the licensee to use *OKL4* on a development project, the scope of which is defined during the licensing process. Once a product enters manufacturing a production license must also be purchased for each device produced that includes a copy of *OKL4*.

Q. **What else should we know about Open Kernel Labs and its products?**

A. One goal of the research and development program undertaken by Open Kernel Labs is to create a system software platform that provides an unparalleled level of trustworthiness. This unparalleled trustworthiness will be enabled through the commercialization of leading-edge research in performance analysis and formal verification of software. Proving the trustworthiness of *OKL4* at a never-before-achieved level for embedded system software will streamline the development of trustworthy systems built on top of the *OKL4* platform. For *OKL4*, Unparalleled Trustworthiness is the result of the following two outcomes:

Creation of a complete execution timing model for *OKL4*, providing comprehensive information on the worst-case execution latencies of kernel functions. Availability of comprehensive timing information for *OKL4* enables its use in latency and timing-sensitive applications with extremely high confidence in its behavior.

An implementation of the *OKL4* kernel whose design specification and implementation in C code have been formally proven correct using mathematical methods. The resulting proof of correctness provides assurance to the user that neither their development activity nor device performance will be impacted by defects in underlying platform software.

OK Labs hopes to deliver a version of *OKL4* that has been characterized and verified as described above within the next few years. If you would like to learn more about the research taking place in support of these goals, check out the Unparalleled Trustworthiness section, found on the Products page of the Open Kernel Labs web site, http://www.ok-labs.com

Q. **Thank you for this interview.**

9 November 2007: OS-9 and Embedded Virtualization

INTERVIEWEE. STEVE JOHNSON
PRODUCT MARKETING MANAGER, OS-9
TEL. **515-327-2305**
EMAIL. **STEVE.JOHNSON@RADISYS.COM**
COMPANY. RADISYS
WEB. **http://www.radisys.com/**

Q. **First of all, tell us a little bit about yourself and your responsibilities at RadiSys.**

A. I am the *OS-9* product-marketing manager for RadiSys and provide market support of *OS-9* for North America, Japan and Europe. I have supported *OS-9* in the embedded market for over 20 years. I began marketing *OS-9* VME board level industrial systems before joining Microware Systems Corporation / RadiSys. My work has followed *OS-9* processor support from single core real-time implementations to multi-core hybrid *OS-9/Linux systems.*

Q. **RadiSys is not especially known as an RTOS vendor, yet the company acquired the *OS-9* product line from Microware and has been quite successful in Japan with it. Can you outline what RadiSys offers in the RTOS space?**

A. *OS-9* is very successful in Europe as well. RadiSys acquired the *OS-9* real-time operating system with its purchase of Microware Systems Corporation in 2001. RadiSys invests in the *OS-9* technology providing feature releases and support to its growing worldwide customer base. *OS-9* is the original real-time plug-in operating system that provides an elegant, modular approach for high performance, compact embedded design.

We continue to maintain and advance *OS-9*'s feature set with the release of *OS-9 v4.8* for *PowerPC*, *ARM/StrongArm*, and multi-core support for *Intel Core Duo* processors. RadiSys also supports *OS-9* on *MIPS*, *SuperH* and *XScale*. RadiSys provides and supports *OS-9* on popular board level architectures including *VME*, *cPCI*, *PC/104*, *COM Express*, and *ATCA*.

Q. **It is rather unusual for a vendor known for its telecommunications boards and systems to also be an RTOS vendor. What is the business strategy in terms of having *OS-9* as part of RadiSys?**

A. *OS-9* is essentially its own business entity within RadiSys, servicing and growing the *OS-9* community. The RadiSys *OS-9* engineering and marketing group works from at the company's Des Moines facilities and is staffed with many of the original *OS-9* contributors from Microware. Our strategy is to expand *OS-9* usage targeting board level manufacturers, system integrators and end users. We provide *OS-9* as an independent software supplier to these customers.

Q. **RadiSys, like so many companies recently, has been focusing on "virtualization." Can you explain how you see virtualization and what are its benefits to embedded systems designs?**

A. Users of our *OS-9* virtualization implementation will get better system performance and reduced hardware costs compared to integrating separate single core systems. We enable real-time and general-purpose processes to run concurrently on multi-core processors with the *OS-9 RTOS*, *VirtualLogix VLX* virtual machine monitor and *Linux*. In this hybrid environment, *OS-9* is able to control time critical, real-world machinery, collect

data and provide it to Linux for non-real-time sensitive processing. For dedicated real-time environments, we are able to run two copies of *OS-9* with *VLX*, thereby increasing throughput in the multiprocessing architecture. This *OS-9* performance will scale with increasing numbers of CPU cores.

We view the availability of multi-core technology as free, unused, available cycles that can be tapped into. Running *OS-9* and *Linux* together with *VLX* on one board allows customers to eliminate the need to run two networked computers. This increases system reliability and execution efficiency while decreasing hardware cost and power. Virtualization also allows the vast portfolio of Linux applications to run in the hybrid real-time environment without any modification or porting.

Q. **What application verticals do you think will see the most immediate benefit from virtualization?**

A. Industrial automation, medical imaging and communication. We support factory automation systems and medical systems using *OS-9* for real-time control networked to Window systems for command center user interface and post processing. Customers have the ability to eliminate the second computer by running both operating systems on one multi-core platform. In addition to realizing cost savings using less hardware, a multi-core environment running a RTOS and a GPOS improves system reliability and allows an abundance of *Linux* and *Windows* applications to run in a real-time hybrid environment. Telecommunication applications can also leverage these efficiencies to reduce the number of compute and switch blades.

Q. **What does *OS-9* bring to virtualization that is "unique?"**

A. *OS-9* was the first real-time operating system to follow the *Unix/Linux* process model for real-time design back in 1980. The *OS-9* architecture and development tools are able to logically port and execute *Linux* applications in real-time. In a hybrid *OS-9* and *Linux* virtual environment, users have the choice to compile *Linux* code for needed real-time performance to run under *OS-9*. For hard real-time, target *OS-9*. For background operation, use *Linux*. *OS-9* with *VLX* virtualization gives you that choice.

Q. **Please explain RadiSys' partnership with VirtualLogix. How does this advance "virtualization" for RadiSys' customers?**

A. RadiSys utilizes VirtualLogix's *VLX* as its virtual machine monitor enabling *OS-9* and *Linux* to run concurrently on a multi-core processor while guaranteeing high throughput and real-time performance. *VLX* provides a multi-processing environment that allows *OS-9* to drop in and communicate to *Linux* through its virtual Ethernet. *VLX* partitions hardware to *OS-9* for real-time processing on one core and partitions the hardware to Linux for general processing on the other core. Customers can create new hybrid applications using virtual Ethernet or run legacy *OS-9* and *Linux* code on this platform without porting or do both.

Q. **How do you see "virtualization" relating to "multi-core?" Are these two constructs tightly related?**

A. They are very much related. Without virtualization you have a collection of cores running in separate environments. With virtualization you have a true multi-processing environment that can execute a hybrid-operating environment with resource and data sharing. Virtualization partitions the hardware so each core can be shared or dedicated to *OS-9* to maintain hard real-time performance. Developers have essentially free cycles at their disposal with multi-core processors. It will be through virtualization that a new computing paradigm will be realized.

Q. **Do you see virtualization as a way to unite *Linux* and hard RTOSes? *Microsoft Windows* and hard RTOSes?**

A. Yes to both questions. Virtualization through intelligent machine monitors like *VLX* can bring hard real-time capability to systems running *Linux, Windows* and other general-

purpose operating systems. This approach allows design teams to now run existing code in a familiar environment without porting. Design teams can concentrate on the specific segment of their application without worrying if the general-purpose operating system is fast enough. Add a proven, reliable RTOS such as *OS-9* to handle the real-time load and you can easily remove that question.

Q. ***Linux* vendors like MontaVista and Microsoft in the *Windows* world are working hard to bring "real-time" to their products. Isn't it redundant to offer a two OS / multicore solution rather than try to enhance the real-time capabilities of these "general purpose" OSes like Linux or Windows?**

A. We feel that having a common virtual machine monitor with a common real-time operating system for both *Linux* and *Windows* is more flexible and productive than redundant. Selected RTOSes can provide unique features to the general-purpose operating system where real-time implementations of *Linux* and *Windows* cannot. For example *OS-9* provides dynamic loading of device drivers while the system is up and running a feature that is not replicated on the general purpose platforms or its real-time variants.

Q. **Thank you for this interview.**

REAL-TIME SYSTEMS GMBH: HYPERVISORS FOR REAL-TIME SYSTEMS

23 October 2007: Hypervisors for Real-Time Systems

INTERVIEWEE. GERD LAMMERS, CEO
TEL. +49 (0)751 359 558 11
EMAIL. Gerd_eg3@real-time-systems.com
COMPANY. REAL-TIME SYSTEMS GMBH
WEB. http://www.real-time-systems.com/

Q. **First of all, tell us a little bit about yourself and Real-time Systems GmbH.**

A. My name is Gerd Lammers, founder and CEO of Real-Time Systems GmbH (RTS). I have worked in the embedded systems industry for almost 20 years.

In my last position before starting up RTS, I headed up the sales and marketing department for the real-time operating systems division at KUKA. After KUKA decided to close our local office in a move to centralize its operations, the idea for RTS was born.

Real-Time Systems GmbH now provides real-time hypervisor technology, time-synchronization solutions as well as other products and engineering services for the embedded market.

Q. **What is a "*hypervisor*?" Why is this beneficial to embedded systems that need real-time performance?**

A. In general, a *hypervisor* is a virtualization platform that allows multiple operating systems to run in parallel on a single host computer. Some solutions, known in the server or IT domain, include products such as *VMware*, *XEN*, or *VirtualBox*. Unfortunately, virtualization has traditionally implied that a virtualized system's hardware be emulated or even shared among operating systems, without direct access to PCI hardware devices and using time slices to distribute processing power to each OS. Running an RTOS in a traditional virtualization environment therefore results in a loss of determinism and hard real-time performance, which are, however, firm requirements for many embedded systems.

The *RTS Real-Time Hypervisor* distinguishes itself from such solutions by virtualizing only the hardware used by a General Purpose Operating System (GPOS), elements that play no role in real-time system performance.

The unique technology implemented in our *Real-Time Hypervisor* does not 'virtualize' Real-Time Operating Systems (RTOS). Instead, it assigns hardware components like processor cores, PCI devices and memory for the exclusive use of an OS, while all other system resources, 'owned' by other operating systems, remain hidden and separate. This allows an RTOS to directly access physical hardware, an important prerequisite for determinism and real-time performance, and for this reason, too, operating systems that run in our *hypervisor* environment require no special device drivers.

Each operating system instance executes completely independent of all others. The independence is so thorough that an OS can even reboot without affecting any other parallel-running OS. Our architecture permits multiple instances of the same operating system to be deployed in an AMP (asynchronous multiprocessing) mode but also enables a mix of operating systems, e.g. a GPOS alongside an RTOS, to run in parallel.

Embedded systems based on our technology generally enjoy the benefits of reduced system costs and increased reliability, this because of the requirement for less hardware. Apart from this, one could speak of an "integrated distributed system" with many of the advantages of such a design but without the interface and communication challenge

generally associated with distributed systems. In our *hypervisor* environment, operating systems easily communicate with one another via shared memory areas or a virtual network. Because the future of high-performance embedded computing is closely related to the further development of multicore processors, our real-time hypervisor technology is logically oriented in that direction as well.

Q. **What specific application areas are in most need of this sort of solution? Are these medical, military, industrial? What technical requirements push developers to find your sort of solution beneficial for their application?**

A. Because our *hypervisor* does not target specific operating systems, we are experiencing tremendous customer interest from a wide range of market segments. Our solution is a general one.

Often developers find our solution beneficial when they have an application in which a GPOS runs on an x86 platform and a controller or other real-time applications runs on a separate embedded hardware platform. In this case, migrating the application to a single multicore platform without having to change operating systems is a very attractive and logical step forward. Applications that fit this description range from industrial automation products like motion controllers to medical devices like patient monitors or X-Ray machines to simulators or complex defense applications.

On the other hand, there are developers who simply desire to move a single core, real-time OS application to a multicore AMP platform. Doing this with the help of our *Hypervisor*, enables them to split a complex application into several manageable, independent and securely separated modules, in which a code change or failure in one module does not affect the rest of the application, even while increasing overall system performance. This multiprocessing approach works even if the out-of-the-box OS only supports older single-core x86 platforms.

For new product development, the *hypervisor* approach gives the system architect the flexibility to design modular, secure, scalable and cost effective systems on modern hardware platforms.

Q. **Does your *RTS Real-time Hypervisor* bring "real-time" performance to Windows XP? Linux? How does this work?**

A. No, our *Real-Time Hypervisor* does not bring real-time features to a conventional non-deterministic operating system. We leave this market to the providers of real-time extensions.

The *Real-Time Hypervisor* is a different kind of solution. It provides a mechanism to "partition" or "segment" an x86 multicore platform into individual systems which are then used by each OS independently. I suppose, if one would view a system running an RTOS and a GPOS as a "black box", one might be tempted to think of it as a GPOS system with real-time performance; but this view applies just as well to a traditional system that uses multiple hardware platforms.

Q. **Both Microsoft (*Windows XP embedded*) and *Linux* (MontaVista or Wind River *Linux*) have "solutions" which claim to offer real-time performance. Why is using a hypervisor a "better" solution? What applications or issues would encourage a designer to use your approach rather than one of these alternative paths to "real-time" for a "general purpose" OS?**

A. In this case, there is no "better" or "worse" - it depends on the requirements.

Before *Real-Time Hypervisors* were available, the customer made a decision whether to use a GPOS, a real-time extension, a stand-alone RTOS or multiple systems with a GPOS and one or more RTOSes. While this has not changed, the *Real-Time Hypervisor* technology eliminates the need for additional hardware but not the need for specific operating systems.

Surely there will be times when a real-time extension is the perfect fit for a certain application, but if the customer has already million of lines of code written for a certain RTOS or if drivers or third-party software is available only for a specific operating system, then the use of a hypervisor is probably the better approach. Another appealing feature of the real-time hypervisor is also the capability to reboot the GUI independent of the real-time application and vice-versa.

Q. **What is the relationship between your solution and "multicore" for embedded systems? Does it take advantage of the new multicore architectures, and if so, how?**

A. When we started Real-Time Systems there were already solutions available that combined an RTOS with a GPOS on a single CPU. Unfortunately as these solutions required the operating systems to share a single processor between them, they either needed to multiplex processing time between operating systems or perform a complete OS context switch every time a real-time interrupt occurred. Both activities affect interrupt latency times and introduce jitter, which many applications cannot tolerate. Since prices for multicore processors have decreased drastically and silicon vendors' have shown roadmaps that stress multicore technology, we correspondingly decided to base our *hypervisor* solution on multicore as well.

Our *hypervisor* assigns individual cores directly and exclusively to individual operating systems. And, since the x86 multicore architecture provides a local interrupt controller for each processor core, PCI devices and legacy IRQs are directly routed to their assigned CPU in hardware. As a consequence, our hypervisor adds no latencies to any incoming interrupt and preserves an operating system's real-time performance.

Q. **Is it only available for x86 / Intel Architectures? Are there any plans to expand beyond x86?**

A. Outside of x86, there are many interesting applications that would benefit from a real-time *hypervisor* solution. These other products however, mainly in the mobile and consumer markets, are often based on the ARM architecture, for which other companies already offer solutions. Because of that and because our expertise is in the x86 architecture and real time, this will for now remain our main focus.

Q. **One of the promises of "multi-core" is better hardware resource utilization. Yet many of the solutions today assign one RTOS / OS to one processor, another to another, etc., making a situation in which one processor may be "idle" while the other is "overworked." Does your solution allow for better sharing of system resources?**

A. Experience says there are very few approaches without tradeoffs. If multiple operating systems share one CPU core, real-time performance is affected. If you dedicate hardware resources to individual OSes, one core might indeed not be utilized fully. But this is of little concern, since the primary purpose of a real-time hypervisor is to achieve safe, independent utilization of processor resources by multiple operating systems while making sure that real-time performance is not impaired. This also means that even if one core is "overworked", an RTOS on a separate core, having it own resources, can and will still run correctly.

Q. **You recently became a distributor of Ardence's real-time products in Europe. How are your products different from theirs, which also give real-time performance to Windows?**

A. That is true. We distribute a number of Ardence's products. One of their best-known products, *RTX*, is a real-time extension for Windows. Just like you said earlier, *RTX* brings real-time performance to Windows. Especially in Industrial Automation, Automotive and Medical there are applications that need real-time performance within the context of Microsoft Windows. It is as we discussed earlier: Often there is no need for a separate RTOS with a completely different tool-chain - especially if all aspects of an

application including real-time tasks can be supported within one development environment and one operating system.

So what is the answer to your question? The best solution is the one that best meets the system's requirements.

Q. **Thank you for this interview.**

TENASYS®: VIRTUALIZATION AND REAL-TIME FOR WINDOWS

31 October 2007: Virtualization and Real-time for Windows

INTERVIEWEE. KIM HARTMAN
VP SALES & MARKETING
TEL. 1.503.748.4725
EMAIL. kim.hartman@tenasys.com
COMPANY. TENASYS
WEB. http://www.tenasys.com/

Q. **First of all, tell us a little bit about yourself and your responsibilities at TenAsys.**

A. I am a degreed Computer Engineer from the University of Illinois, Urbana-Champaign and have ample design experience to assist me well as the lead of sales and marketing at TenAsys®. I was part of the marketing launch team of our *INtime*® operating system in 1997 and have enjoyed exciting times in growing the *INtime* product and the TenAsys company.

I have routine world wide involvement with OEM customers in raising awareness of what's available now, where virtualization technology from TenAsys is going, as well as taking requirements for needs in a number of focus industry segments. I have the privilege of meeting with CTO and chief architects of major organizations exposing them to new concepts, technologies and products that save them money while reducing their time to market.

Q. **TenAsys is known as a company that brings "real-time" to Microsoft Windows for embedded. Can you briefly outline your product portfolio?**

A. For the past ten years we've focused our attention to our *INtime* real-time operating system and tools for Windows platforms. We had already 17 years experience in developing RTOS software products for x86 based systems, but we were enthralled with the mixed general purpose (GPOS)/RTOS combination. It seemed so natural that most advanced real-time applications had some degree of general-purpose user interface, enterprise or third party app, which Windows excelled, at the same time requiring absolute control and determinism provided by a true RTOS. Serving both needs with *INtime* reduces hardware cost and complexity while unifying development under the Microsoft Visual Studio environment.

Today multiprocessor PCs have awoken the market to prompt interesting use models. Customers often come with the question "I have two processor cores, so can I have two operating systems?" The answer is unequivocally "YES", and in fact it's not limited to two cores, or just Windows and *INtime*. *INtime* remains an RTOS running stand alone, and now it runs asymmetric on multi-cores. This model extends across multi cores with Windows too. Imagine the possibilities of running a quad-core with up to four operating systems all dedicated to different tasks; one running vision recognition, another doing coordinated motion, a third running a tightly coupled supervisor control application, and still another running Windows GUI, possibly back office manufacturing software and any number of 3rd party applications. All on one PC platform!

Q. **Prior to the interview, you mentioned that you wanted to focus on virtualization and its impact on embedded and real-time systems. How do you see "virtualization?" What applications and benefits do you think will be the early beneficiaries of it?**

A. "Virtualization" is a hot buzzword; everyone's trying to get in on the action. The truth is that TenAsys has been creating virtualization software products for real-time and embedded use for 17 years. In fact *INtime* uses para-virtualization technology to share critical PC devices with Windows. This approach has been substantially refined over the past 10 years, and the Windows & *INtime* RTOS model is the best available anywhere.

Going beyond just Windows & *INtime*, TenAsys' experience combined with new silicon from Intel is allowing a greater number of customers to realize the cost reduction benefits without having to re-write their existing real-time applications. We're evolving new real-time hypervisor, or VMM models protecting customer's proven intellectual property. This technology leaves applications and operating systems in tact, and is deployed directly on dedicated IA cores. Any customer who has a mixed GPOS and RTOS solution needs to be asking serious questions about how to obtain VMM software technology that will help them cut hardware costs without causing a complete rewrite of their application.

Q. **What do you think is wrong with current ideas on virtualization?**

A. It's more about applying the right kind of virtualization technology in the proper place. The big play so far has been in server consolidation and for software developers. VMware went public this year and the stock soared to nearly $125 per share, a market cap of $12.5B. Clearly the use case here has been validated with profound returns. Why would anyone buy 10 systems, when one server running some virtualization software can do the same job?

The problem is that embedded and real-time operating systems and applications don't work on this virtualization model. Determinism and services to RTOS guest operating systems is not the problem being solved by these solutions. There's clearly need for a better approach in serving embedded and time critical processing needs.

Q. **How is the TenAsys solution different?**

A. We've had very specific goals for embedded and time critical application of our real-time hypervisor, or *VMM* technology. The emphasis is in minimizing the number of devices virtualized, drawing guest operating systems and their applications directly to native hardware.

We don't virtualize CPUs, graphics and other PC devices, which results in substantial driver stack up and interface abstractions masking the unique capabilities of the modern hardware they're running on. Instead, the desired model is to partition the hardware to the needs of the mixture of applications and their native (guest) operating system and drivers. This results in the most tightly coupled solution between each set of application, RTOS, driver, CPU and I/O hardware, delivering the highest performance possible.

Q. **The real promise of multi-core and virtualization is, of course, when applications can share resources in real-time and the OS (or some other software) can "police" the sharing of resources so as not to imperil critical real-time processes. Is this vision close to being a reality?**

A. Yes, and the work is done in the *VMM* where the real benefits from modern CPU silicon are in new services made available for TenAsys to extend our *VMM* technology. Virtualization support, such as Intel VT-d is starting to show up in devices making the sharing easier for us to implement. The policing you refer to is still necessary, and needs to be fast and efficient. But the fundamental approach to optimizing multi-core, and multi-OS utilization is in partitioning the hardware resources directly for the guest operating systems. The guest operating system only sees what devices it has access to.

INtime for Windows is an advanced example where the device configuration tool is used to graphically partition and assign devices to either *Windows* or *INtime OS*. The model is similar when applied to our real-time hypervisor technology where the partitioning is done prior to the GPOS boot resulting in less policing required. If a guest OS driver tries

to access a resource not already assigned, it explicitly calls the *VMM* to block and fail the attempt.

Q. **What about virtualization and the legacy application code out there. Doesn't this present a problem, unless the virtualization is handled at a more fundamental level as at the OS or below?**

A. We see legacy support as the most profound opportunity for virtualization in the embedded space. Customers business' are at stake. The constant pressure to drive cost down while adding features and enhancing performance leaves many thinking they have to redesign their product from the ground up. Going off line for two years engineering would seriously damage their business, if not destroy it completely.

TenAsys believes we're addressing an evolutionary software adoption made possible with revolutionary hardware. The real-time hypervisor technology we develop is revolutionary in concept, but evolutionary for us as the hardware functionality fills in. It is allowing TenAsys to take our lengthy experience with *Windows* and *INtime* and efficiently apply it to other GPOS and RTOS guests.

Q. **What applications are best served by *INtime* and your real-time hypervisor?**

A. Any PC hosted application where *Windows* is used and time critical processing is required can use *INtime* today. We support *Vista, XP, XP Embedded* and *W2k* operating systems with full debug support in *Visual Studio 2005, 2003* .net and *2008* ready for release.

TenAsys is also previewing developing technology allowing an existing deployed application, which uses two dedicated computers, *Windows* and other guest operating systems, to be deployed on a multi-core CPU with the same tightly coupled *INtime* functionality without having to port their real time application.

Using this technology, most any headless RTOS can be assigned its own CPU and run native drivers to the I/O they control. Additional 'virtual devices' are possible to service IPC for data sharing or signaling as needed. Look for this technology to be more widely accessible in the coming months.

Q. **Thank you for this interview.**

A. "You're very welcome. It's my privilege."

TRANGO VIRTUAL PROCESSORS: SECURE VIRTUALIZATION

31 October 2007: Secure Virtualization

INTERVIEWEE. FRANK ALTSCHULER, MARKETING MANAGER
TEL. (512) 947-7215
EMAIL. FRANK.ALTSCHULER@TRANGO-VP.COM
COMPANY. TRANGO VIRTUAL PROCESSORS
WEB. http://www.trango-vp.com/

Q. **First of all, tell us a little bit about yourself and your responsibilities at TRANGO.**

A. I'm new to TRANGO, having started less than two months ago. Previously I'd been at StarCore LLC, a DSP intellectual property firm, and before that, Cirrus Logic, a fabless semiconductor company doing mixed signal devices such as audio and video codecs.

My role at TRANGO is marketing with an emphasis on communications activities for North America. The company is based in Grenoble, France so a large part of what I do on a daily basis is to take the work that my French colleagues have done, and put a more Western spin on it.

Q. **TRANGO is not an RTOS company, but your products work with RTOSes to carry out "virtualization." Can you briefly explain your product offerings?**

A. Virtualization is a technology that allows developers to create multiple simultaneous execution environments. In practical terms, that means that software executing within each environment behaves as though it was the sole owner and proprietor of the SOC, when in fact execution is occurring on a virtual representation of the underlying hardware.

This creates a rich set of design possibilities. For example, this makes it possible to run multiple heterogeneous operating systems; such as an RTOS in one environment, a rich OS to drive the GUI and other user interfaces in another, and a security-oriented application such as DRM in a third. Previously this may have required two or three separate CPUs to meet the performance and security requirements of the system. Now, with virtualization, you can combine everything into one core.

We believe that we have created the thinnest, lightest, and most secure *hypervisor* in the embedded industry. At between 20 and 30 Kbytes of code it represents a substantially smaller footprint than competing solutions. Also, through our commitment to hard real-time design concepts, our developers have kept the performance overhead to 2% or less, making integration with an RTOS extremely straight-forward.

Q. **What do you see as the differences between the current industry buzz words, "virtualization" and "multi-core?" How are these related, and how are they different in your mind?**

A. Well, "virtualization" is the technique that lets you create multiple virtual processors on a single core, and "multi-core" is the architectural direction that SOC designers have charged down to get around the power limits that they were banging into by their previous focus on ever higher clock speeds.

They're basically complimentary concepts in that on a single-core system you can run an SMP OS by creating a pool of virtual processors to host the OS, and then when you move to a true multi-core SOC a lot of what you've created should just work. However, in a multi-core environment there are additional benefits to virtualization.

For example, when designers are setting up the software architecture for a multi-core system they typically assign cores to specific tasks such as power monitoring, system health, packet analysis or what have you. This can result in uneven loading if some tasks are less demanding than others. With virtualization you can create a number of virtual machines and spread them evenly across the available cores during peak processing demand. Then, as demand falls off, the virtual processors can be migrated to fewer and fewer cores, thereby allowing unused cores to be powered down. This produces more optimal resource utilization by essentially using "fractional cores", and more effective power management than can easily be achieved by other means.

Q. **What target markets or applications are most likely to benefit from your approach? Are these cell phones, consumer products, small form factors, military? What characteristics make an application most likely to benefit from TRANGO?**

A. This is the great thing about virtualization. For each vertical market that you approach, there's a different value proposition. It's an extremely general-purpose technology. It's almost like asking what markets embedded processors themselves are good for.

For example in Point of Sale (POS) terminals, there's typically been one dedicated device for the magnetic card reader/keypad, and another for the user-facing display/GUI. Virtualization allows those two execution environments to be combined into one, saving a complete CPU along with its attendant memory subsystem. That's a great savings on the bill of materials but it's also a lot easier to develop on a single platform than on two.

In cell phones there is a value to device consolidation but there is an additional benefit to having a highly modular approach to system integration. Once the handset maker has the base phone functionality working, integrating new features while not disturbing the existing validated code base can be a huge savings in both time and effort.

In high reliability applications such as telecom, surveillance, or military, you can set up a spare virtual environment, boot an extra copy of the OS in that environment, and then "freeze" that environment. Then, if something bad happens in the active run-time environment you can migrate active processes to the back-up OS, and not lose all of the normal time to boot. This same mechanism can be used to upgrade firmware in the background, validate the new image, and then migrate active process threads to the newly updated run-time image.

Once you start thinking in terms of having lots of extra-virtualized processors, it's pretty easy to think of places to use them.

Q. **A few RTOS companies such as QNX have focused lately on taking advantage of multi-core. How is your solution different from that of a QNX or a Green Hills Software?**

A. In essence, we sit underneath the OS as a very thin extended function HAL. We're cooperative with any multi-core OS and synergistic in that we can improve the resource loading and power management picture in a fashion that's very transparent. If the OS is looking down at a pool of 8 virtual processors, it's unimportant to its execution whether those 8 virtual processors are running on 8 physical cores, or just one. In other words, we can dynamically reassign the virtual to physical resource mapping in a fashion transparent to the OS.

Q. **Does *Trango Hypervisor* work with every RTOS? Or is it available in designs only with *Embedded Linux*?**

A. Our product uses a virtualization method called "paravirtualization". What that means is that in order to improve performance and code size; we make some edits to the OS. Anything that was a privileged-mode access to hardware in the native unmodified OS, becomes an API call to the *hypervisor*. It turns out that there aren't too many of these

calls in a typical OS so it's not a major effort either from the standpoint of the engineering time and effort, or from the standpoint of the actual behavior of the OS.

So far, we've done several flavors of *Linux*, *Windows CE*, *MicroC/OS-II*, and *eCos*, and our customers have done their own changes for several proprietary systems.

Q. **On your web page, we see that you support the *ARM* and *MIPS* architectures. Your solution is thus hardware-centric, meaning that someone doing a *PowerPC* design (for example) cannot use it. Is that correct? Do you plan to support additional architectures?**

A. It is correct that the solution is very hardware-centric, but as we are extremely customer-driven I can state unequivocally that if we were to engage with someone working with a *PowerPC* design they would be supported. The only real requirement for our technology to work is that the underlying architecture has an MMU, or an MPU with enough flexibility for us to manage memory effectively, and at least two privilege levels.

Q. **How much does it cost? What are the available pricing options?**

A. There's a degree of NRE in porting to a new architecture, and for popular architectures that's probably going to be pretty low. Then there's a bit of NRE associated with creating the board support package (BSP) for the target application.

With respect to licensee fees and royalties, we follow typical practice in the software industry. We're extremely flexible as no two customer's needs are identical. There is of course a dependency on the product's ASP and expected volume. In addition, some customers prefer a bigger up front fee and a smaller royalty, while others prefer it the other way around.

Q. **Thank you for your interview.**

VIRTUALLOGIX: VIRTUALIZATION FOR THE "CONNECTED WORLD"

18 October 2007: VirtualLogix: Virtualization for the "Connected World"

INTERVIEWEE. MARK MILLIGAN
VICE PRESIDENT OF MARKETING
TEL. (408) 636-2800
EMAIL. VIRTUALLOGIX@SCHWARTZ-PR.COM
COMPANY. VIRTUALLOGIX
WEB. http://www.virtuallogix.com/

Q. **First of all, tell us a little bit about yourself and VirtualLogix. What is your 10-second "elevator pitch?"**

A. As Vice President of Marketing, I am responsible for driving VirtualLogix's product strategy, product roadmap, and marketing communications programs. I have more than 15 years of executive marketing and management experience in semiconductor and system software. Previously, I served as Vice President of Marketing for CoWare, Inc., a leading supplier of system-level electronic design automation software and services. During that time, I was also president of the Open SystemC Initiative (OSCI), an open source industry organization establishing a next-generation language standard for the development of electronic systems combining hardware and extensive embedded software. In addition, I have held executive marketing positions at Synopsys, a leading electronic design-automation company, and Vice President / General Manager at Viewlogic Systems and HPL Technologies (two companies that were acquired by Synopsys).

VirtualLogix provides what we call "*Real-Time Virtualization*™," which is software that creates virtual copies of hardware, enabling operating systems and their applications to run more efficiently, with reduced bill of materials costs, improved management, and higher security. Our unique technology is able to handle the real-time requirements of embedded systems such as what is found in a 3G mobile phone or WiMax base station, as examples. We are a global company with close alliances with semiconductor companies such as Intel, Texas Instruments, NXP Semiconductor, as well as operating systems companies such as Monta Vista and WindRiver. To-date we have raised $36M in venture funding from notable investors including Atlas Venture, Index Ventures, DFJEspirit, Cisco, and Intel.

Q. **What, in a nutshell, is "virtualization?" How is this related to real-time operating systems?**

A. Virtualization enables multiple operating systems to run "simultaneously" on shared hardware resources. The benefits of virtualization include reducing bill of materials by consolidating hardware and improving device management and security.

But another major trend we see today is the interest of embedded systems developers to leverage rich and open operating systems such as *Linux*. Virtualization has a unique role to play to accelerate the adoption of *Linux* by enabling *Linux* to run side by side with a legacy operating system such as a commercial or proprietary RTOS. This eliminates the need to rewrite and reverify legacy applications, yet gain the benefits of *Linux*. It also means that the hard real time requirements can be managed by an RTOS optimized for that specific environment, without trying to meet these requirements in *Linux*. There are also additional benefits of using virtualization to combine an RTOS with *Linux*, namely unique intellectual property may be developed in the RTOS, thereby protecting it from GPL contamination. Lastly, but importantly, virtualization can be used to create "open" operating environments, and combine it with protected "closed" environments. So on a

mobile phone for example, the open stack might get corrupted from a virus whereas critical functionality like the basic phone functionality, user authentication, billing, digital rights management, etc, could be placed in a "sandboxed" OS and protected from corruption by the virtualization layer.

Q. **What, specifically, is the VirtualLogix product offering? Is *VLX* an add on or a replacement to the developer's RTOS?**

A. *VLX* most definitely does not replace the RTOS. In fact it is completely OS agnostic, it can work with literally any RTOS together with a rich operating system such as Linux. Given the dozens or hundreds of RTOS out there in the embedded world, this is a crucial capability. Developers do not need to rewrite those applications optimized to run on a legacy RTOS, yet they can still leverage the open source community and the many available applications available using a rich OS.

We have several variations of *VLX* optimized for specific end-markets. For example *VLX* for Network Infrastructure is optimized for the high reliability and throughput requirements found in communications gear – whether it is an ATCA core network component or a DSP-based wireless base station. *VLX* for Mobile Handsets is targeted at bringing the rich functionality of a smart phone into a feature phone sub-$100 price point, and also adding improved security and manageability to all manners of smart- and feature-phones. *VLX* for Digital Multimedia leverages the strong real-time capabilities of Texas Instruments' TI/DSP Bios, and adds in the richness of *Linux*, all running together on a TI DSP without the need to add an additional applications processor. *VLX* can also be applied to many other traditional embedded environments, like Military, Medical, automotive, etc. We like to say, if it is a connected device, real-time virtualization™ could make it better, cheaper, safer, and easier to maintain.

Q. **A developer might usually think of "virtualization" as a cutting edge technology available today only in the server market. How is "virtualization" relevant to embedded systems? Is it relevant today or is this something for the future?**

A. Yes, indeed many developers hear the term virtualization and immediately think of the technology used in a datacenter, and think "no way could it be applied to my embedded design". That is why we call our technology "real-time virtualization™ for connected devices". Our expertise and our unique technology are able to guarantee hard- or soft-real time performance, with extremely low latency and small memory footprint. Now, we have customers designing everything from mobile phones, to set-top boxes, to video surveillance cameras, to wimax base stations, to soft switches. The end products are either on the market or will be soon. So clearly, virtualization is ready for embedded systems – it just requires a totally different technology under the hood compared to what is needed in a datacenter. It is the next wave for virtualization, to go after these "front end" devices to go along with the virtualized, back-end datacenter.

Q. **What vertical applications do you think will most benefit from virtualization?**

A. We see all kinds of communications gear and digital multimedia devices as being able to benefit immediately from virtualization. One of the business drivers is connectivity – the more devices converge and require connection to something else, the more the virtualization becomes an ideal solution. From a technical point of view, the drive towards leveraging the open source community, and the semiconductor industry's move toward multi-core processors, are key drivers. As stated before, virtualization has a great role to play to speed the incorporation of *Linux* into embedded systems by combining it with legacy RTOS and their applications. Virtualization also enables software that was written for single core processors to be moved onto multi-core processor without rewriting the entire application stack. This has tremendous value for companies wanting to leverage their existing software investments but need to get on board with the

semiconductor roadmap of multi-core processors. This later example was a key reason why Intel made an equity investment in VirtualLogix.

Q. **Does the developer have to go with certain RTOSes to be able to take advantage of your technology? Or can it be employed with any and all RTOSes?**

A. It can be employed with any and all RTOSs. We are both technically and strategically OS agnostic. This means companies can standardize on VirtualLogix as their embedded virtualization supplier, and use it on many different projects within the organization that may use many different RTOSs in different parts of the business. These RTOSs may be internally developed "in-house", or may be commercially provided such as WindRiver.

Q. **What about hardware? Is *VLX* hardware, or chip-specific? Does it only work on certain architectures such as ARM?**

A. VirtualLogix is also strategically hardware independent and can run on many different hardware architectures. This is another key difference compared to virtualization companies in the datacenter space, which are primarily x86. *VLX* today supports *Intel Core Microarchitecture®*, *ARM*, and TI DSPs. And we also have existing customers using *VLX* with other well-known processors, so this list will clearly expand, as customers require. I will add that in order to take advantage of each unique processing architecture and be able to ensure maximum performance and throughput, we put a lot of effort into optimization for each processor architecture that our customers request. We are able to leverage specific hardware features such as virtualization support in hardware, like the *Intel VT* capabilities, or security additions such as *ARM TrustZone*™

Q. **Other vendors, for example QNX, have talked about multicore (both synchronous and asynchronous). How is "virtualization" different from "multicore"?**

A. Well, in one sense all our customers' designs are multicore, because in effect we are creating multiple – but virtual -- processors from what the software sees. But *VLX* can work on both single- and mult-core processors, and in all cases we are able to get the maximum benefit from the hardware, making the software run more efficiently and safer.

Q. **Some of these RTOSes - QNX *Neutrino*, for example - proclaim that they are well on the road to exploiting multicore. Is the *VLX* solution competitive or complementary to these RTOS-based solutions?**

A. Complementary. It isn't just about running the same RTOS on multiple cores, but often combining different operating systems together on a multi-core system, and doing it safely and securely. For example, today SMP Linux is sometimes run on top of *VLX*, and the developers additionally combine it with an in-house RTOS (and its applications), or *VXworks*. It is this mixing and matching, plus the need to make some OS stacks open and some closed in today's connected world, that create the requirement for an OS-agnostic virtualization technology.

Q. **When *VLX* is employed, how does it allow the various RTOSes or GPOSes to share resources with each other? Does it? Or does each OS live in its own "walled garden?"**

A. There are lots of options here, since the key to embedded system design is optimizing for a given set of market requirements. That is why we provide *VLX Developer*, a set of *Eclipse*-based development tools that enable engineers to optimize, configure, and build the virtualized environments. We also offer a range of performance and security tradeoffs to be made within *VLX* to create just the right characteristics for our customers to meet their market needs.

For example, devices may be shared or "owned" by one OS, it is select-able by the developer. Resources can be assigned to each guest OS and the scheduling scheme selected from a range of options, so you might want to guarantee a certain percentage of

resources flow to a specific task, or you might want to use pre-emptive scheduling. Memory may be partitioned, and if a MMU is present, memory access restricted to prevent one guest from corrupting another.

Q. **How much does the product cost? Tell us about your sales model - is it royalty based, up-front licensing? What are the ways in which one can purchase the product?**

A. Our customers purchase a development license of the software for their design phase, and a deployment license for their production shipments. We also provide optional professional services to assist with speeding our customer's time to market. The pricing varies according to various factors such as the volumes and end-unit pricing.

Q. **Virtualization seems to have caught on first in the datacenter, when do you see it flourishing in the embedded space?**

A. With virtualization becoming mainstream in the datacenter, it is just a matter of time before it also becomes mainstream in the embedded market. Often, technologies start in the IT environments, and later migrate into embedded designs – at least the concepts do. Usually, the technologies end up being quite different to meet the real-time and other unique needs of embedded applications. That is the case for VirtualLogix – while the concepts are very similar to what is happening in the datacenter, our *Real-Time Virtualization*™ technology is quite different under the hood. We see these connected devices as being the new frontier for virtualization.

Q. **Thank you for this interview.**

Interviews: Beyond the RTOS

Many of the real experts on RTOSes are the vendors themselves. We interviewed many leading vendors and tried to tie each to an important "theme" to help those making an RTOS choice. Here are interviews relating to "beyond the RTOS," meaning GUIs, enterprise / embedded convergence, porting, quick development and more..

- ALT Software: Beyond the RTOS, GUI and Graphics Development
- Ardence: Real-time Windows, Instant On, and Multiple Device Personalities
- Enea: Middleware and Application Software Beyond the RTOS
- Eridon: *UnifiedLogic Development Platform* for "Beyond the RTOS"
- MapuSoft Technologies: the Benefits of RTOS Porting & Abstraction
- Microsoft: Connected Experiences based on Windows Embedded
- Zeidman Technologies: Synthesize your Own RTOS

Insiders' Guide Embedded RTOS: Interviews: Beyond the RTOS

ALT SOFTWARE: BEYOND THE RTOS, GUI AND GRAPHICS DEVELOPMENT

29 October 2007: Beyond the RTOS, GUI and Graphics Development

INTERVIEWEE. DAVE NICKERSON
DIRECTOR OF PRODUCTS
TEL. 1 416.203.8508 EXT 1105
EMAIL. info@altsoftware.com
COMPANY. ALT SOFTWARE, INC.
WEB. http://www.altsoftware.com/

Q. **First of all, tell us a little bit about yourself and your position at ALT Software.**

A. My background combines 10 years of embedded hardware experience maintaining specialized military airborne reconnaissance systems with over 10 years experience in the embedded RTOS and graphics market. While at QNX Software Systems, a leading RTOS company, I worked closely with industry leaders such as 3Com, Intel, and Freescale to define and integrate software functionality for a variety of reference boards and commercial devices. I was responsible for developing the strategic direction for the Automotive market segment. This included defining the QNX's hardware support for automotive platforms from various silicon providers such as Freescale, Renasas, and Texas Instruments, ultimately helping QNX secure its leadership position in the telematics market.

At ALT Software, I am responsible for all products, product management, the development of product roadmaps and business models, and am a key figure in evaluations of all existing and potential partner technologies, including RTOSes.

Q. **This edition of our *Insiders' Guide: RTOS* has a theme of "Beyond the RTOS." How does ALT Software take an engineer "beyond" the RTOS?**

A. Many of our customers require expertise and resources to help them get the OS/RTOS of their choice running successfully on their hardware platforms. ALT has over a decade of experience developing system level software for custom or commercial hardware platforms. We are experts at developing board support packages, device drivers for a variety of peripheral devices for many different OSes.

In order for device manufacturers to differentiate their products in the market place, they must provide their own custom look and feel. Today, that often means bringing a much more complete looking user interface to an embedded device; be it a cell phone, a heart rate monitor or a flight control system. End customer expectations have risen to where they demand interfaces and functionality that was once confined to the desktop space. ALT has the experience and products to bring full 3D accelerated graphics functionality to the embedded market. This is what really differentiates most products today.

Q. **What sorts of applications might benefit from your graphics technology? What sorts of engineers might be interested in buying your software as opposed to building it themselves, and why?**

A. ALT Software's team of engineers is involved with projects from every segment of the market that employs graphical user interfaces. Drivers written for the embedded space have different demands than those written for the desktop. Drivers for embedded devices must have high performance, high reliability while maintaining a small memory footprint. Many device manufactures have large development teams dedicated to application development but lack the specialized teams with the expertise required to write low-level drivers and put these together with the RTOS on the embedded hardware

platform. This is where ALT software expertise can help software teams reduce their development cycles as well as costs.

ALT also provides drivers and solutions for the latest 3D hardware and industry standard OpenGL API. OpenGL provides a standard programming interface for 3D graphics primitives, allowing application teams to quickly write compelling and intuitive user interfaces for their applications and devices. Many of the latest Graphic Processor Units (GPUs) from industry leaders like Intel, ATI, and Fujitsu provide hardware acceleration for OpenGL applications. This allows developers to offload 3D interface rendering from the main CPU onto the GPU, reducing the load on the processor.

Q. **Tell us a little bit about your specialty vis-a-vis military or "safety critical" applications that benefit from advanced graphics.**

A. ALT Software has been involved in numerous aerospace and defense programs over the years, with customers based both in North America and Europe. First and foremost, we have been involved in this industry with our OpenGL suite of products. In fact, our first OpenGL driver for the embedded market space was done as a part of a military cockpit upgrade covering Primary Flight Displays, Multi-Function Displays, and Heads-Up-Displays. Since that time, we have added support for numerous GPUs, extended support onto multiple operating systems, and added a line to the product for high-end graphics processors that has been certified to the highest and most stringent standards for embedded software in an Airborne System, DO-178B Level A. We are currently under contract with multiple large airframe manufacturers worldwide with this product line.

While our initial exposure was through our OpenGL solutions, we have extended ourselves far beyond this point. ALT has a strong background in driver and other low level development for embedded systems, and has expanded this into the safety critical realm. We've been involved in projects for a variety of communications busses (ARINC-429, RS422, MIL_STD_1553, ASCB, Ethernet), peripheral devices (Cursor Control Devices, XM weather, and other data loaders), and application development. Related to work with safety critical applications, we've been involved in efforts creating software for Primary Flight Displays, Multi-Function Displays, and Navigation displays, as well as several application porting efforts leveraging the use of Previously Developed Software on certified programs.

In addition to this, ALT has been brought into several programs as a design or certification consultant. Our background in low level embedded software, OpenGL graphics, and mitigating the risks of using COTS GPUs in avionics, as well as our sought-after QA services and strict software development methodologies, has been a valuable service to our customers in order to ensure effective system design and efficient program certification at all phases of the program lifecycle.

Q. **ALT Software's graphical products are based on OpenGL. Can you tell us in a nutshell what is "OpenGL" and why is this advantageous?**

A. OpenGL is an open standard for a graphical rendering API. Originally developed by SGI, and currently maintained by The Khronos Group, OpenGL offers an API designed to facilitate rendering high quality 3D graphical images. The API itself is published to the general public, and industry specialists are able to develop their own implementation of the specified standard either in software or hardware specific to their purpose.

As a standard, use of OpenGL supports a lower cost to market, a lower cost for maintenance, and a lower cost to upgrade for embedded platforms. OpenGL is used across industries from gaming to Computer Aided Design, and from embedded medical devices through automotive, industrial, and aerospace. Many Universities offer courses in computer graphics with a focus on OpenGL. This means that competent OpenGL programmers are readily available. This helps to ensure development of higher quality software over a shorter software lifecycle, greatly increasing its overall maintainability.

In aerospace programs, it is common to have to support programs for 20 years or more. By using an open standard graphics language such as OpenGL, risks are mitigated when it comes to maintaining the software in the future by the continued availability of an educated pool of developers. Finally, while all OpenGL implementations offer variances, the basic concepts are standard. By using the OpenGL standard, a cost savings can be realized on near-term programs where hardware is upgraded from a predecessor, a new hardware/software vendor is selected, or an application suite is being ported from one hardware platform to another.

Q. **Is OpenGL subject to the GPL license like Linux or BSD? How does ALT Software's product relate? Is it also "open source?" How does your own licensing work?**

A. OpenGL source code is subject to various license terms as referenced on the OpenGL.org website, http://www.opengl.org. However, in general OpenGL application developers do not license the OpenGL API. Instead, ALT Software provides our customers with the linkable versions of the OpenGL libraries required by developers to write their OpenGL applications.

Q. **How is your product sold? What are the available models for a business engagement with you?**

A. ALT Software sells general purpose OpenGL drivers and safety critical OpenGL drivers. The general-purpose products, which also support the GLX, GLU and GLUT APIs are only sold in binary format. In order to ease the certification process, ALT's safety critical driver supports a subset of OpenGL 1.3 and also offers the option of procuring source code and a sophisticated certification evidence package. Both the general purpose and safety critical software licenses are site based and linked to a single product configuration.

ALT Software's *COTS* graphics libraries have been used in so many different systems over the years that we have been able to design them in a way that they will satisfy the requirements of 90% of our customers. In the case where custom features are required for a given system, ALT has a professional services team which is able to work with customers and deliver those special features when required.

ALT Software also offers extended product maintenance and support contracts that entitle the customer to product upgrades, technical support and bug fixes during the contracted maintenance term.

Q. **Thank you for this interview.**

ARDENCE: REAL-TIME WINDOWS, INSTANT ON, AND MULTIPLE DEVICE PERSONALITIES

24 October 2007: Real-time Windows, Instant On, and Multiple Device Personalities

INTERVIEWEE. TODD TRUEX
LEAD PRODUCT MARKETING MANAGER
TEL. **781 693 6256**
EMAIL. **TODD.TRUEX@CITRIX.COM**
COMPANY. ARDENCE, A CITRIX COMPANY
WEB. **http://www.ardence.com/**

Q. **First of all, tell us a little bit about yourself and your role at Ardence.**

A. Ardence is about to celebrate its 28th year in the software market. Since its inception, Ardence has focused on real-time enhancements for Microsoft systems running on Intel hardware platforms. In response to customer demands for performance and a common development environment, we introduced *Ardence RTX*, which provides hard real-time on Microsoft platforms for critical applications. *RTX* is a *Visual Studio* plug-in therefore preserving a common development tool chain. *Phar Lap ETS*, our stand-alone RTOS, is one of the most widely deployed real-time system platforms in the industry. Additionally, we provide "instant-on" and multiple configuration capabilities with *Select*, and *ReadyOn*, making systems more secure, with immediate availability.

My current responsibilities involve all aspects of product marketing for the Ardence Embedded Product line.

Q. **Ardence recently became part of Citrix. What is Ardence's "vision" for embedded systems and networked devices? How does this fit in with Citrix?**

A. Citrix specializes in application delivery and desktop virtualization. With the recent acquisition of XenSource, Citrix now has an end-to-end virtualization solution, from server to desktop. One key component of this virtualization strategy is the Ardence streaming software, which provides streaming OS delivery to diskless or thin clients. The Ardence Embedded Group takes advantage of this capability to stream operating system environments to embedded devices, providing secure and corruption-proof management and configuration of remote embedded systems, creating embedded Dynamic Devices. We see this convergence of networks and managed embedded systems as a growing trend in the industry, and we are fortunately well equipped to provide these end-to-end solutions.

On the other side of the coin, Ardence Embedded specializes in kernel-level software, which provides Citrix significant resources to address "under the hood" solutions at very fundamental levels within their software architectures. This enhances our capabilities and overall solution flexibility and creates a nice balance between the two sides of the business, Enterprise and Embedded, as both groups offer capabilities and solutions that benefit the other. The Ardence Embedded business will continue to service existing embedded systems markets, and will continue to grow into new areas, in addition to providing solutions to further extend the Citrix virtualization strategy.

Q. **Tell us about *RTX*®, or "Real-time Extensions for Windows." Microsoft has their own *Windows XP Embedded*. So how does your product enhance Windows *XPe*?**

A. *Microsoft's XP embedded OS* is a componentized version of their *XP operating system*, capable of being customized for specific implementations, allowing developers to

minimize the software footprint. *RTX* is implemented as a collection of libraries (both static and dynamic), a real-time subsystem (RTSS) realized as a Windows kernel device driver, and an extended HAL. The subsystem implements the real-time objects and scheduler previously mentioned. The libraries provide access to the subsystem via a real-time API, known as *RtWinAPI*, which provides access to these objects and is callable from the standard *Win32* environment as well as from within RTSS. So what we have basically is a kernel level driver that control all interrupts to Windows, providing highly deterministic response and performance. This is attractive to our customers as they can operate totally within a Windows environment, reducing overall costs, and allows them to develop using common and familiar tools, also saving them time and money. The result is a windows platform that has the necessary speed and response times needed to execute and control critical operations.

Q. **How is your solution different from competitors, such as Tenasys, that also bring real-time to Windows? What technical points would you recommend a designer consider to compare and contrast?**

A. As with any tool, program or application, there can be any number of ways desired functionality could be achieved. Our competitors certainly are able to deliver a platform that performs the desired functions, albeit it in a different manner than Ardence, and at a much lower performance level. We believe that a platform that has the more familiar and easier interface to use will likely win out, as developers are prone to working within environments in which they are comfortable. *RTX* allows developers to develop code in user mode, taking advantage of Windows debugging tools and memory protection available in hardware. Once optimized, the code is easily recompiled into kernel mode, where it will operate at much higher performance levels, without the impairments associated with memory protection while running in user mode. This allows developer to do what they do best- focus on developing value-add features and functionality, not spending cycles performance tuning and recompiling. Others can do this, but not with the ease and user-friendly interface we offer. *RTX* has supported multicore processor architectures prior to our competitors for several years and has a proven customer base of thousands, and installed systems numbering 10s of 1000s. Being close to Intel as an ICA and SPP member has facilitated our support of their architectures through the years.

Q. **What is "ReadyOn®?" How does this product relate to Ardence's vision of "instant on" for devices?**

A. *ReadyOn* is a great product from a number of perspectives. It enables very rapid availability of systems upon power up, bringing them to a pre-configured, non-corruptible state. This is ideal for embedded applications where specific purpose applications are used- users can simply power cycle the system to reset the platform to the preset image. This ensures protection against performance degradation due to malware that may try to install itself over networked applications. It also allows for rapid power down without the usual shutdown processes- one needs merely to shut off the power, without risk of software degradation. *ReadyOn* is also useful for ghosting images in high volume manufacturing operations, such as PCs, where a standard application image can be copied to many target devices. Finally, *ReadyOn* also plays well into the growing concern for "green" products, as devices can be literally be turned on and off as needed with no delays or unnecessary power consumption during idle time. *ReadyOn* is our answer for secure, instant-on capabilities.

Q. **Ardence has a vision of devices' having "multiple personalities." What is meant by this? Is this your *Ardence Select*™ product?**

A. Yes. *Ardence Select* is an extension to *ReadyOn*, allowing instant-on capabilities to multiple system states, or images. This is useful in environments such as media PCs, where appliance-like performance is a vital to the user experience, or in POS applications, where multiple product lines or services may be required. In most cases, this system transition is transparent to the user, as the device image can reboot in seconds. *Select* is a

product that we offer more as a service, as each OEM has unique requirements for the products they offer, so we usually engage them on a case-by-case basis to make certain we provide the solution that exactly fits their requirements.

Q. **We hear a lot nowadays about the convergence of enterprise and embedded devices, especially in terms of network traffic. What sorts of devices and applications is Ardence most interested in pursuing? What applications have the most to gain from Ardence?**

A. I think the answer is more along the lines of what devices and applications we are *not* interested in pursuing at this time. If a device is networked, there is a strong possibility we have something to offer that will improve its performance, security, or availability. If a device is embedded, we have products that can provide networked delivery of operating systems, device states, and other tools that enable remote device management, as well as insuring critical performance criteria. No other embedded software company has the breadth of solutions available to the market today. We will continue to play on our strengths in the embedded market while leveraging the opportunities available to us through Citrix and XenSource.

Q. **Is there a vision of how the various Ardence products might "work together" and not just with each other but also with Microsoft Windows to create a more total solution? What might this be?**

A. Working with *Windows* to enhance capabilities and extend its market reach is one of the cornerstones of our market approach. Ardence has enjoyed a long-term relationship with Microsoft, and we are currently a gold level embedded partner. That relationship is only strengthened with our integration into Citrix.

As previously mentioned, we have a wide range of solutions that we can provide. One such implementation may be a streaming OS delivery to a remote device with real-time capabilities. Another could be an embedded, instant-on kiosk for gaming or POS applications. We could create multiple images, and stream various platforms on demand, depending on selected use or application for a given point in time. There is a wide range of applications, not in the traditional embedded markets, where we can extend and enhance *Microsoft windows platforms.*

Q. **Thank you for this interview.**

ENEA: MIDDLEWARE AND APPLICATION SOFTWARE BEYOND THE RTOS

17 December 2007: Middleware and Application Software Beyond the RTOS

INTERVIEWEE. TERRY PEARSON
VICE PRESIDENT, MARKETING
TEL. (603) 888-7575
EMAIL. Terry.pearson@enea.com
COMPANY. ENEA
WEB. http://www.enea.com/

Q. **First of all, tell us a little bit about yourself and your responsibilities at Enea.**

A. I am the VP of Marketing at Enea. I was one of the key designers and contributors to our Element Middleware product. Prior to Enea, I spent more than 15 years developing and leading software teams that were chartered to build distributed high-availability datacom and telecom platforms at Tier 1 telecom equipment manufacturers and early stage startups. I hold a master's degree in computer engineering from Boston University.

Q. **Enea is one of the older and more known RTOS companies, but today the impression is that Enea sees the future "beyond the RTOS," especially in your middleware product, *Element*. What is Enea's vision for the embedded software market, where do you see "value?"**

A. As described, Enea has had several RTOSes in our product portfolio since the mid-eighties. The last couple of years we have expanded our product portfolio in order to meet the market requirements as we see them. Our identification is that customers today request more than an RTOS. We see this as a natural extension to the fact that they stopped building their own hardware and their own Operating Systems. We believe that they are now in a transition phase to where they cannot justify building their own proprietary middleware any longer with time to market and internal cost structure as the main drivers.

Element, our middleware product, is a direct response to this trend. But we have taken our offering one step further and developed the *Enea Accelerator*™ platform which is a pre-integrated SW-platform addressing very much of the definitions made by SCOPE Alliance. *Enea Accelerator*™ consists of:

- Operating Systems of your choice - e.g. ENEA *RTOS* but also including several *Linux* distributions.
- *LINX* - A system wide, scalable and hi-performance Inter Process Communication giving your system one interface to communicate in your system. A *Linux* version is available as Open Source.
- *Element* – The most comprehensive Middleware product in the market supporting "Carrier Grade" requirements. e.g. High-Availability, Platform Management, SW Management and Embedded Management
- *Polyhedra* – A family of standards based relational databases, in-memory or flash based and designed for embedded use.
- Professional Services - To further accelerate the customer's time-to-market demands and ensure the customer's sustainable development efficiency

The *Accelerator*™ offering provides our customers with a platform where the most difficult "Carrier Grade" requirements are addressed and accessed via Standardized (SAF) interfaces.

Q. **Embedded *Linux* has all but conquered the telecom infrastructure verticle. How does *Element* in particular and Enea in general "play" with *Linux*?**

A. As described above both *Element* and *Accelerator* supports a number of *Linux* distributions. Our *LINX* product is also distributed as an Open Source version for Linux in order to support our customers' heterogeneous Operating System environments.

Finally Enea has also established a *Linux Competence Centre* to emphasize us as being "OS-Agnostic."

Q. **Is it fair to say that *Linux* is commoditizing the RTOS business and thereby moving competition to application software, middleware, and the like? Should developers begin assessing companies not on their RTOS but on their other software products and services?**

A. *Linux* has definitely captured a portion of the RTOS market and is high on our customers list of Operating Systems when new designs are considered. In terms of assessing companies, Enea believes that the most important aspect to assess is the value the companies can offer - e.g. we strongly believe that a standards based carrier grade platform letting the customers focus on their application development offers great value in terms of time to market and sustainable efficiency and "feature velocity."

Q. **On a technical level, can your customers purchase *Element* without purchasing one of your RTOSes?**

A. Yes absolutely, see above!

Q. **Tell us about your *dSPEED* product. It sounds ideal for telecommunications infrastructure or possibly military communications applications, is that correct?**

A. Correct! A basic identification Enea has made over the last couple of years is that integration is key to success. By *dSPEED*, Enea has productized another area of complex development and integration that is both tedious and resource consuming. *dSPEED* gives you a solid and flexible middleware foundation for managing DSP farms (supports TI and Freescale DSPs today). *dSPEED* is a unique offering in itself today and when using *LINX* to connect the control plane (*Element MW*) and data plane *(dSPEED MW*) together our customers get a unique standards based system platform architecture that significantly affects time to market and development cost.

Q. **How does Enea sell its products? What are the available business models for engagement?**

A. Enea's ambition is to work close with our customers and when it comes to pricing we are flexible since we realize that our customers have different preferences. We work with the full range of options in co-operation with our customer, e.g. perpetual development licenses per project, rentals and royalties.

Q. **Thank you for this interview.**

ERIDON: *UNIFIEDLOGIC DEVELOPMENT PLATFORM* FOR "BEYOND THE RTOS"

23 October 2007: *UnifiedLogic Development Platform* for "Beyond the RTOS"

INTERVIEWEE. MADISON TURNER
DIRECTOR OF MARKETING
TEL. (952) 474-5110 EXT.372
EMAIL. MADISONT@ERIDON.COM
COMPANY. ERIDON
WEB. http://www.eridon.com/

Q. **Eridon is obviously not a "traditional" RTOS vendor, yet your technology has implications to designs that would employ an RTOS. Can you briefly explain your technology?**

A. The *UnifiedLogic™ Development Platform* spans both hardware and software, utilizing FPGAs to simplify embedded design. The platform includes an RTOS and software-centric IDE along with reference designs and prototyping hardware, all tightly integrated to form a complete development system. You use the *UnifiedLogic IDE* to specify your hardware requirements in terms of modular subsystems. The IDE then automatically performs two major steps in both the hardware and software design efforts—it automatically generates the necessary logic to integrate your processor and peripherals (in an FPGA-based platform) *and* configures the *UnifiedLogic RTOS* with the appropriate drivers and libraries to support the design. In short order, you go from concept to coding your application on real hardware.

If you're using Eridon's *uCards*™ for development, you simply snap together the boards you need to create a working prototype of your product (which includes means for adding your own custom sections). Off-the-shelf peripheral modules include Ethernet, serial communications, USB 2.0, CAN, audio and video, among others. Third-party development boards from Xilinx and others are also supported, and it is easy to go directly to your own custom design as well.

UnifiedLogic reference designs are available online and come with complete schematics and bills of materials (BOMs). So regardless of how you prototype, you can go straight from a high-level functional specification to laying out a production board without having to research and integrate discrete components. You can even get an initial production cost estimate from the online calculator on our website at http://www. eridon. com.

The main idea is to let you build a product much faster and with much less risk by using modular subsystems and automating the integration of the RTOS and hardware design, not to mention providing a programming and debugging environment, all with a single highly-integrated IDE.

Q. **And, briefly, what is your position at Eridon?**

A. As Director of Marketing, my job is to spread the word about *UnifiedLogic* and to listen to the industry in turn.

Q. **What is the relationship of Eridon's *Unified Logic Development Platform* to FPGAs? Which FPGAs does it support - Xilinx, only? Is it FPGA dependent?**

A. FPGAs are the key enabling technology that lets *UnifiedLogic* automatically integrate peripherals around a processor in a modular fashion, without the added overhead of a standard bus. FPGA technology also provides the "liquid silicon" environment that allows for pervasive but transparent use of smart peripherals. Incorporated into the *UnifiedLogic* RTOS and software drivers, these hardware-software blended peripherals

offload compute-intensive functions to gates (hardware) to better deliver real-time performance.

Since no knowledge of FPGAs is required, a software engineer with only a basic hardware background can quickly assemble a functioning prototype (RTOS plus hardware) and immediately begin coding the application.

UnifiedLogic only supports Xilinx FPGAs at this time, including a wide variety of *Virtex-II Pro*, *Spartan-3* and *Virtex-4 chips*.

Q. **In terms of technical RTOS characteristics, what exactly is the *UnifiedLogic RTOS*? How does it compare and contrast in terms of technical performance with more traditional RTOSes like *VxWorks, Nucleus,* or even embedded *Linux*?**

A. The *UnifiedLogic RTOS* is a modern, object-oriented RTOS designed to be programmed in either C or C++. In many ways, it is a fairly typical RTOS with the kinds of services you would expect: threads, mutexes, semaphores, events, timers, I/O requests, message queues, and so on. It's small and fast, and the source code is provided.

It excels in the breadth and depth of its drivers and application support libraries. FAT and Flash file systems, USB and Ethernet are just a few examples. The drivers are robust and task-oriented rather than just exposing the hardware. The APIs are designed for simplicity and ease of use.

But what really differentiates *UnifiedLogic RTOS* is that it is FPGA-enabled, built from the ground up to take advantage of an FPGA's programmable hardware to deliver flexibility and real-time performance. It views a peripheral as having both a software interface as well as a hardware component formally defined in HDL along with schematics and a BOM. This means you can put just the peripherals you need in what is essentially a cost-effective custom chip. Further, as an integral part of the *UnifiedLogic Development Platform*, the RTOS is automatically configured around your peripherals with just the required services and libraries and seamlessly integrated with a powerful IDE. This gets you to a custom platform much more quickly and lets you focus on coding and debugging your application.

Q. **Is your target customer a software engineer, a hardware engineer, or both? How much knowledge is required of FPGAs?**

A. Both. Software engineers use the *UnifiedLogic IDE* to specify the system and then write an application on top of the *UnifiedLogic RTOS*. Hardware engineers can focus on adding their own secret sauce in custom modules, as many common but complex subsystems are simply pulled in as reference designs. Ultimately, the hardware engineer uses the provided schematics and BOMs to lay out the production hardware. He or she will likely want to "right size" the FPGA—selecting the most cost-effective FPGA for the design. Otherwise, you can treat the FPGA as just another chip. You don't need to know how to write VHDL or use FPGA design tools.

Opening a door to software engineers, *UnifiedLogic* allows a software engineer without any in-depth hardware expertise, not to mention knowledge of FPGAs, to assemble a hardware prototype and automatically generate an FPGA image for it. This allows companies to more quickly and with less risk put hardware together around a product concept, focus on the application software, and get it in front of their customers.

Q. **What target applications or vertical markets do you think will be most excited about this novel approach to RTOS and embedded hardware? Why?**

A. Any development team that would prefer to spend their days building their own application rather than integrating the underlying components will be excited about *UnifiedLogic*. Our current customers are using it for applications as diverse as industrial control, retail automation and multimedia processing. That being said, there are some

specific industries and scenarios where the benefits of using *UnifiedLogic* are particularly relevant.

For design houses and contract engineering firms, delivering a working prototype of the application running on hardware along with your proposal often clinches the contract. *UnifiedLogic*'s rapid prototyping capabilities can provide a major boon to startups seeking funding and entrepreneurial groups within larger organizations as well.

The selection of off-the-shelf peripheral modules available for *UnifiedLogic* also suggests particular application types. Industrial applications benefit from an on-chip CAN implementation. Multimedia devices can take advantage of audio and video support. And for consumer devices, a fault-tolerant, wear-balancing Flash file system along with networking and USB 2.0 support are often a required foundation.

Q. **What is Eridon's relationship to the "open source" RTOS, *eCOS*? How do designs employ both technologies, and what does this do for them?**

A. We provide a compatibility layer that lets *eCo*s packages and applications run on top of the *UnifiedLogic RTOS*. This has several benefits for *UnifiedLogic* users and *eCos* users alike. First, it makes available a variety of tried-and-true software packages – like networking protocols and web servers, for instance. Nobody has to reinvent the wheel. It also lets *eCos* users migrate their legacy code to *UnifiedLogic* with very little effort. They can take advantage of all the flexibility and integration of *UnifiedLogic* without starting over from scratch. And because the *UnifiedLogic RTOS* is considerably faster than *eCos*, there is a performance advantage to be gained as well.

Q. **Once a customer has created a design with your technology, is it meant to actually deploy on an FPGA? Or is it meant as an ASIC prototyping system? At the end of the process, what does a designer actually end up with?**

A. Let's look at the scenario where you've chosen to use Eridon's *uCards* for prototyping. At the end of the day (and in this case that means literally within a day of your hardware being delivered), you have a hardware prototype with all of your chosen peripherals, a fully configured RTOS and a complete IDE all ready to go.

You also have the schematics and BOMs for the hardware, so you can immediately start laying out a production board. And when the production board is ready, you can move your software and FPGA design over with minimal effort—the Eridon IDE automatically generates the FPGA configuration image for both your prototype and your final production board. Whether you want to take the additional step of manufacturing a custom ASIC to replace the FPGA depends on the specifics of your application and the economics of your business. The FPGA vendors are experienced in this transition.

Q. **How much does it cost? Please explain the pricing and purchase options?**

A. Our basic pricing model is that you can play with the system for free, for a relatively low investment you can get the development tools and hardware and create a prototype, and ultimately you license the RTOS and IP (gate-level logic in the FPGA) when you go to production.

The free evaluation of the *UnifiedLogic Development Platform* can be downloaded from our site and runs on low-cost evaluation boards offered by Xilinx ($225), Eridon and others. You can even play with it without any hardware.

Otherwise, the *UnifiedLogic Development Platform* (RTOS and tools) starts at $2,900 per seat. Prices for Eridon's prototyping hardware are posted on our website at http://www. eridon. com; you can get into a system for well under $1,000.

The cost of production licensing depends on the subsystems you use. It is generally a one-time payment and royalty-free.

Q. **Thank you for your interview**

MAPUSOFT TECHNOLOGIES: THE BENEFITS OF RTOS PORTING & ABSTRACTION

30 October 2007: The Benefits of RTOS Porting & Abstraction

INTERVIEWEE. RAJ JOHNSON
PRESIDENT & CEO
TEL. 251-665-0280
EMAIL. info@mapusoft.com
COMPANY. MAPUSOFT TECHNOLOGIES, INC.
WEB. http://www.mapusoft.com/

Q. **First of all, tell us a little bit about yourself and your responsibilities at MapuSoft.**

A. After being an embedded software developer myself for over 10 years, I realized that there ought to be a better way to migrate existing code to the next generation. Rewriting code in-house was always tedious and error prone work and it also took my focus away from developing my core product. I founded MapuSoft in 2001 to offer developers an easy way to quickly port existing applications to many different OS platforms. We added more code reusability solutions providing a greater level of development freedom without being tied to an OS provided by a single vendor.

At MapuSoft I am responsible for the direction and development of the corporation. My goal is to ensure that MapuSoft will continue to be the leader in the embedded market offering innovative code reuse products and services which provide tremendous value and satisfaction to embedded developers.

Q. **Mapusoft is not known as an "RTOS company" but rather a company whose products help designers "free" themselves from dependence on a given RTOS. Can you succinctly explain your products?**

A. MapuSoft offers *OS Changer*, *OS Abstractor* and *OS PAL* products to help developers reuse their embedded software on one or more operating systems.

OS Changer gives users the freedom to switch operating systems while leveraging on their existing embedded code and knowledge base to protect their software investment and avoid costly porting issues.

OS Abstractor provides a standard OS interface architecture for portable application development, giving code the ability to adapt to multiple operating system platforms, thereby eliminating OS dependency to protect software investment.

OS PAL (OS Porting and Abstraction Lab) leverages the existing *OS Changer* and *OS Abstractor* technologies while adding advanced code optimization capacities on multiple OS environments. With OS PAL's Eclipse-based GUIs developers can easily port, abstract and optimize code on a host machine and run the application on different target platforms.

MapuSoft also offers porting, integration, support and training services to help developers easily migrate from existing platforms to the next generation.

Q. **What are common scenarios why a developer might want to port from one RTOS to another?**

A. There are many different reasons why developers would need or want to change their operating systems. Some of these reasons include their operating system becoming unsupported or obsolete, their new chip architecture supports a different OS, their customer demands for a different OS, they must now adhere to industry standards, a

change in their technical requirements, a change in their business model, a need to use a better development environment, a need for an operating system that offers a wider range of driver, protocol and middleware support and a need to upgrade their OS to the latest version.

To sum it up, as the times change, so do the technologies. Platforms will be evolving constantly in order to keep up with the demands of the embedded market, hence the need to reuse existing code and develop highly portable code.

Q. **And why might developers be interested in *OS Abstractor*? Doesn't it add another level of complexity to RTOS choice and management?**

A. Actually, *OS Abstractor* simplifies RTOS choice and management by creating a common landscape for all operating systems.

By using a standard OS interface architecture with our *Base OS Abstractor* product; OS choice is simplified by giving developers the ability to reduce the learning curve associated with adopting a new OS by using intuitive, flexible and standard APIs across multiple operating systems.

Code management across operating systems is also simplified because one set of code base is maintained efficiently across multiple operating systems and manual updates when upgrading or changing operating systems are eliminated.

Developers can also use an industry standard *POSIX* API with our *POSIX OS Abstractor* product to reuse or develop *POSIX* code that will run on across many *POSIX* and non-*POSIX* operating systems.

Q. **Is *OS Abstractor* useful as an RTOS evaluation tool? How might this work?**

A. Yes, by developing code with *OS Abstractor*, developers can easily run their code on multiple operating systems to evaluate performance of their application on each operating system.

Developers could also use *OS Changer* to accomplish this with existing legacy code.

By using *OS PAL*, developers have the ability to carry out these evaluations with *OS Changer* and *OS Abstractor* easier by using the code optimization wizard GUIs to produce code optimized for each specific operating system.

Q. **Does *OS Abstractor* support all RTOSes? *Embedded Linux, VxWorks, ThreadX, Neutrino*, etc.? Or just some?**

A. As a result of our partnerships with many OS vendors, *OS Abstractor* has been validated on multiple versions of many prominent operating systems including *VxWorks*, *Linux*, *LynxOS*, *Solaris*, *Unix*, *eCOS*, *Windows XP*, *Windows CE*, *Nucleus*, *ThreadX*, *MQX*, *QNX*, *T-Kernel* and *uITRON*.

OS Abstractor is designed to be easily extended to support new commercial and proprietary operating systems as requested.

Q. **Does *OS Abstractor* have a negative impact on code size? Is it difficult to learn?**

A. All of MapuSoft's products are designed to have very little impact on code size and speed.

In some cases, using *OS Abstractor* will enhance the speed of applications due to our advanced development features including real-time enhancements, a unique process feature to enable independent development of complex and multiple applications and dynamic application reconfiguration and restart. *OS Abstractor* also enhances performance by maximizing the use of compile-time translations and low level functions, using static allocation of control blocks, offering scalability at component and feature levels and providing a task pooling feature to reuse task envelopes. *OS Abstractor* has been tried and tested in many performance critical areas such as medical and mil/aero

applications. The exact size of the footprint of our products varies by target operating system. In some target operating systems it can be as little as 2k.

OS Abstractor's APIs are very simple and intuitive; in fact we have had customers tell us they prefer to use *OS Abstractor* APIs rather than the OS's native APIs because they are simpler and in some cases add more functionality to the OS.

Q. **Why would a developer use *POSIX OS Abstractor* on a POSIX-based OS?**

A. Since *POSIX* offerings, implementations and versions differ from vendor to vendor; *POSIX* code is not really portable. However, *POSIX OS Abstractor* offers true portability across *POSIX* and non-*POSIX* operating systems. In addition, *OS Abstractor* adds performance, OS hardening and reliability features to satisfy embedded application requirements while maintaining *POSIX* compliance.

Q. **What about *OS Changer*? Does it support many or only some RTOSes?**

A. *OS Changer* is available to port existing code from *VxWorks*, *pSOS* and *Nucleus* to different target operating systems. Since *OS Changer* leverages *OS Abstractor* technology, target operating systems supported by OS Changer include the ones supported by *OS Abstractor*. Existing *POSIX* code can also be reused with our *POSIX OS Abstractor* product. A table outlining the target operating systems supported by each of our products can be found here: http://mapusoft.com/products/offerings/

Q. **How much do your products cost? What is your business model?**

A. All of MapuSoft's products are royalty free and come with target source code. Our standalone products, *OS Changer* and *OS Abstractor*, are licensed by one-time fees for developer and target licenses. *OS Changer* developer licenses start at $2,000 and *OS Abstractor* developer licenses start at $1,000. *OS PAL* is licensed by a yearly subscription basis for developer, code generation and target licenses. *OS PAL* developer licenses start at $500.

Compared to the time and cost spent rewriting code, MapuSoft's products are the clear economical alternative. A complete explanation of licensing and pricing can be given by contacting our sales department at sales@mapusoft.com or 251-665-0280.

Q. **Thank you for this interview.**

MICROSOFT: CONNECTED EXPERIENCES BASED ON WINDOWS EMBEDDED

30 November 2007: Connected Experiences based on Microsoft Windows

INTERVIEWEE. MIKE HALL
SOFTWARE ARCHITECT
TEL. 425-707-5223
EMAIL. MIKEHALL@WINDOWS.MICROSOFT.COM
COMPANY. MICROSOFT
WEB. HTTP://WWW.MICROSOFT.COM/EMBEDDED

Q. **First of all, tell us a little bit about yourself and your role at Microsoft.**

A. My name is Mike Hall and I'm a Software Architect for Windows Embedded Business at Microsoft, working with *Windows Embedded CE* and *Windows XP Embedded*. I've been working at Microsoft for over 10 years – originally in Developer Support, focusing on C/C++, MFC, COM, device driver development, Win32, MASM, and then as a systems engineer in the Embedded Devices Group. I'm also responsible for many of the technical resources available on *Windows Embedded CE* on MSDN. Founded in 1975, Microsoft (NASDAQ "MSFT") is the worldwide leader in software, services and solutions that helps people and business realize their full potential.

Q. **There are a lot of really exciting things going on at Microsoft for embedded and real-time systems (so we will ask about those below), but first can you give us just a quick "bird's eye" view of Microsoft's main products for embedded systems - *Windows XP Embedded, Windows Embedded CE,* and *Windows Embedded for Point of Service*?**

A. *Windows Embedded CE* and *Windows XP Embedded* are both componentized operating systems, both expose similar programming interfaces (Win32, MFC, ATL, and support for managed .NET applications), both expose similar operating system technologies, which include support for networking, Internet browsers, media players, and so on.

Windows Embedded CE has been designed to be a small footprint, componentized, hard real-time operating system that runs on multiple processor architectures. The open, scalable, 32-bit operating system (OS) allows developers to build a wide range of innovative, small footprint devices. A typical *Windows Embedded CE*–based device is typically designed for a specific use, often runs disconnected from other computers, and requires a small footprint OS that has a deterministic response to interrupts. Examples include industrial controllers, communications hubs, point-of-sale terminals, and consumer products, such as cameras, Internet appliances, and interactive televisions.

Windows XP Embedded can be considered to be a componentized version of Windows XP Professional Service Pack 2 with additional embedded enabling features. Embedded enabling features include the ability to boot the embedded operating system from USB media, or from read-only media such as CD-ROM or Flash media – in these cases you may want to protect the underlying media so that all write operations are re-directed into an in-memory cache, this can be achieved through the use of the File Based Write Filter (FBWF) or Enhanced Write Filter (EWF).

Based on *Windows XP Embedded, Windows Embedded for Point of Service* is Microsoft's first operating system specifically developed for vertical markets. Microsoft is committed to providing an operating system and application environment that delivers the functionality that retailers demand for point of service (POS) systems. Additionally, the product's design enables partner companies to cost-effectively build and market their POS systems based on the *Windows Embedded for Point of Service* platform.

Q. **What about *Vista* for embedded systems? I understand there is actually a version of *Vista* for embedded. How does this work? What applications does it target?**

A. Yes, there is – *Windows Vista for Embedded Systems* is an operating system appropriate for dedicated embedded devices that require *Windows Vista* application compatibility and flexibility to: deploy a custom user interface with familiar management technologies that make it easy to set up, use, and manage standalone or within an enterprise environment

Working with *Windows Vista for Embedded Systems* can help you shorten your development cycle through compatibility with thousands of production quality off-the-shelf applications, services and certified drivers. It supports the most commonly used PC architecture processor families giving you the flexibility to easily migrate customized operating systems and applications.

Unlike the other *Windows Embedded* operating systems, *Windows Vista for Embedded Systems* is not a componentized offering and the image is not customizable by the developer. *Windows Vista for Embedded System* was designed for the desktop but is available for fixed-function or dedicated embedded systems.

The *Windows Vista® for Embedded Systems* product line offers two choices:

Windows Vista® Ultimate for Embedded Systems

Windows Vista® Business for Embedded Systems

Windows Vista Business for Embedded Systems operating system is designed to meet the needs of device makers of all sizes. It provides dramatic new infrastructure improvements, enabling device makers to spend less time focused on the day-to-day maintenance of embedded devices and more time adding value in area such as applications.

Windows Vista Business for Embedded Systems also includes new technology and tools to help ensure your embedded devices are always up-to-date, more secure, and running smoothly. For instance, *Windows Vista Business for Embedded Systems* will help make your embedded devices safer with built-in protection against malicious software. You will be warned of impending hardware failures long before you risk losing any important data. An array of sophisticated new backup technologies helps protect your information even in the event of a catastrophic hardware failure. *Windows Vista Business for Embedded Systems* has been designed from the ground up to improve the deployment and management of the operating system.

Windows Vista® Ultimate for Embedded Systems

Windows Vista Ultimate for Embedded Systems is the first operating system that combines the advanced infrastructure of a business-focused operating system and the digital entertainment features of a consumer-focused operating system. It also includes *Windows BitLocker*™ Drive Encryption that provides new levels of protection against theft of important business data stored on the embedded device. With access to Multi-Lingual UI and availability of all worldwide language packs supporting embedded devices with multilingual requirements, device makers can install multiple languages and switch between them as required.

Windows Vista Ultimate for Embedded Systems helps reduce application migration and compatibility costs with Subsystem for UNIX-Based Applications (SUA) making your Windows-based embedded device more versatile. SUA enables you to run UNIX-based applications on a *Windows Vista Ultimate for Embedded Systems*-based embedded device. It includes *Virtual PC Express* providing a safety net for operating system migration by making it possible to run a legacy operating system and legacy applications at the same time that you are running *Windows Vista*.

Q. **At the Fall Embedded Systems Conference, Kevin Dallas (General Manager, Windows Embedded) spoke on the theme of "connected experiences." What (briefly) is Microsoft's vision of "connected experiences?" How does this relate to embedded designs?**

A. The Windows Embedded vision revolves around the "Connected Experience." *Windows Embedded* is driving the next generation of helping to create leading edge consumer and enterprise experiences. With built in networking and communications infrastructure, Windows Embedded helps devices easily connect to other devices, PCs, servers and services. Further, Windows Embedded is at the forefront of innovation in the embedded space. From humble beginnings, Windows Embedded has become the number one commercial embedded software in the world. It has defined a generation of small-footprint devices for consumer electronics, healthcare, manufacturing, and more.

Q. **Can you identify a few "application examples" that bring to life this vision of "connected experiences?" i.e., some "food for thought" for the developer / reader as to ways in which he or she might "connect" their device and thereby make it a better application.**

A. Sure – there are a number of "Connected Experience" scenarios. First, let me start off with communications. Today if you receive a call in your home, that call will usually be on a landline. If you go out into the Wide Area Network, you'd have to hang up, call the person back on your mobile phone. Then if you get into your corporate headquarters, you'll normally have to hang up again, pick up a separate phone, your IPBX phone, and then return that call. You're literally on three separate phones. With 'connected experiences', what you'll see is the ability to receive that call in your home on your mobile device that knows it's sitting on a Local Area Network, and will use VoIP for that communication. Let's say then that you need to begin traveling to the office, so you move toward your car and switch seamlessly to a Wide Area Network. As you get into your car, the hands-free capabilities instantly switch in. As you continue that call and you get to the office, you can expect that call to then seamlessly switch over to your IPBX, with your mobile phone recognizing that you're now on your corporate network. As a user, this is a very seamless, connected experience that went straight from the home environment to the mobile environment to the corporate environment.

On the management side, if the manufacturer were able to connect to a consumer device, whether it be a set-top box or digital video recorder, they would be able to proactively monitor that device, determine when inputs are needed, and transparently send those updates to that device, as opposed to having the consumer see a problem and then place that call. Another opportunity is to remotely monitor the device so the manufacturer knows when to replenish a device. For example, in the enterprise space, the manufacturer could see when toner needed to be replaced in a multi-function printer, and deliver the ink before the customer asked. This is a service enabled by the 'Connected Experience' that an OEM could actually provide to an enterprise customer.

Another area is usage - Imagine a scenario where the end customer gives the manufacturer the ability or the permission to monitor how that device is used. Two opportunities then are put forth to the OEM or service provider: First, it can monitor how the device is being used, and as a result present new services to the customer, which will drive an even improved service to them as well as a revenue opportunity.

Another benefit for OEMs or service providers is determining a device's location, which is also important information for an enterprise. In the future, when devices are shipped with service-oriented capabilities, OEMs and service providers will be able to determine whether they are capable of displaying advertisements, and if so, be able to publish advertisements on that device. Of course, the user would subscribe to this capability, but it would give the OEM or the service provider an opportunity for an improved service and a new business model.

Q. **What are the features in the Microsoft embedded products that give you a competitive edge over other solutions (e.g., *Linux*, traditional RTOSes) to make this vision of "connected devices" a reality?**

A. Actually, Microsoft offers a shared success model that translates to low up front investments for device makers and faster time to market – this means that *Windows Embedded* is actually a more cost-effective alternative to embedded *Linux*. Windows Embedded designs, on average, cost less to bring to market than embedded *Linux* designs.

Microsoft embraces shared source, with millions of lines of source code broadly available to customers, partners, developers, governments, academics and other interested individuals. For example, in November Microsoft reaffirmed its commitment to hobbyist developers and academics with the launch of its "*SPARK*" Initiative, a partnership between Microsoft and select hardware vendors to bring a complete package offering of hardware and software to these two groups through a simple and affordable engagement model. Microsoft provides the most critical code to get the job done – all the while allowing developers to keep their IP. In addition, developers pay no runtime fees for either *Windows Embedded CE* or *Windows XP Embedded* until devices ship and receive free, downloadable versions of the complete product and toolset available for a 120-day evaluation period.

With these investments, Microsoft's Windows Embedded portfolio of operating systems ranks #1 in both total revenue and worldwide embedded/real-time operating system shipments (VDC). As the #1 commercial embedded software in the world, Windows Embedded CE has not only defined a generation of small-footprint devices (consumer, medical, manufacturing, etc.), but is at the forefront of driving the ultimate connected experience.

Q. **Would you explain the notion of "web services for devices?" What is a web service for an embedded device? What aspects of Microsoft *Windows XPe* or *CE* make it easier for a developer to deploy a Web service in his device?**

A. With the launch of *Windows Embedded CE 6.0 R2* on November 14, Microsoft has added support for Web Services on Devices, which is a methodology for detecting network attached devices and the services they provide. *WSD* is a Microsoft implementation of the "Devices Profile for Web Services" standard. Web Services on Devices provide a method for discovery to take place between new devices plugged into a network and the devices currently on the network. As a device is attached to the network it broadcasts a "hello" message to announce its availability. Other devices and Vista-based PCs can detect these hello packets and then question the new device to discover what services it provides.

The *Windows Embedded CE* implementation of *WSD* allows a device to expose services as well as detect and consume services on other devices using *WSD*. The *Windows Embedded CE* implementation of *WSD* provides the complete Web Services on Devices API *(WSDAPI)* that is provided on *Windows Vista*. The web services on *Windows Embedded CE* are still limited to native (not managed) code and currently there is no *WSD* client for *Windows XP* based PCs. However, this is a promising technology that opens the door to some interesting devices.

By helping to enable Windows Rally technologies such as web services on devices in *Windows Embedded CE 6.0 R2*, OEMs can provide richer experiences and product differentiation when their devices connect virtually to a PC over the network and yet behave as if they were directly connected.

Q. **Other real-time operating systems are "Royalty Free," meaning I pay an up-front cost only. Yet both *Windows Embedded CE* and *Windows XP Embedded* charge run-time royalties. Doesn't this make them more expensive? Can you explain the royalty model for Windows?**

A. Microsoft uses a royalty-based licensing model, which is dependant on a rapid and large-scale deployment of working devices. The alternative, a royalty-free tools and services model, is prohibitively expensive for the original equipment manufacturer (OEM) because the total costs will depend on how many types of devices the OEM is working on (it may need new tools and service contracts) and on how long the OEM takes to bring a product to market.

Microsoft has aligned its business model with operators and OEMs. The faster the product is developed, the less the OEM spends on development; Microsoft, the OEM, and the operator can then collect revenues faster.

Q. **Thank you for this interview.**

Zeidman Technologies: Synthesize your Own RTOS

30 October 2007: Synthesize your Own RTOS

INTERVIEWEE. **BOB ZEIDMAN,** PRESIDENT
TEL. (408) 741-5809
EMAIL. BOB@ZEIDMAN.BIZ
COMPANY. ZEIDMAN TECHNOLOGIES
WEB. *http://www.zeidman.biz/*

Q. **First of all, tell us a little bit about yourself and your responsibilities at Zeidman Technologies.**

A. I am the founder and president of Zeidman Technologies. I guess I'm the "visionary" behind the technology of "software synthesis" and its application to embedded systems and real-time operating systems. I also manage the engineers writing the code and handle sales and business issues as they arise. I've been designing hardware for about 25 years and software for even longer (it was a hobby/obsession in high school). I have a bachelor's degree in physics and another in EE from Cornell and a master's degree in EE from Stanford.

Q. **Zeidman's *SynthOS* would certainly make our list of most intriguing concepts for the RTOS market. Can you summarize what this product is at a conceptual level?**

A. *SynthOS* uses patented algorithms to apply the concept of hardware synthesis to software. Users write application tasks in C, adding a few statements (called *SynthOS "primitives"*) to describe inter-task communication. A simple configuration file specifies operating system parameters like the scheduling algorithm and relative task periods and priorities. *SynthOS* then automatically generates C code for an optimized RTOS including all necessary data structures while eliminating many potential deadlocks and race conditions. The generated code is royalty-free. Users don't need to code semaphores, mutexes, mailboxes, or message queues. *SynthOS* works with any processor that has a C compiler, and most debuggers, interpreters, and emulators are also compatible. *SynthOS* combines the advantages of proprietary and open-source RTOSes, resulting in a smaller footprint, lower task latencies, and faster development time. *SynthOS* squeezes performance out of low-cost, low-power processors that otherwise could not support a real-time operating system.

Q. **It seems to promise code portability, device-independence, and a custom fit for each application. It's one of those approaches that are so obvious in its benefits, the question becomes: if it's so great, why are the other companies (Wind River, Green Hills, MontaVista) focusing on a more "traditional" route to creating RTOSes?**

A. That's a really good question. I believe that the other RTOS companies are focusing where the market wants them to focus. Even technology companies are slow to adopt new technology, especially if it means a new development methodology. That's a risk – switching to new tools and new languages that they're afraid might not be around long. So technology companies ask for the old technology but faster, smaller, better. We've found it difficult to get early adopters to cross the so-called "chasm." However, research in computer science departments at major universities is starting to catch up with us. Papers are being written and research projects are being undertaken on software synthesis. Zeidman Technologies is sponsoring one of those projects at the University of Washington. We're looking forward to training students in the technology so that they'll

request it when they go out into industry. I've learned to be patient. Some of my ideas have not had commercial success until 5 to 10 years after I developed them.

Q. **At the end of the process what does a developer end up with? A true full-fledged RTOS?**

A. In the traditional sense of an RTOS, yes. It's a flexible, real-time scheduler, and all tasks in the system communicate as required. These days many operating systems in devices are called an RTOS, but they're not real time. Similarly many so-called embedded systems are not really embedded systems. Real-time operating systems have strict timing requirements. Embedded systems are embedded in a product before shipment and are not changed after that. Devices like cell phones, though, are really like desktop systems because there are not strict timing constraints and users can download new applications. Those kinds of systems are not ideal for *SynthOS*. Our market is really the true embedded system where the user is not aware that there's a processor. In those systems, *SynthOS* creates a tiny operating system and can get real-time performance out of small, inexpensive, low-power processors that are not thought to be able to support an RTOS.

Q. **How does the generated RTOS compare with commercial RTOS solutions such as *VxWorks*, *RTXC*, or *ThreadX* in terms of footprint, real-time capabilities, etc.?**

A. For the kinds of systems I just described, *SynthOS* will typically generate a much smaller footprint and lower latency RTOS. The other companies have been focusing on the high end of the market with operating systems for complex devices. But there's a huge market for the true embedded devices and *SynthOS* will create the smallest, fastest, cheapest RTOS possible because it does it using optimizing algorithms, not fixed-size modules.

Q. **Many of the benefits of a standard RTOS - especially embedded Linux - are not in the RTOS itself but in the wide range of applications and peripheral middleware (e.g., TCP/IP stack, security, Web browsers, etc.) that can be "added on." Using your product, isn't the developer all "by himself" without any RTOS-specific application software or middleware?**

A. Another very good question. The fact that there is not a library of available drivers has been an impediment to acceptance of *SynthOS*. For systems on a chip and other custom systems where device drivers need to be written from scratch, writing a *SynthOS*-usable driver is much faster than writing a driver for any other RTOS. We can create drivers for our customers as part of a development contract, and this has worked out well to minimize the customer's perceived risk. Also, we have in the works a tool that will automatically convert standard drivers to *SynthOS*-compatible drivers so we can leverage much of the driver code developed for other RTOSes.

Q. **Many development managers require source code in order to safeguard against the business risk of their RTOS partner exiting the market. How does Zeidman Technologies address this concern?**

A. This is where *SynthOS* is even better than open source alternatives like *Linux*. *SynthOS* generates source code and the customer has full ownership of the source code. The source code generated by *SynthOS* is ANSI C, so even if *SynthOS* is eventually dropped from the development process, the customer keeps the synthesized C code. This is a better solution than open source for two main reasons. First, the code is proprietary to the customer, not shared or limited by the GNU General Public License (GPL). Second, the code was not developed by hundreds of unknown programmers but is directly traced back to the customer's programmers. It can be fully documented and is much easier to debug.

Q. **How many customers do you have already? Are there application examples or customer stories that a potential developer can turn to?**

A. We don't release customer data. We're still a fairly small company. We're writing an article for publication by the end of the year that will explain a real example of using *SynthOS*. There are also white papers and user's guides available on our website.

Q. **What does *SynthOS* cost? What are your business models for engagement?**

A. At this time we work with companies to decide whether *SynthOS* is the correct solution for them and then determine a fee. We also make our development services available to them so that they have *SynthOS* experts ready to tackle their problems and lessen their perceived risks. I welcome any company that is interested in our technology to call me to discuss how it can help them speed development time and save costs.

Q. **Thank you for this interview.**

Appendix A: New RTOS Products

In this Appendix, we identify "new products" released in the last six months relating to RTOSes and Linux. For up-to-date information, be sure to go to h t t p : / / w w w . eg3. c o m /rtos and sign up for e-clips (h t t p : / / w w w . eg3. c o m /eclips/).

new product

Green Hills Software Announces Support for AMCC PowerPC 405EX and 405EXr

Green Hills Software, Inc., the technology leader in device software optimization (DSO) and real-time operating systems (RTOS), today announced the availability of its software development solution targeting Applied Micro Circuit Corporation's (AMCC) [NASDAQ:AMCC, www.amcc. c o m] PowerPC 405EX and 405EXr embedded processors. Components of the Green Hills Software development toolkit include: optimizing C/C++ compilers for Power Architecture,...

Web. h t t p : / / w w w . ghs. c o m /news/20071210_amcc_support_405ex_405exr.html
Score. 4
Date. 12/10/2007

new product

Express Logic Introduces ThreadX® RTOS Support for Tensilica's New Diamond Standard 106Micro

Express Logic, Inc., the worldwide leader in royalty-free real-time operating systems (RTOS), today announced ThreadX RTOS and middleware support for Tensilica's new Diamond Standard 106Micro 32-bit microcontroller core. With the addition of support for the new 106Micro, Express Logic continues to provide ThreadX for the entire line of Tensilica's Xtensa configurable cores and its Diamond Standard series of preconfigured cores. ThreadX is Express

Web. h t t p : / / w w w . rtos. c o m /news/detail/?prid=133
Score. 5
Date. 12/5/2007

new product

QNX Aviage Multimedia Suite Cuts Design Time of In-car Infotainment Systems

Combining the sophistication of a full-featured media player for the car with a highly intuitive user experience, QNX Software Systems today announced the commercial availability of the QNX® Aviage® multimedia suite. This automotive-grade middleware solution allows infotainment designers to integrate consumer devices such as iPods into the vehicle, while ensuring the end user experience is simplified and seamless. The new QNX Aviage...

Web. h t t p : / / w w w . qnx. c o m /news/pr_2675_1.html
Score. 4
Date. 12/5/2007

new product

Express Logic Introduces ThreadX RTOS Support for Tensilica's New Diamond Standard 106Micro

Express Logic, Inc., the worldwide leader in royalty-free real-time operating systems (RTOS), today announced ThreadX RTOS and middleware support for Tensilica's new Diamond Standard 106Micro 32-bit microcontroller core. ThreadX supports the entire line of Tensilica's Xtensa configurable cores, and its Diamond Standard series of preconfigured cores. The addition of the 106Micro continues ThreadX support for the complete Tensilica processor...

Web. h t t p : / / w w w . tensilica. c o m /news_events/pr_2007_12_3.htm
Score. 5
Date. 12/4/2007

new product

SYSGO releases 4.2 version of its ELinOS Industrial Grade Linux product

SYSGO, leading supplier of software solutions for the most demanding safety and security applications, announced today the release of ELinOS 4.2, its Embedded Linux environment especially designed for industrial applications. Launched in 1998, ELinOS was the first industrial grade embedded Linux solution. By assisting early adopters in their projects, especially at the beginning in the industrial automation sector, SYSGO acquired an...

Web. http://www.sysgo.com/news-events/press-releases/article/51/sysgo-releases-42-version-of-its-elinos-industrial-grade-linux-product/
Score. 4
Date. 12/3/2007

new product

OKL4 Now Supports ARMv6 Architecture

Open Kernel Labs, a global provider of embedded systems software and virtualization technology, and part of the ARM® Connected Community, announces that its flagship microkernel OKL4 now supports the ARMv6 architecture. ARM technology is at the heart of the world's leading-edge mobile communications devices. For demanding embedded-systems applications, such as in mobile phones requiring Digital Rights Management (DRM), the OKL4...

Web. http://www.ok-labs.com/company/press_releases/ok_labs_support_armv6_achitecture
Score. 4
Date. 11/30/2007

new product

Quadros Systems RTXC RTOS Supports IAR Development Tools for ColdFire Processors

Quadros Systems, Inc., a leading supplier of configurable and scalable real-time operating systems (RTOS), today announced support for IAR Embedded Workbench for ColdFire microcontrollers, an integrated environment for building and debugging applications. Now, developers looking to move to the ColdFire family of processors can access the flexibility and power of the RTXC Quadros RTOS and the efficiency and code density of the IAR Embedded...

Web. http://www.quadros.com/news-and-events/press-releases/2007-11-19/
Score. 4
Date. 11/30/2007

new product

Green Hills Software Announces Software Development Solution for AMCC Power Architecture 405EZ

Green Hills Software, Inc., the technology leader in device software optimization (DSO) and real-time operating systems (RTOS), has announced the immediate availability of its comprehensive software development solution targeting Applied Micro Circuit Corporation's Power Architecture 405EZ embedded processor. Components of the Green Hills Software development toolkit include: royalty-free RTOSes, optimizing C/C++ compilers for Power...

Web. http://www.ghs.com/news/20071127_amcc_power_archi_405ez.html
Score. 5
Date. 11/27/2007

new product

Quadros Systems RTXC RTOS Supports IAR Development Tools for ColdFire© Processors

Quadros Systems, Inc., a leading supplier of configurable and scalable real-time operating systems (RTOS), today announced support for IAR Embedded Workbench® for ColdFire microcontrollers, an integrated environment for building and debugging applications. Now, developers looking to move to the ColdFire family of processors can access the flexibility and power of the RTXC Quadros RTOS and the efficiency and code density of the IAR Embedded...

Web. http://www.prweb.com/releases/2007/11/prweb570686.htm
Score. 4
Date. 11/26/2007

new product

ADvantage Version 8.2 Preview Now Available

In anticipation of its upcoming January 2008 release, Applied Dynamics International (ADI) has provided a preview of new features and capabilities in ADvantage version 8.2 that can be viewed on the Applied Dynamics website. "ADvantage version 8 was a major re-architecture several years in the making." Said Scott James, Applied Dynamics Chief Executive. "ADvantage version 8.2 has taken every advantage of the new architecture to deliver a...

Web. http://www.adi.com/pdfs/pressreleases/07_11_14.pdf
Score. 4
Date. 11/23/2007

new product

LynuxWorks' Open-source RTOS Supports new Xilinx MicroBlaze Embedded Processor

Continuing its tradition of supporting the latest processors available to embedded developers, LynuxWorks™, Inc., a world leader in embedded software, announced that its BlueCat-ME Linux® fully supports version 7 of the MicroBlaze™ embedded processor launched today by Xilinx, Inc., with the release of its Embedded Development Kit v9.2. The addition of an optional Memory Management Unit (MMU) means the MicroBlaze embedded processor now...

Web. http://www.lynuxworks.com/corporate/press/2007/microblaze.php
Score. 4
Date. 11/23/2007

new product

National Instruments and QNX Software Systems Deliver Measurement-Quality Analog I/O to OEMs

National Instruments and QNX Software Systems today announced an OEM evaluation bundle of software and hardware that reduces the barrier to entry for engineers and scientists using graphical system design to create embedded devices that require measurement-quality analog I/O. The new bundle features a low-cost USB data acquisition device from NI, a single-board computer and drivers as well as evaluation versions of the QNX® Momentics®...

Web. http://digital.ni.com/worldwide/bwcontent.nsf/web/all/3F616BE3EF5D2CEA8625738D00645E7C
Score. 4
Date. 11/22/2007

new product

MontaVista® Linux® Carrier Grade Edition New 5.0 Version Announced

MontaVista® Software, Inc. the leading provider of Linux® for intelligent devices and communications infrastructure, today announced its new 5.0 release of MontaVista Linux Carrier Grade Edition (CGE), the carrier grade Linux operating system preferred by 7 of the top 8 network equipment providers (NEPs). New MontaVista Linux CGE 5.0 provides telecom carriers with new scalability features to handle unexpected jumps in protocol traffic, new...

Web. http://www.mvista.com/press_release_detail.php?fid=news/2007/cge50.html
Score. 5
Date. 11/16/2007

new product

New Capabilities for Windows Embedded CE to Power Smart, Connected, Service-Oriented Devices

Microsoft Corp. announced the release of its Windows Embedded CE 6.0 R2 offering, designed for developers and device-makers wanting to quickly build smart, connected, hard real-time commercial and consumer electronics devices. With new features, including the Web Services on Devices API, voice over Internet protocol (VoIP) telephony capabilities and thin-client technology enhancements, Windows Embedded CE 6.0 R2 enables devices to seamlessly...

Web. http://www.microsoft.com/Presspass/press/2007/nov07/11-13WER2InnovationsPR.mspx
Score. 5
Date. 11/16/2007

new product

Acceleration Kit for Microsoft Windows Embedded CE 6.0

PHYTEC and Adeneo announce a new low cost Acceleration Kit for Microsoft Windows Embedded CE 6.0. The Acceleration Kit includes the ARM-core based phyCORE-PXA270 module and carrier board; LCD with integrated touch panel; Windows Embedded CE 6.0 Board Support Package (BSP) source code access; and all the contents required to enable users to successfully set-up target hardware, build and load a Windows Embedded CE image with Platform Builder,...

Web. http://www.phytec.com/news/news.html?m=detailed&detailed=74
Score. 4
Date. 11/7/2007

new product

Concurrent Announces Availability of its SIMulation Workbench Linux-based, Real-Time Modeling Tools

Concurrent Computer Corporation (NASDAQ:CCUR), a leading provider of time-critical Linux® operating systems and integrated software and computer solutions for mission-critical applications, today announced that its SIMulation Workbench modeling environment is now generally available on its popular iHawk real-time Linux platforms. SIMulation Workbench provides a complete framework to develop and execute real-time hardware-in-the-loop and...

Web. http://news.ccur.com/corp_news_pressrelease.asp?pressreleaseid=533
Score. 4
Date. 11/2/2007

new product

Express Logic Unveils ThreadX® RTOS for Microchip Technology's New PIC32 32-bit ...

Express Logic, Inc., the worldwide leader in royalty-free real-time operating systems (RTOS), today announced the availability of its ThreadX® RTOS for the new PIC32 microcontroller family from Microchip Technology Inc. (NASDAQ: MCHP), a leading provider of microcontroller and analog semiconductors. In August, Express Logic announced its ThreadX/MCU Edition for Microchip's 16-bit PIC24 and dsPIC33 controllers. With this addition, ThreadX...

Web. http://rtos.com/staff/PRPreview.asp?prid=132
Score. 5
Date. 11/2/2007

new product

Wind River Announces Symmetric Multiprocessing Support for VxWorks Real-Time Operating System

Wind River Systems, Inc. (NASDAQ: WIND), the global leader in Device Software Optimization (DSO), today announced VxWorks® 6.6 with Symmetric Multiprocessing (SMP) support will be available on all Wind River VxWorks-based platforms in late 2007. Already the most widely used and thoroughly tested commercial real-time operating system (OS) in the market today, VxWorks 6.6 SMP support for multicore processors will further empower developers to...

Web. http://www.windriver.com/news/press/pr.html?ID=5241
Score. 5
Date. 11/2/2007

new product

Enea Announces dSPEED Platform

Enea (Nordic Exchange/Small Cap/ENEA), a leading provider of network software and services, today announced the availability of dSPEED Platform, the first commercial high availability platform designed to manage clusters of Digital Signal Processors (DSPs). High performance digital signal processing is critical to delivering the computationally intense applications and services that consumer's demand. In network equipment categories...

Web. http://www.enea.com/Templates/NewsPage____23124.aspx
Score. 5
Date. 10/30/2007

new product

DDC-I Announces Eclipse-Based Mixed Language Development Suite for Real Time ...

DDC-I, a leading supplier of development tools for safety-critical applications, announced the first Eclipse-based mixed-language development and run-time environment to integrate C, Embedded C++, Ada, and real-time Java. Known as OpenArbor, the new IDE makes it possible to develop hard real-time applications that combine Java, C, EC++, and Ada. "OpenArbor is the only Eclipse-based IDE that supports true mixed language C, Embedded...

Web. http://www.ddci.com/display_news_item-filename-news_Eclipse-BasedMixedLanguage_release.htm
Score. 5
Date. 10/25/2007

new product

QNX Harnesses Adobe Flash Lite 3 to Revolutionize Embedded User Interface Design

QNX Software Systems announced the availability of a new design solution that speeds the creation and performance of graphical user interfaces for embedded products. Combined with the power of Adobe Flash Lite 3 software, the new QNX® Aviage® graphics suite allows software designers to create and implement entire user interfaces in Adobe Flash, while leveraging the rich multimedia capabilities and exceptional reliability of the QNX...

Web. http://www.qnx.com/news/pr_2609_1.html
Score. 4
Date. 10/25/2007

new product

Symbian Drives Developer Innovation with Improved Symbian Signed ...
Symbian Limited, developer and licensor of Symbian OS the market-leading operating system for advanced data-enabled mobile phones known as smartphones, today announced its ongoing efforts to make developing easier and more economically viable for developers worldwide. Symbian Signed made faster, easier

Web. http://www.symbian.com/news/pr/2007/pr20079454.html
Score. 4
Date. 10/25/2007

new product

ACCESS Develops New Server Solution, NetFront Transcoder
ACCESS CO., LTD., a global provider of advanced software technologies to the mobile and beyond-PC markets, announced NetFront Transcoder, a new Web-to-mobile content adaptation server technology that enables end users to browse any Web site from any mobile device. NetFront Transcoder, based on ACCESS' award-winning NetFront Browser engine for mobile handsets and information appliances, will be marketed worldwide. The full-featured...

Web. http://www.access-company.com/news/press/ACCESS/2007/20071016_transcoder.html
Score. 4
Date. 10/24/2007

new product

Wind River Releases Comprehensive Device Software Platform and Networking Stack for ARINC 653
Wind River Systems, Inc. (NASDAQ:WIND), the global leader in Device Software Optimization (DSO), today announced the release of VxWorks® 653 Platform 2.2, its enhanced ARINC 653 IMA platform for safety-critical systems. The new version of VxWorks 653 underscores Wind River's commitment to providing the avionics market with a proven, high-performance, low-risk ARINC 653 operating system, with complete DO-178B Level A certification artifacts...

Web. http://www.windriver.com/news/press/pr.html?ID=5202
Score. 4
Date. 10/23/2007

new product

Wind River and Freescale Deliver Linux® OS-based Enablement Solution ...
Wind River Systems, Inc. (NASDAQ:WIND), the global leader in Device Software Optimization (DSO), and Freescale Semiconductor have introduced an enablement platform that addresses growth opportunities in the embedded mass market. The joint platform is designed to simplify and accelerate application development for embedded

computing applications using Freescale's multi-core MPC5121e processor and the Wind River Linux operating system (OS). ...

Web. http://www.windriver.com/news/press/pr.html?ID=5181
Score. 4
Date. 10/19/2007

new product

IAR Systems announces IAR PowerPac TCP/IP for ARM

IAR Systems announces the imminent launch of a TCP/IP stack that works with IAR PowerPac RTOS and file system to offer the benefits of an easy-to-use TCP/IP interface to developers using IAR Embedded Workbench® for ARM®. This new addition to the company's integrated middleware portfolio is particularly suitable for use in portable devices that need a convenient connection to a computer network such as the Internet.With the TCP/IP stack...

Web. http://www.iar.com/index.php?show=414_ENG&article=190&page_anchor=http://www.iar.com/p414/p414_eng.php?article=190
Score. 5
Date. 10/18/2007

new product

IEC 61131-3 Runtime System and Development Environment with CANopen Support

Based on the cooperation between IXXAT and KW-Software, the IEC 61131-3 compliant runtime environment ProConOS and the develompment environment Multiprog are now offered with CANopen support. Main feature of this solution is a full integration of the CANopen manager (master) into ProConOS and the CANopen configuration tool in Multiprog. In comparison to other solutions, ProConOS and the CANopen manager can be integrated on one CPU with...

Web. http://www.ixxat.de/download/presse/press_2007_cop_auf_sps_(kw-software).doc
Score. 4
Date. 10/17/2007

new product

New MontaVista TestDrive Center Reduces Timeframe for Embedded Linux Evaluations from Months to Days

MontaVista® Software, Inc. the leading provider of Linux® for intelligent devices and communications infrastructure, today released its new MontaVista® TestDrive, a virtual evaluation center which lets embedded Linux developers run their own applications using MontaVista Linux on their choice of processor architectures. Because MontaVista TestDrive is a hosted web service requiring no setup, it can shrink the time required for the technical...

Web. http://www.mvista.com/press_release_detail.php?fid=news/2007/testdrive.html
Score. 5
Date. 10/12/2007

new product

TimeSys Introduces LinuxLink Subscriptions for MIPS 74K Processor Cores

TimeSys, the leading service provider to developers in the embedded Linux market, has announced the availability of LinuxLink subscriptions for the MIPS32® 74KTMfamily of processor cores. This high performance processor core is a new addition to the MIPS®

32-bit line of processor cores. Through the collaboration between MIPS and TimeSys, customers can build their own custom Linux software platforms using the MIPS32 74K core.The combination...

Web. http://www.timesys.com/releases/home_bdy_news.php?show_article=200712
Score. 4
Date. 10/12/2007

new product

Trolltech Releases Technology Preview of Qt for Windows CE

Trolltech® released an initial technology preview of Qt® for Windows® CE (Qt/WinCE) to its commercial customers and the open source community for testing and feedback. Qt/WinCE is a port of the Qt API and developer tools to the Windows CE operating system. Qt/WinCE continues Trolltech's cross-platform

Web. http://trolltech.com/company/newsroom/announcements/press.2007-10-03.3664932915
Score. 5
Date. 10/12/2007

new product

Micrium Adds Simulator to uC/Probe Real Time Monitoring Tool

Micrium has upgraded its award-winning uC/Probe tool that monitors embedded designs in a live environment. Users can now test a good portion of their embedded software and be able to visualize it using uC/Probe's new simulation capability. Further, uC/Probe can serve as a demonstration and marketing tool, as it can demo new products without needing to have hardware connected to the PC. The new version adds TCP/IP and USB connectivity to the...

Web. http://www.micrium.com/news/Micrium_Probe_071009.htm
Score. 5
Date. 10/9/2007

new product

eSOL Extends Support to ARM Cortex-A9 MPCore Processor

eSOL, a leading developer of real-time embedded software solutions, announced today they have reached an agreement with ARM to support their new Cortex-A9 MPCore multicore processor with eSOL's eT-Kernel Multi-Core Edition real-time operating system (RTOS). This, coupled with eBinder integrated tools for enhanced multiprocessor solutions, will optimize the Cortex-A9 MPCore multicore processor designs in the automotive, mobile, enterprise...

Web. http://www.esol.co.jp/english/company/press/emb_press071004.html
Score. 4
Date. 10/4/2007

new product

eSOL Releases Further Blends on Multi-Core RTOS

Following its debut at the ARM Developers' Conference last year, eSOL, a leading developer of realtime embedded solutions, has announced that the enhanced version of its realtime operating system, eT-Kernel Multi-Core Edition, is now available for ARM MPCore multi-core processors. At last year's ARM Developers' Conference, eSOL introduced a one-of-a-kind RTOS that takes advantage of the capability of the ARM11 MPCore multi-core processor...

Web. http://www.esol.co.jp/english/company/press/emb_press071003_eTKMCE.html
Score. 5
Date. 10/4/2007

new product

BSQUARE Demonstrates Rapid Windows CE 6.0 Porting Capability on Marvell® PXA 300...

BSQUARE Corporation, a leading Windows Embedded solution provider to the global embedded device community, announced that it has completed a port of the Windows Embedded CE 6.0 operating system to Marvell's PXA 300 Handheld Platform Development Kit (formerly codenamed Littleton). BSQUARE finished its software port in less than one month, orders of magnitude less time than is typically spent on a Board Support Package...

Web. http://www.bsquare.com/about/story.asp?PressID=465
Score. 4
Date. 10/2/2007

new product

eSOL Releases PrCONNECT/Pro:New high-speed TCP/IP Protocol Stack

eSOL has announced the release of their 'PrCONNECT/Pro' TCP/IP protocol stack for embedded systems with BSD socket interface. The PrCONNECT/Pro is ideal for full-scale networking equipment, as well as handheld information terminals, including multimedia devices, since its performance is over 74Mbps and has a rich set of security, routing, and other protocols. PrCONNECT/Pro offers the...

Web. http://www.esol.co.jp/english/company/press/emb_press070928.html
Score. 4
Date. 10/1/2007

new product

SJJ Embedded Micro Solutions Releases the Embedded Development Kit...

SJJ Embedded Micro Solutions is pleased to announce the release of the Embedded Development Kit (EDK) for the Microsoft .NET Micro Framework. The EDK contains a multipurpose development board with the .NET Micro Framework already installed and ready to run C# applications. The kit contains a step-by-step instruction manual to guide engineers, students, and hobbyists through the process of developing .NET Micro Framework applications with...

Web. http://www.sjjmicro.com/PR_EDK_Release_070918.html
Score. 5
Date. 9/28/2007

new product

Symobi for everybody!

The company Miray Software presents to the public its newly developed embedded RTOS Symobi with an own website and a free live demo. Symobi was introduced in professional circles for the first time on the Systems 2006 exposition. Symobi is a graphical RTOS with a newly designed architecture, according to state-of-the-art technology. It is based on the microkernel system µnOS, which was also developed by Miray Software. Therefore, Symobi...

Web. http://www.miray.de/home/prrel.html#20070913
Score. 4
Date. 9/28/2007

new product

Applied Data Launches Windows CE 6.0 on Embedded Systems Products

Applied Data Systems, a leader in low-power embedded computer systems and a subsidiary of Eurotech Group, S.p.A., announces the availability of Windows CE 6.0 on the BitsyXb systems. With new features to boost efficiency and processing power, Windows CE 6.0 truly expands the benefits of using BitsyXb for a wide range of low-power embedded applications. Windows CE 6.0 was developed from the ground up by Microsoft and offers several...

Web. http://www.applieddata.net/msword/news_pr_070918_Window_CE6.doc
Score. 4
Date. 9/26/2007

new product

New MontaVista® Mobilinux 5.0, the World's Most Advanced Mobile Operating System

MontaVista® Software, Inc. the leading provider of Linux® for intelligent devices and communications infrastructure, today announced the new 5.0 release of MontaVista? Mobilinux, the mobile operating system used in 90 percent of Linux-based smartphones. Mobilinux provides a commercial-quality, field-proven embedded Linux operating system (OS) optimized for mobile devices plus a rich development environment for the engineers who design them....

Web. http://www.mvista.com/press_release_detail.php?fid=news/2007/mob5.html
Score. 5
Date. 9/26/2007

new product

Collaboratively Developed LINUX Developers Toolkit Debuts at Power Architecture Developer Conference

Power.org introduced the Power Architecture LINUX Developer Toolkit and Reference Platform, 970MP Edition, an offering that helps members and their customers lower cost of development and accelerate their product's time to market. Based on the Power Architecture dual-socket 2.5 GHz dual-core 970MP reference platform with 8 DDR2 667 MHz DIMM slots, this offering is available as both a high performance Toolkit to developers, and as a reference...

Web. http://www.power.org/news/pr/view?item_key=e6edc76b12015b289dbc18089d2d7cd556a4f83d
Score. 5
Date. 9/24/2007

new product

MontaVista to Provide Real-Time Linux on Future AMCC Power Architecture Processors

MontaVista® Software, Inc. the leading provider of Linux® for intelligent devices and communications infrastructure, announced today that the company is partnering with chip maker AMCC to provide design teams and developers planning projects on Power Architecture platforms with the ability to leverage the real-time capabilities, small memory footprint, high-performance development tools, and PhD-quality support of MontaVista Linux...

Web. http://www.mvista.com/press_release_detail.php?fid=news/2007/amcc.html
Score. 4
Date. 9/24/2007

new product

Wind River and Freescale introduce multiprocessing solution for the Freescale MPC8572 processor

Wind River Systems, Inc. (NASDAQ:WIND) and Freescale Semiconductor today demonstrated an end-to-end multicore solution optimized for Freescale's MPC8572 dual-core processor at the 2007 Power Architecture Developer Conference, hosted by Power. o r g. Wind River optimized the solution by tailoring its industry-leading VxWorks real-time operating system (RTOS) specifically to support symmetric multiprocessing (SMP). The demonstration, currently...

Web. http://media.freescale.com/phoenix.zhtml?c=196520&p=irol-newsArticle&ID=1054660&highlight=
Score. 5
Date. 9/24/2007

new product

Wind River Unveils Next Generation of Wind River Commercial Grade Linux Platforms

Wind River Systems, Inc. (NASDAQ: WIND), the global leader in Device Software Optimization (DSO), today announced the availability of the next generation of Wind River Linux, the high-performance, commercial-grade Linux distribution optimized for embedded device development. Offering a greater level of sophistication and functionality demanded by the Linux device developer community, the new release of Wind River Linux will provide a...

Web. http://www.windriver.com/news/press/pr.html?ID=5101
Score. 5
Date. 9/24/2007

new product

Aonix Releases Next-Generation Safety-Critical Platform

Aonix®, a provider of solutions for safety- and mission-critical applications, announced the release of ObjectAda RAVEN for Windows targeting the Wind River VxWorks 653 multi-partition RTOS for PowerPC. ObjectAda RAVEN features an enhanced safety-critical Ada runtime that communicates to the underlying RTOS through the ARINC-653 APEX interface available in the Wind River VxWorks 653 environment - enabling high execution and safety...

Web. http://www.aonix.com/pr_09.18.07b.html
Score. 5
Date. 9/21/2007

new product

Microsoft Releases .NET Micro Framework 2.0 Service Pack 1, Announces New Partner Deployments

Microsoft Corp. announced the availability of Service Pack 1 (SP1) for the Microsoft® .NET Micro Framework 2.0, featuring new enhancements to build more secure, attractive, innovative and globalized applications. Demonstrating further momentum, Microsoft also announced that several partners completed ports of the .NET Micro Framework to additional development platforms and released new products built on the .NET Micro Framework. The new...

Web. http://www.microsoft.com/presspass/press/2007/Sep07/09-18ESCBostonPR.mspx
Score. 5
Date. 9/21/2007

new product

Quadros Systems Announces VisualRTXC Design Tool

Addressing the realities of shorter design cycles for systems of ever-increasing complexity, Quadros Systems, Inc., a leading supplier of configurable and scalable real-time operating systems (RTOS), announced the new VisualRTXC design tool. This powerful visual design environment works with the RTXC Quadros RTOS to dramatically reduce development time for embedded systems. VisualRTXC offers a highly intuitive user interface and high-level...

Web. http://www.quadros.com/news-and-events/press-releases/2007-09-18/
Score. 5
Date. 9/21/2007

new product

SYSGO releases version 2.2 of separation kernel PikeOS

SYSGO, leading supplier of software solutions for the most demanding safety and security applications, announced the release of PikeOS 2.2, its powerful and efficient paravirtualization real-time operating system (PVOS) based on a separation microkernel. This version provides improvements aimed more specifically at the aerospace and defense market. Since its introduction in 2005, PikeOS has gained momentum among industry players thanks to...

Web. http://www.sysgo.com/news-events/press-releases/article/51/sysgo-releases-version-22-of-separation-kernel-pikeos/
Score. 4
Date. 9/21/2007

new product

Express Logic Unveils TraceX New Tool for ThreadX® System Event Analysis

Express Logic, Inc., the worldwide leader in royalty-free real-time operating systems (RTOS), announced the release of TraceX, its first host-based embedded development tool. TraceX enables embedded developers to visualize and better understand the behavior of their real-time systems. With TraceX, developers can see clearly the occurrence of system events like interrupts and context switches that occur out of view of standard debugging tools....

Web. http://www.rtos.com/news/detail/?prid=130
Score. 5
Date. 9/20/2007

new product

Mentor Graphics Introduces Multimedia Solution for Nucleus OS and Inflexion Platform

Mentor Graphics Corporation announced the launch of the Inflexion Platform Multimedia Feature Pack for Nucleus® OS, a new paradigm for manufacturers of mass market consumer electronic devices to bring advanced audio and video capabilities to their products. Inflexion Platform UI, launched in December 2006, enables the agile creation and customization of visually compelling, easy to use interfaces for consumer electronic devices such as...

Web. http://www.mentor.com/products/embedded_software/news/multimedianucleusosinflexionplatform.cfm
Score. 5
Date. 9/20/2007

new product

Green Hills Software Announces velOSity Support for Renesas Technology's SH726x SuperH RISC...
Green Hills Software, Inc., the technology leader in device software optimization (DSO) and real-time operating systems (RTOS), announced that the Green Hills product suite is now available for the Renesas SH726x series of SuperH microcontrollers. Included in Green Hills Software's product family are the MULTI integrated development environment, TimeMachine tool suite, Green Hills compilers, DoubleCheck static analysis tools, velOSity...

Web. http://www.ghs.com/news/200700919_velosity_renesas.html
Score. 4
Date. 9/19/2007

new product

eSOL releases complete RTOS suites and IDE for TI's DM355
eSOL releases that its flagship eBinder integrated development environment (IDE) and its PrKERNELv4 real-time operating system (RTOS) suites are now available for Texas Instruments' new TMS320DM355 low-power digital media processor based on DaVinci technology.

Web. http://www.esol.co.jp/english/company/press/emb_press070905.html
Score. 4
Date. 9/13/2007

new product

Green Hills Software's Source Code Analysis Tool Increases Software Quality
Green Hills Software, Inc., the technology leader in device software optimization (DSO) and real-time operating systems (RTOS), and Swell Software, a leading supplier of graphics software for embedded systems, announced that Swell Software has adopted the Green Hills Software DoubleCheck source code analysis tool suite to help ensure the highest level of quality for Swell Software's products.

Web. http://www.ghs.com/news/20070911_swell_doublecheck.html
Score. 5
Date. 9/12/2007

new product

Express Logic Launches ThreadX® MCU Edition for Microchip Technology's PIC24 Microcontrollers..
Express Logic, Inc., the worldwide leader in royalty-free real-time operating systems (RTOS), today announced the availability of its new ThreadX® Microcontroller Edition RTOS, ThreadX®/MCU,for the 16-bit PIC24 microcontrollers (MCUs) and dsPIC® digital signal controllers (DSCs) from Microchip Technology Inc. (NASDAQ: MCHP), a leading provider of microcontroller and analog semiconductors. ThreadX/MCU for 16-bit MCUs and DSCs Until now,...

Web. http://www.rtos.com/news/detail/?prid=127
Score. 5
Date. 9/6/2007

new product

DSPnano© Open Source RTOS Targets Microchip Technology's 16 bit dsPIC® DSCs and PIC24 MCUs
RoweBots Research Inc., a Waterloo supplier of software tools and embedded system consulting, today announced the launch of DSPnano Version 2 for the 16 bit dsPIC® Digital Signal Controller (DSC) and PIC 24 microcontroller (MCU) families from Microchip Technology [NASDAQ: MCHP], a leading supplier of microcontroller and

analog semicondutors. DSPnano is an open source RTOS and Eclipse based tool set which increases small embedded signal...

Web. http://rowebots.com/downloads/pressreleaseopensourcertos.pdf
Score. 4
Date. 8/31/2007

new product

Mistral Introduces Windows® Embedded CE 6.0 Software Development Kit for use with TI's Davinci ...

Mistral Solutions, a leading product realization company specializing in real-time embedded solutions, today announced the availability of its Accelerator Program for Windows® Embedded CE 6.0 for use with Texas Instruments Incorporated's (TI) DaVinci technology. Mistral's Accelerator Program for Windows Embedded CE 6.0 is a complete Software Development Kit (SDK) for TI's TMS320DM6446 Digital Video Evaluation Module (DVEVM) based on...

Web. http://www.mistralsoftware.com/html/news_events/press_releases1.htm#Pr1
Score. 4
Date. 8/23/2007

new product

Nucleus OS and EDGE (Eclipse) Developer Suite for the IBM Cell Broadband Engine

Mentor Graphics Corporation (NASDAQ: MENT) today announced a joint collaboration with IBM on the targeting and porting of Mentor's Nucleus® operating system (OS) and EDGE Developer Suite for the IBM Cell Broadband Engine (Cell/B.E.) processor, initially targeting IBM BladeCenter servers. This marks the availability of the first full-featured real-time OS port for Cell/B.E. with integrated Eclipse tools and MAJIC® multicore JTAG target...

Web. http://www.mentor.com/products/embedded_software/news/nucleusosedgeeclipsesuiteibmcellbroadband.cfm
Score. 4
Date. 8/23/2007

new product

Qt and Qtopia Core 4.3.1 Deliver Bug Fixes and Performance Optimizations

Trolltech released Qt and Qtopia Core 4.3.1 to commercial customers and the open source community. Qt is the standard framework for high-performance cross-platform application development, and Qtopia Core is the leading application framework for single-purpose devices powered by embedded Linux. Both releases are available for evaluation and open source download. Qt and Qtopia Core 4.3.1 include bug fixes and performance optimizations...

Web. http://trolltech.com/company/newsroom/announcements/press.2007-08-07.3596046644
Score. 4
Date. 8/13/2007

new product

Certicom Launches Security Builder API for .NET

Certicom launched Security Builder® API for .NET, enabling developers to easily add elliptic curve algorithms, including those in Suite B, to the Microsoft .NET Framework and .NET Compact Framework applications. Certicom Security Builder API for the .NET Framework also provides backward compatibility. Developers can add elliptic curve and

Suite B algorithms to new and legacy applications running on multiple Windows platforms....

Web. http://www.certicom.com/index.php?action=company,press_archive&view=713
Score. 5
Date. 8/6/2007

new product

Concurrent Announces NightStar LX Tools Version 4.1 for Linux

Concurrent, a leading provider of real-time Linux operating systems and integrated software and computer solutions for mission-critical applications, today announced the release of a new version of its NightStar LX debugging and analysis tools. NightStar is a powerful, integrated Qt-based GUI tool set for developing and tuning time-critical 32-bit and 64-bit applications. NightStar tools reduce test time, increase productivity and lower...

Web. http://news.ccur.com/corp_news_pressrelease.asp?pressreleaseid=519
Score. 4
Date. 8/6/2007

new product

Lineo Delivers Embedded Linux Development Environment ...

Lineo Solutions, Inc. (Lineo, HQ: Nagano, Japan/CEO & President: Kenji Futatsugi), a leading provider of embedded Linux solutions/development tools for consumer electronics devices, announced the release of the latest embedded Linux development environment. 'Following the RBTX4938 version released last month, Lineo uLinux ELITE Board Support Package MIPS Technologies Malta Board/MIPS32® 34K CodeSourcery Edition employs...

Web. http://www.lineo.co.jp/eng/news-events/press-release/20070801.html
Score. 4
Date. 8/6/2007

new product

Ardence Releases Version 2.0 of Select, Includes Support for Windows Vista

Ardence, a Citrix Company, and a leading developer of software platforms for the on-demand world, today announced the release of Version 2.0 of Select, which includes support for the Windows Vista operating system and enables OEMs to deliver Vista-based devices with multi-purpose functionality on a single operating system as well as provide instant-on/instant-off capability. Ardence said Select will reduces OEMs' time, effort and expense...

Web. http://www.ardence.com/embedded/news.aspx?id=1741
Score. 4
Date. 8/1/2007

new product

Wind River Enhances Lab Diagnostics 2.1 and Delivers Innovative Automated Test Capability

Wind River Systems, Inc. (NASDAQ: WIND), the global leader in Device Software Optimization (DSO), today announced the availability of Wind River® Lab Diagnostics 2.1, an updated release of its enterprise-wide software diagnostics solution that enables cross-functional development teams to collaborate on dynamic debugging, testing and quality assurance (QA) throughout the entire development lifecycle. Wind River Lab Diagnostics furthers the...

Web. http://www.windriver.com/news/press/pr.html?ID=4861
Score. 4
Date. 7/25/2007

new product

BSQUARE Launches New Version of Windows CE Board Support Package Development Tool

BSQUARE is now shipping version 6.0 of SchemaBSP, a development tool for creating Windows® Embedded CE Board Support Packages (BSPs) for ARM-based processors. When used with the Microsoft® Platform Builder tool, SchemaBSP enables developers to quickly and easily configure BSP source code for an embedded hardware design,...

Web. http://www.bsquare.com/about/story.asp?PressID=457
Score. 4
Date. 7/24/2007

new product

Green Hills Software Announces µ-velOSity for NEC Electronics ...

Green Hills Software, Inc., the technology leader in device software optimization (DSO) and real-time operating systems (RTOS), announced that its µ-velOSity RTOS is now available for the NEC Electronics V850 family of embedded microcontrollers. With a ROM footprint as small as 2K bytes, RAM footprint as small as 1K bytes, and service call times as low as 30 cycles, µ-velOSity is ideal for the cost-sensitive, high-performance...

Web. http://www.ghs.com/news/200700723_microvelosity_nec_v850.html
Score. 4
Date. 7/23/2007

new product

Altera and eCosCentric Announce eCosPro Support for Nios II Embedded Processor

Altera Corporation and eCosCentric Limited announced the availability of the eCosPro® real-time operating system (RTOS) for Altera's Nios® II embedded processor, the world's most versatile soft-core processor. The combination of the Nios II embedded processor and the eCosPro RTOS meets the needs of a wide range of applications, including industrial automation, consumer appliances, telematics, communications, networking...

Web. http://www.altera.com/corporate/news_room/releases/products/nr-niosiiecos.html
Score. 4
Date. 7/13/2007

new product

Altera and eCosCentric Announce eCosPro Support for Nios II Embedded Processor

Altera Corporation (NASDAQ: ALTR) and eCosCentric Limited, today announced the availability of the eCosPro® real-time operating system (RTOS) for Altera's Nios® II embedded processor, the world's most widely used configurable soft-core processor. The combination of Nios II and eCosPro meet the needs of a wide range of embedded applications including industrial automation, consumer appliances, telematics, communications, networking and...

Web. http://www.ecoscentric.com/news/press-070709.shtml
Score. 4
Date. 7/10/2007

new product

Wind River Enhances On-Chip Device Debugging Solutions
Wind River Systems, Inc. (NASDAQ: WIND), the global leader in Device Software Optimization (DSO), today announced enhancements to Wind River Workbench® 2.6.1On-Chip Debugging Edition, the company's comprehensive device software development toolset for mitigating the complexities of on-chip debugging. The upgraded Wind River Workbench, OCD Edition, provides a development toolset that does not require kernel instrumentation to debug the Linux...

Web. http://www.windriver.com/news/press/pr.html?ID=4381
Score. 4
Date. 7/10/2007

new product

Wind River to Support Freescale Multi-core Communications Platform
Wind River Systems, Inc. (NASDAQ: WIND), the global leader in Device Software Optimization (DSO), today announced its support for Freescale Semiconductor's new Multi-core Communications Platform. Announced today at the Freescale Technology Forum, the platform is an innovative multi-core architecture designed to deliver breakthrough efficiencies, performance and scale, while addressing the emerging challenges of multi-core software...

Web. http://www.windriver.com/news/press/pr.html?ID=4721
Score. 4
Date. 7/10/2007

new product

Wind River Unveils Device Software Optimization Industry's Most Advanced Networking Solution
Wind River Systems, Inc. (NASDAQ: WIND), the global leader in Device Software Optimization (DSO), today announced the immediate availability of the DSO industry's most comprehensive advanced networking technology portfolio, Wind River Advanced Networking Technologies. Leveraging technology from the company's March 2006 acquisition of privately-held Interpeak AB, Wind River Advanced Networking Technologies integrate functionality including...

Web. http://www.windriver.com/news/press/pr.html?ID=4622
Score. 5
Date. 7/10/2007

new product

DDC-I Announces Industry's Highest Performance Java Solution for Hard Real-Time Applications
DDC-I, a leading supplier of development tools for safety-critical applications, today announced the industry's most responsive Java solution for hard real-time applications. Known as Scorpion, the new Eclipse-based tool set delivers two orders of magnitude lower latency than competitive real-time Java solutions. Scorpion is also the first Java solution to support mixed language development, which makes it possible to combine Java with...

Web. http://www.ddci.com/display_news_item-filename-news_Highest_Performance_Java_Solution_for_Hard_Real-Time_Applications_release.htm
Score. 5
Date. 7/9/2007

new product

embOS now supports the STM32

SEGGER Microcontroller Systeme GmbH, one of the leading embedded software manufacturer, today announced that its complete software product tool chain, including embOS, the powerful easy to use RTOS, emWin, the very efficient GUI, emFile, the high performance file system and emUSB, the USB device stack is now ported to the STM32-EVAL board from ST Microelectronics. "SEGGER continues the strong partnership with ST Microelectronics by...

Web. http://www.segger.com/pub/PR/pr_embos_cm3_stm32.pdf
Score. 4
Date. 7/9/2007

new product

Lattice Semiconductor Announces µitron Rtos Support For Latticemico32 Microprocessor

Lattice Semiconductor Corporation (NASDAQ: LSCC) today announced that it has validated the operation of the TOPPERS open source implementation of the uITRON 4.0 Real Time Operating System (RTOS) with its LatticeMico32 32-Bit soft microprocessor. This announcement expands the RTOS options available to users of the LatticeMico32 microprocessor, and is particularly significant because uITRON represents the de facto RTOS for embedded...

Web. http://www.latticesemi.com/corporate/newscenter/productnews/2007/r070618announcesultronrto.cfm
Score. 4
Date. 7/9/2007

new product

MontaVista Furthers Innovation of Power Architecture Technology

MontaVista Software, Inc., a leading provider of Linux for intelligent devices and communications infrastructure, has joined the Power.org community at the sponsoring membership level. As a sponsor, MontaVista will support the mission of Power.org to develop, enable, and promote Power Architecture technology as the preferred open standard hardware development platform for the electronics industry.

Web. http://www.mvista.com/press_release_detail.php?fid=news/2007/power.html
Score. 4
Date. 7/9/2007

new product

MontaVista Provides Linux and Developer Tools for New Freescale Multi-Core

MontaVista Software, Inc., a leading provider of Linux for intelligent devices and communications infrastructure, today announced its support for Freescale Semiconductor's new Multi-Core Communications Platform. MontaVista Linux and MontaVista's developer tools will take advantage of Freescale's innovative multi-core architecture to enable device designers and embedded application developers to create new products that are dramatically...

Web. http://www.mvista.com/press_release_detail.php?fid=news/2007/freescale.html
Score. 4
Date. 7/9/2007

new product

Thales and QNX unveil realtime secure partitioning on PENTXM2 and PENTXM4

Thales and QNX Software Systems today will demonstrate at the Paris Air Show 2007 an innovative hardware and software computing solution which meets the computing performance and the application software security level required in military and aerospace applications. Thales computers activity is a leading provider of VMEbus and CompactPCI single-board computer (SBC) products and systems, and QNX Software Systems is the industry leader in...

Web. http://www.qnx.com/news/pr_2376_1.html
Score. 4
Date. 7/9/2007

new product

VirtualLogix Announces VLX for Network Infrastructure 2.0 supporting Intel

VirtualLogix, the Real-Time Virtualization company, announced VLX for Network Infrastructure 2.0. This newest version of VLX for Network Infrastructure offers a host of new features and capabilities, including enhanced availability and increased OS and platform support, while taking advantage of the latest hardware from Intel, including support for the Intel NetStructure MPCBL0050 Single Board Computer, the first AdvancedTCA* (ATCA) blade...

Web. http://www.virtuallogix.com/index.php?id=62&tx_ttnews[tt_news]=137&tx_ttnews[backPid]=85&cHash=450ce817f7
Score. 4
Date. 7/9/2007

Appendix B: Commercial RTOSes, Broad

In this Appendix, we identify and summarize commercial real-time operating systems that are "broad" in nature – meaning that they generally support many chip architectures and/or target a broad range of vertical markets. In most cases, they have good tools and technical support and seek to address a wide range of embedded systems needs. All are commercial in nature meaning that they are for sale and not GPL or other "free" license..

Ardence, Inc.

Ardence develops software platforms for the on-demand world. Products include: RTX (RTX® - Real-time Extension for Control of Windows®), Phar Lap ETS (Real-time Operating System (RTOS) for x86 and Intel Architectures), ReadyON (Instant On for Windows® Devices), Ardence Select (Delivering multi-functional devices on a single operating system), and Ardence Global Services. [RTOS, Windows XP, XPe]
Web. http://www.ardence.com/
Score. 5

Avocet Systems, Inc.

From in-circuit emulators to Background Mode Debugger (BDM) solutions, from C compilers, assemblers, and simulators to an RTOS solution for any processor, we build the professional tools you need, offer the customer service you hope for, and provide the technical service you expect. [68K, 68HCxx, 8051, ZiLOG, Renesas, AvSys RTOS, BMD, HMI emulator]
Web. http://www.avocetsystems.com/
Score. 2

Blunk Microsystems

Hgh-performance royalty-free turnkey packages for embedded development under a simple licensing model. Products include RTOS, TCP/IP stack, file systems (FAT, NOR, NAND, RAM, ZLIB), and a LAPB stack. Systems are highly portable and are designed to run on any 32-bit CPU that has a C compiler and supports 8, 16, and 32 bit access. Systems are developed using TargetOS, but can easily ported to any commercial RTOS.
Web. http://www.blunkmicro.com/
Score. 4

CMX Systems, Inc.

CMX Systems' core business is to develop and support real-time, multi-tasking operating systems (RTOS), TCP/IP stacks, Flash File Systems, USB stacks, and CANopen protocols for a wide variety of 8-bit, 16-bit, and 32-bit microcomputers, microprocessors, and digital signal processors. The company's embedded software comes with full source code, is royalty-free, and supports more than 50 processor families and over 30 C-compiler vendors.
Web. http://www.cmx.com/
Score. 4

EBSnet, Inc.

EBSnet offers a high performance real time operating system (RTOS), a TCP-IP network stack, a full suite of Application Protocol, a powerful HTML-based embedded GUI SDK, a Web browser, an SDK for UPnP certified devices, CIFS/SMB protocol, 802.11 wireless support, a portable DOS/WIN9X compatible file system with real time extensions and FailSafe support, and a CD ROM file system. EBSnet products feature Extreme Portability to any kernel, any...
Web. http://www.ebsnetinc.com/
Score. 4

eSOL, Inc.

Founded in 1975 in Tokyo, Japan, eSOL is a leading embedded software developer with core technologies in real time operating systems. We develop, market and sell proven RTOS suites, along with a rich set of vertical oriented middleware libraries. Our rugged software development tools provide optimal reliability in backing up the highly complicated development process for RTOS-based applications. [TRON, iTRON, DSP, RTOS]
Web. http://www.esol.co.jp/english/index.html
Score. 5

Evidence

Erika Enterprise: the minimal RTOS for single and multicore embedded systems that solves the major problems related with the development of multicore applications. RT-Druid: the configuration and schedulability analysis tool for modelling, analyzing, and simulating the timing behaviour of embedded real-time systems. FLEX: embedded board based on Microchip dsPIC. Embedded Linux technical support and consulting. [Multicore, Linux Services]

Web. http://www.evidence.eu.com/

Score. 3

Express Logic, Inc.

Develops, markets and supports the ThreadX® real-time operating system (RTOS), NetX™ TCP/IP networking stack, USBX™ USB stack, and FileX® embedded file system for embedded applications. ThreadX is a royalty-free, full source code, small-footprint, low-overhead RTOS that is extremely easy to learn and use. ThreadX is one of the most widely deployed RTOS products in the world, with over 450 million products based on ThreadX. [ARM, ARC, PowerPC, XScale, ColdFire, 68K, 386, Hitachi H8, StarCore, SHARC, BlackFin, MicroBlaze, embedded, RTOS, USB stack, royalty-free, microcontroller]

Web. http://www.rtos.com/eg3_company.asp

Score. 5

Green Hills Software, Inc.

Founded in 1982, Green Hills Software, Inc. is the technology leader in real-time operating systems (RTOS) and device software optimization (DSO) for 32- and 64-bit embedded systems. Our royalty-free INTEGRITY(R) RTOS, velOSity(tm) kernel, µ-velOSity(tm) microkernel, compilers, MULTI(R) and AdaMULTI(tm) integrated development environments and TimeMachine(tm) debugger offer a complete development solution that addresses both...

Web. http://www.ghs.com/

Score. 5

HighTec EDV-Systeme GmbH

The Portable eXtendible real time operating system PXROS is a realtime operating system which was developed by HighTec. It offers hard real time with complete avoidance of disabling interrupts or interrupt latency and is extremely robust with the hightest event rates. Besides PXROS core other components can be added if this is required. HighTec GNU C/C++-Compiler. [x86, compiler, Linux support]

Web. http://www.hightec-rt.com/

Score. 3

KADAK Products Ltd.

KADAK provides integrated RTOS, TCP/IP and GUI solutions for embedded developers rushing to deliver their products to market. KADAK Products Ltd. provides the AMX RTOS, KwikNet. TCP/IP Stack and Web Server and KwikPeg GUI. KADAK's extensive documentation, Windows. Configuration Builder and KwikLook Fault Finder speed application development. Best of all, there are no royalties. [Real-Time Operating System]

Web. http://www.kadak.com/

Score. 2

LynuxWorks, Inc.

LynuxWorks is a world leader in the embedded software market, providing operating systems, software development products and consulting services for the world's most

successful communications, aerospace/defense, and consumer products companies. The company is a technology leader in the real-time operating systems (RTOS) industry, and a founding member of the Embedded Linux Consortium. [POSIX, LynxOS, BlueCat Linux]
Web. http://www.lynuxworks.com/
Score. 5

Mentor Graphics Corporation

Mentor Graphics® is a technology leader in electronic design automation (EDA), providing software and hardware design solutions that enable companies to develop better electronic products faster and more cost-effectively. The company offers innovative products and solutions that help engineers overcome the design challenges they face in the increasingly complex worlds of board and chip design. [FPGA, ASIC, Embedded, RTOS]
Web. http://www.mentor.com/
Score. 5

Micrium, Inc.

Micrium provides high quality, royalty-free software for the embedded market. Micrium is also renowned for the quality of its source code and documentation. In fact, the source code is probably the cleanest and most consistent code in the industry.

A validation suite exist for uC/OS-II and one is being developed for µC/TCP-IP and provides all the documentation necessary to prove that Micrium products are suitable for Safety Critical Systems common to Aviation and Medical products. Although this feature may not be applicable to your needs, it does prove that the Micrium products are VERY robust. [RTOS, GUI, USB, ARM, DRIVER]
Web. http://www.Micrium.com/
Score. 5

Micro Digital, Inc.

Micro Digital produces the modular SMX RTOS consisting of the high-performance smx Kernel, smxNS TCP/IP networking products, with add-on packages such as SNMP and WebServer, smxFS FAT file system, smxFFS flash file system, smxUSBH USB host stack, smxUSBD USB device stack, and smxUSBO USB OTG. SMX products are delivered ready to run on most ColdFires, ARM, PowerPC, and x86 processors. SMX software products are fully integrated with Code...
Web. http://www.smxinfo.com/
Score. 3

Microsoft Corp.

The Windows Embedded Family - a time-to-market advantage with the best set of tools and resources to accelerate development of next-generation intelligent devices. Windows CE - an advanced, real-time embedded operating system and powerful tools for creating smart, connected small-footprint devices. Windows XP Embedded - Windows XP in componentized form to optimize functionality when managing a customized device. .Net Microframework. [RTOS, WindowsCE, WindowsXP, Windows CE, Windows XP]
Web. http://clk.atdmt.com/MRT/go/g3xxxvse0130000032mrt/direct/01/
Score. 5

mSquared

mSquared offers a broad and complete line of RTOS that are priced aggressively to meet your technical and business needs. CPU's supported: 32-bit: MPC82xx (ARM 7/9), Freescale (ColdFire, MPC500, MPC555x MPC8xx, MPC82xx, MPC52xx), XScale, x86 16-bit: HCS12, M16C Others: Renesas SH2/M32C/H8S, NEC V850, Infineion, TriCore, MPX85xx, TI MSP430, TI OMAP (ARM + DSP) [VoIP, CIFS, cameras]
Web. http://www.m2resources.com/
Score. 2

OAR Corporation

OAR is the developer of RTEMS, the Real-Time Executive for Multiprocessor Systems. We proudly provide a unique set of services including support, training, and custom development. RTEMS is available free of charge with full source code provided. No royalties are required for applications that use RTEMS. OAR also provides GNAT/RTEMS, an Ada95 Compiler System suitable for cross development efforts. [RTOSemulation/simulation]

Web. http://www.oarcorp.com/

Score. 2

QNX Software Systems

The component-based architectures of the QNX® Neutrino® RTOS and QNX Momentics® development suite together provide the industry's most reliable and scalable framework for building innovative, high-performance embedded systems. [Keywords: RTOS, Real-time, x86, MIPS, ARM, PowerPC, Xscale, Eclipse, Posix, Hitachi, IDE, Multicore]

Web. http://www.qnx.com/

Score. 5

Quadros Systems, Inc.

Quadros Systems offers the RTXC Quadros RTOS family with four highly scalable and configurable kernels, offering distinct advantages for each of the major processing models--control processing, signal processing, convergent processing (unified RISC and DSP) and multi-processor/multi-core. Get all the advantages of this powerful operating system with the ability to easily configure it to fit the processing needs of your application. The RTOS is complemented by integrated middleware. [8051, Xscale, ST10, Renesas, C166, 68K, 68HCxx, ColdFire, TCP/IP, DSP, ARM, PowerPC, PPC, C166, Irda, CAN, USB, GUI, Real-time Operating Systems]

Web. http://www.quadros.com/

Score. 5

RadiSys Corp

Modular integrated platforms and solutions. Board-level embedded computers; blade servers; software-rich blades, network interfaces and packet processing engines; communications middleware and software; platforms and turnkey gateway systems. RadiSys is also an industry leader in technology development - VME, MicroTCA, AdvancedTCA (ATCA), COM Express, PMC, OS-9 RTOS from Microware, PCI, and more.

Web. http://www.radisys.com/

Score. 4

Sciopta Systems AG

Develops, sells, supports and maintains systems software for safety-critical embedded applications. This includes real-time operating systems, network software, file systems, software for interface bus systems, board support packages and other system software. SCIOPTA is a message based real-time operating system with many built-in safety functions and is therefore very well suited for software forming part of safety-related systems. [USB]

Web. http://www.sciopta.com/

Score. 4

SEGGER

SEGGER develops and distributes software development tools and ANSI "C" software components (middleware) for embedded systems. Main products are emWin a universal graphic software and GUI, embOS a small royalty-free realtime kernel, emFile a

FAT12/16/32 compatible Filesystem for embedded applications, emUSB designed to work on any embedded system with USB device controller and J-Link the JTAG emulator for ARM cores. [RTOS, MS430]
Web. http://www.segger.com/
Score. 4

SYSGO AG

SYSGO is specialized in development, implementation and customization of system software for embedded systems with strong focus on the OS environment. ELinOS is a distribution and toolchain for embedded Linux. PikeOS [RTOS] is a new multi-OS for secure combination of various OS in separate partitions on one hardware. SYSGO also offers consulting, support and trainings. The team has become synonymous to reliable high quality products and services.
Web. http://www.sysgo.com/
Score. 5

Wind River

Wind River is the global leader in Device Software Optimization (DSO). Wind River enables companies to develop, run, and manage device software faster, better, at lower cost, and more reliably. [RTOS, VxWorks, In-Circuit Emulator, IDE, Linux]
Web. http://www.windriver.com/
Score. 5

WITTENSTEIN high integrity systems

WITTENSTEIN high integrity systems are your partner for engineering systems including real time software.OpenRTOS (formally FreeRTOS-pro) provides commercial licensing and optional support packages for a popular and reliable real time kernel targeted at small embedded systems. WITTENSTEIN high integrity systems also offer SAFERTOS - a real time kernel with similar functionality that conforms to the IEC 61508 functional safety standard.
Web. http://www.highintegritysystems.com/
Score. 2

Appendix C: Commercial RTOSes, Focused

In this Appendix, we identify and summarize commercial real-time operating systems that are "focused" in nature – meaning that they generally support one or a very few chip architectures and/or target one or a very few vertical markets. All are commercial in nature meaning that they are for sale and not GPL or other "free" license.

Antcor

IkarusOS 'wireless (Wi-Fi) networking software', is a networking operating system which provides high-speed wireless connection. IkarusOS empowers modern OEMs with substantial benefits such as, compliance with the latest IEEE802.11 standards, innovative wireless features, support for WLAN chipsets from multiple vendors, accelerated time to market, lower development costs and risks, extensible for OEM differentiation and branding. Focusing on...
Web. http://www.antcor.com/
Score. 2

Arcticus Systems AB

Arcticus Systems AB is a Swedish corporation, formed in 1985, with recognized competence in the design of real-time applications and related program development tools. The goal of Arcticus Systems is to offer methods and tools to its customers, which provide for the cost-effective development and support of safety critical embedded systems. RTOS: Rubus OS, a real-time OS designed for dependable real-time systems [Automotive, Vehicle]
Web. http://www.arcticus-systems.com/
Score. 2

Ardence, Inc.

Ardence develops software platforms for the on-demand world. Products include: RTX (RTX® - Real-time Extension for Control of Windows®), Phar Lap ETS (Real-time Operating System (RTOS) for x86 and Intel Architectures), ReadyON (Instant On for Windows® Devices), Ardence Select (Delivering multi-functional devices on a single operating system), and Ardence Global Services. [RTOS, Windows XP, XPe]
Web. http://www.ardence.com/
Score. 5

AVIX-RT

Products and services for concurrent embedded systems. The main product of AVIX-RT is a real time preemptive scheduler called AVIX, targeting Microchip 16 bit microcontroller families. This scheduler exposes a number of unique features, the most prominent being full interrupt integration with no increased latency and an integrated real time thread activation trace mechanism. AVIX [RTOS, dsPIC, Microchip, zero latency]
Web. http://www.avix-rt.com/
Score. 2

Bae Systems: CsLEOS Real-Time Operating System

The CsLEOS RTOS is a commercial, off-the-shelf real-time operating system developed by BAE SYSTEMS to reduce lifecycle costs in safety-critical applications. The system incorporates the ARINC-653 open system applications programming interface to provide brick-wall time and space partitioning of separately loadable applications having different levels of criticality, resulting in greatly reduced test and certification costs. Developed and...
Web. http://www.baesystems-ps.com/CsLeos/
Score.

BTI Electronics

Engineering, design and manufacturing services with quality, value and on-time delivery. Main expertise is the development and manufacture of automotive, industrial and consumer products using embedded software controllers. Additionally, we develop custom equipment supporting our product development, manufacturing, and maintenance efforts. [NEMOS - Nexus Embedded Operating System, Freescale (68HCxx) and Renesas SH]

Web. http://www.btielectronics.com/
Score. 2

Datalight, Inc.

Datalight(r) is the market leader in Device Data Storage software technologies that manage data in embedded devices. Focus on portable, flexible solutions has enabled developers to save money, reduce development time and build customer confidence in their products. Datalight solutions result in unparalleled interoperability and increased customer satisfaction. [DOS, BIOS, File System, RTOS, Windows CE, VxWorks, Nucleus, intelligent flash driver]
Web. http://www.datalight.com/
Score. 3

DRDOS

As a desktop solution or an embedded application DR DOS is the ideal DOS system, designed for straight forward out-of-the-box implementations. Small, easy, well understood and priced right. Embedded Linux with the simplicity of DR DOS. DRLX allows the ability to load and run Linux applications from DR DOS.
Web. http://www.drdos.com/
Score. 2

EB

EB's cutting edge carrier-class Mobile WiMAX and 3G base station modules and R&D services ensure optimal cost-quality ratio and reliable global delivery to our customers. EB tresos® (RTOS) is a family of integrated products for developing embedded automotive software.
Web. http://www.automotive.elektrobit.com/
Score. 2

ECROS Technology

ECROS Operating System Priority-based Realtime Multi-Tasking in highly resource constrained microcontrollers - timers, events, stream I/O & more - easy to use - configurable. (RTOS). ZiLOG Z8 Encore! MCU Prototyping System [Automotive, Atmel AVR] The AVR 'ICE-Cube' is a compact and inexpensive replacement for the Atmel JTAG ICE. With it, you can access the OCD (On-Chip Debug) features of the AVR microcontrollers that have JTAG interfaces.
Web. http://www.ecrostech.com/
Score.

Eddy Solutions

Eddy Solutions is the source for Nimble -- the System on Chip RTOS. Eddy Solutions also provides professional services for developing deeply embedded systems. Nimble is a very fast, very compact, fully preemptive Real Time Operating System (RTOS) specifically designed for deeply embedded applications. Nimble comes complete with source code and a royalty-free licensing agreement that makes it easy to obtain. [SoC, ARM]
Web. http://www.eddysolutions.com/
Score. 3

Embedded Access Inc.

Embedded Access Inc. (EAI) has a suite of embedded software products that provide full support for Freescale's ColdFire, Power Quicc I, II, and IIpro CPUs. We offer fast and scalable platforms that are royalty free and come complete with source code and drivers. Our products include the MQX Real Time Operating System (RTOS), TCP/IP stack, USB Host, USB Device, USB OTG, Flash File System, Web Server, CANopen, and 1588. We also offer...

Web. http://www.embedded-access.com/
Score. 2

EmINENT Microsystems

ESF RTOS is a real-time executive that provides an unusually efficient and elegant suite of functions for creating and coordinating multiple, asynchronous threads of code execution. A unique object-oriented architecture contributes to code compactness, interface simplicity, ease of customization, and powerful functionality. It is a light weight real-time kernel, where threads share the same address space, and where operations [RTOS]
Web. http://www.EmINENTmicro.com/
Score.

Empower Technologies

Working alongside Texas Instruments (TI), Empower Technologies has developed LEOs® (Linux Embedded Operating System) configured for TI's OMAP5910 and OMAP5912 dual processor. The versatility of LEOs® and the power of OMAP initiate a new level of performance expectations for intelligent devices, with increased headroom for applications and expansion capabilities to best suit the needs of developers, manufacturers and consumers.
Web. http://www.empowertechnologies.com/
Score. 3

Enea Embedded Technology

Enea is the leading supplier of carrier grade platform software and professional engineering services to the global telecommunications industry, with approximately 50% market share of the world's new mobile phones and base stations. Embedded in more than 200 million devices worldwide, Enea's high-availability software technology also provides the foundation for a wide range of other complex, [RTOS] ,...
Web. http://www.enea.com/
Score. 3

Eointec Solutions

DSPEngine-- an operating system that runs on the DSP that simplifies the integration of ExpressDSP software components in order to create a complete DSP application. DSPIgnition -- a windows application and development interface that allows users to communicate with a DSP from their PC and allows engineers to design applications on their PC.
Web. http://www.eointecsolutions.com/
Score. 2

eSOL, Inc.

Founded in 1975 in Tokyo, Japan, eSOL is a leading embedded software developer with core technologies in real time operating systems. We develop, market and sell proven RTOS suites, along with a rich set of vertical oriented middleware libraries. Our rugged software development tools provide optimal reliability in backing up the highly complicated development process for RTOS-based applications. [TRON, iTRON, DSP, RTOS]
Web. http://www.esol.co.jp/english/index.html
Score. 5

Evidence

Erika Enterprise: the minimal RTOS for single and multicore embedded systems that solves the major problems related with the development of multicore applications. RT-Druid: the configuration and schedulability analysis tool for modelling, analyzing, and

simulating the timing behaviour of embedded real-time systems. FLEX: embedded board based on Microchip dsPIC. Embedded Linux technical support and consulting. [Multicore, Linux Services]
Web. http://www.evidence.eu.com/
Score. 3

General Software, Inc.

General Software provides highly configurable firmware solutions, software tools, engineering services, and technical support to designers of embedded x86 devices. General's products reduce risk, cost and time-to-market. Products: Embedded BIOSR 2000 offers maximum configurability at the source code level; Firmware Apps provide High Availability, Boot Security, & Platform updates; FirmbaseR SDK provides tools to build custom Apps that run in SMM.
Web. http://www.gensw.com/
Score. 4

HighTec EDV-Systeme GmbH

The Portable eXtendible real time operating system PXROS is a realtime operating system which was developed by HighTec. It offers hard real time with complete avoidance of disabling interrupts or interrupt latency and is extremely robust with the hightest event rates. Besides PXROS core other components can be added if this is required. HighTec GNU C/C++-Compiler. [x86, compiler, Linux support]
Web. http://www.hightec-rt.com/
Score. 3

Hi-Tech Software LLC

HI-TECH Software specializes in ANSI C cross compilers for over 12 different 8-bit, 16-bit, 32-bit, and DSP chip architectures. Our wide product range and comprehensive technical support is renowned for delivering cutting-edge technology and robust results for development teams worldwide. Salvo RTOS for 8051, Microchip PIC, and ARClite. [PIC, 8051, ARM, HOLTEK, 68HC11, RTOS, MSP430, HOLTEK, ARClite, XA, Z80].
Web. http://www.htsoft.com/
Score. 4

IAR Systems

IAR Systems provides a range of development tools for embedded systems: integrated development environments (IDE) with C/C++ compilers and debuggers, development kits, hardware debug probes and state machine design tools. The product line supports 8051, ARM, AVR, MSP430 and many other 8-, 16-, and 32-bit MCUs from different chip manufacturers. [PowerPAC RTOS for ARM]
Web. http://www.iar.com/
Score. 5

Insyde Software

Insyde Software is a leading global provider of system firmware and software engineering consulting services for companies in the notebook, desktop, server, and embedded systems. Founded in September 1998, the company is based in Taiwan. Insyde Software develops traditional and emerging firmware, supporting BIOS along with new standards Including UEFI/EFI Framework. [AMD Geode]
Web. http://www.insydesw.com/
Score. 3

Intrinsyc Software International, Inc.

Software and services that enable next-generation handheld and embedded products, including mobile handsets, smart phones and converged devices. Soleus is a complete

mobile handset software platform built on Windows® Embedded CE, with pre-certified telephony and a large application portfolio designed specifically to address the consumer mobile handset opportunity. [Windows Embedded CE, Windows, Java, JVM, Symbian, RTOS]
Web. http://www.intrinsyc.com/
Score. 3

JMI Software Systems, Inc.

JMI Software Systems, Inc. offers consulting services for custom drivers and product extensions for C EXECUTIVE and PSX, as well as complete applications systems. Additional major consulting services are available through associated companies . C EXECUTIVE is a real-time, multi-tasking, ROMable kernel for embedded systems, and is available for 8-, 16- and 32-bit CISC processors, a wide variety of RISC processors, and DSP. It provides fast...
Web. http://www.jmi.com/
Score.

KROS

KROS provides a royalty free, standards based, real time operating system for the Altera Nios embedded processor. KROS Real-Time Suite for the Altera Nios II processor, which includes a complete TCP/IP protocol suite, Filesystem, Graphical User Interface (GUI) and Real-Time kernel. NicheStack TCP/IP Network Stack for KROS provides TCP/IP support to your Nios powered device.
Web. http://kros.shugyodesign.com/
Score. 2

KUKA Roboter GmbH

KUKA's Real-Time Technology enables either Windows CE or VxWorks real-time operating systems to co-exist with Windows XP/2000 on the same machine. The hard real-time capabilities of Windows CE or VxWorks are not affected by the co-existent Windows XP. KUKA Controls' real-time extensions, CeWin® and VxWin®, offer a simple, reliable and cost effective solution for a range of applications. [robotics]
Web. http://www.kuka-controls.com/
Score. 3

Lassar Systems

Having trouble designing your next Microchip based Embedded System? Ava might be able to help. Exploiting many architecture specific features, only found in the PIC24 and dsPIC, Ava guarantees your system will always be in Real-Time. Ava is provided FREE for educational use. [RTOS, Microchip PIC]
Web. http://www.lassarsystems.com/
Score. 2

Microsoft Corp.

The Windows Embedded Family - a time-to-market advantage with the best set of tools and resources to accelerate development of next-generation intelligent devices. Windows CE - an advanced, real-time embedded operating system and powerful tools for creating smart, connected small-footprint devices. Windows XP Embedded - Windows XP in componentized form to optimize functionality when managing a customized device. .Net Microframework. [RTOS, WindowsCE, WindowsXP, Windows CE, Windows XP]
Web. http://clk.atdmt.com/MRT/go/g3xxxvse0130000032mrt/direct/01/
Score. 5

Miray Software

Sphere SP - the RT microkernel OS Miray has developed a realtime microkernel named Sphere. Sphere SP is a tiny RTOS that is based only on that microkernel. µnOS - the embeddable OS µnOS (speak: [mju:nos]) is based on Sphere SP. It's a highly developed 32-Bit-OS with GUI, Client/Server-Architecture and more features. [x86 RTOS, Symobi RTOS]
Web. http://www.miray.de/
Score. 3

NexGen Software

Designs and markets a complete TCP/IP suite and a portable graphics stacks for the Embedded Market. Products use a Low Layer software componant called NexGenOS which encapsulates all CPU, OS and tools dependencies giving the unique possibility to deliver the stacks either in Binary or Source code. The NexGenIP TCP/IP suite and NexGenGUI4 portable GUI-SDK are ported on the most common 16/32-Bit CPU(s) and RTOS(es). [XML]
Web. http://www.nexgen-software.com/
Score. 2

On Time Software

Produces On Time RTOS-32, a 32-bit royalty-free real-time operating system (RTOS) for Intel x86 compatible embedded systems. Its Windows NT subset kernel implements a Win32 subset API for source and binary compatibility with Windows NT. Available components include a real-time scheduler, file system, TCP/IP, USB host stack, and a Windows 95 look-and-feel GUI. Also available is RTKernel-C, a 16-bit RTOS for MS-DOS.
Web. http://www.on-time.com/
Score. 2

Pumpkin - Real-Time Software

Pumpkin's Salvo™ is the first Real-Time Operating System (RTOS) designed expressly for very-low-cost embedded systems with severely limited ROM and RAM. It even works without a software stack! By bringing real-time multitasking to the highest-volume end of the embedded processor market, Salvo gives you the power to quickly create low-cost, smart and sophisticated products for the Internet-enabled post-PC age.
Web. http://www.pumpkininc.com/
Score. 2

Real-Time Systems GmbH

Engineering services and products for embedded and real-time systems. This product enables modern multicore x86-processors to simultaneously run either multiple instances of a real-time operating system, such as Wind River VxWorks or a heterogeneous mixture of operating systems, including Linux and Microsoft Windows on a single execution platform. [RTOS]
Web. http://www.real-time-systems.com/
Score. 3

S&H Computer Systems

Learn more about the capabilites of the TSX-32 operating system. TSX-32 is a general purpose multi-user multi-tasking OS with real-time facilities for the X86 platform. Use it as an Internet server or as a development platform for your own custom applications.
Web. http://www.sandh.com/
Score. 2

Smart Network Devices

Small-memory footprint operating systems HyNetOS and 4NetOS (ARM & Hyperstone), geared to wire(less) networks and ideal for System-on-

chip. With the Micro WebTarget and StarTarget for system development (including tools, etc.), or as OEM Boards for customer solutions, Smart Network Devices has become a leading worldwide supplier. SND offers a complete and certified version 1.2 upper layer Bluetooth stack comprising L2CAP, RFCOMM and...
Web. http://www.smartnd.com/
Score. 3

Softools, Inc.

We specialize in tools for the Rabbit 2000/3000, Z80 & Z180, 8085 (and opcode compatible) processors. We are available for custom compiler design and have completed 3 in addition to our Z80/Z180 tools. We also have the most complete and flexible bank switching support available for large programs. TurboTask A tiny but full-featured and royalty-free real time operating system for the Rabbit and Z-80/Z180 microprocessors. [RTOS, ZiLOG]
Web. http://www.softools.com/
Score. 2

Symbian Ltd.

Symbian is a software licensing company that develops and supplies the advanced, open, standard operating system - Symbian OS - for data-enabled mobile phones. Symbian OS is the advanced, open operating system licensed by the world's leading mobile phone manufacturers. It is designed for the specific requirements of advanced 2.5G and 3G mobile phones. Symbian OS combines the power of an integrated applications environment... [Smart Phone, RTOS]
Web. http://www.symbian.com/
Score. 2

Tenasys

TenAsys provides the only real-time operating system (RTOS) designed and optimized specifically for the x86 architecture and Microsoft Windows software. Our products help you take advantage of the benefits of Windows - standard APIs, networking, GUI, and development environment - wile simultaneously providing you with the means to support your application's needs for a reliable real-time kernel to address your time-critical needs. [RMX]
Web. http://www.tenasys.com/
Score. 3

TTTech Computertechnik AG

Supplier of technology and software products in the field of time-triggered systems and TTP (Time-Triggered Protocol). TTTech products enable developers of aerospace, automotive, and industrial control equipment to deliver reliable embedded systems quickly and efficiently. TTTech's products comprise a complete software development environment for TTP-based systems, including hardware as well as TTP chip models. [RTOS, safety-critical]
Web. http://www.tttech.com/
Score. 2

Unicoi Systems, Inc.

Unicoi Systems is a leading provider of VoIP and multimedia software and reference designs to embedded device developers and OEMs worldwide. Unicoi' s Fusion line of software includes Fusion TCP/IP, the world's most widely deployed TCP/IP stack. Founded in 2002, over 200 leading companies rely on Unicoi to power their solutions, including Motorola, Raytheon, Scientific-Atlanta, Philips and Thomson. For more info, go to www.unicoi. com .
Web. http://www.unicoi.com/
Score. 4

VINJEY Software Systems (P) Ltd

VINJEY is here to enrich the embedded software markets with its unique featured products. Success through customer satisfaction, Solutions through innovations, Embark the best quality in all products are VINJEY's values. We grow by being at your reachable distance always and through your valuable feedbacks. VINJEY brings with it the expertise of Signal Processing, Audio Codecs, application-morphic RTOS, File systems, Drivers to provide the...

Web. http://www.vinjey.com/

Score. 2

Vita Nuova Inc.

Vita Nuova is an operating systems [RTOS] company specializing in technologies for distributed application development on network devices. Vita Nuova's major product, Inferno, is a small, low cost operating system for use in embedded systems. Inferno applications can be developed under many different operating systems, and run on x86, StrongARM, PowerPC and MIPS platforms. Full source code is available.

Web. http://www.vitanuova.com/

Score.

XPLab

XPLab current activities are: XPOLYPLUS software for Real Time applications Research laboratory for automation Development of vertical applications (diecasting) Hardware design and production Corporate Internet Solution Provider (CISP) [RTOS]

Web. http://www.xplab.net/

Score. 2

Appendix D: Linux / Embedded Linux

In this Appendix, we identify and summarize commercial real-time operating systems or products that related to embedded and real-time Linux. Even though they target the "free" Linux market, in all cases there is a strategy to make money by charging for services, additional software products, or other features. In most cases, they have good tools and technical support and seek to address a wide range of embedded systems needs. eg3.com's Linux coverage is at http://www.eg3.com/embedded-linux.htm.

Adescom, Inc.

Customized system solutions and intellectual property for real-time Internet applications. Our cross-functional teams provide protocol expertise and design experience for development, integration, and optimization of communication building blocks across hardware/software boundaries. [Embedded Linux for TriCore DSP, RTOS] Adescom offers products and design services for: VoIP telephony,IPTV & 3-play, Industrial and car...

Web. http://www.adescom.com/

Score. 2

Century Embedded Technologies

Century Software has developed core technologies for the new and fast paced embedded Linux industry. These technologies include: a graphical development environment; customized Internet browsers and HTML viewers; multimedia, including MP3 audio players and MPEG video viewers; and a PDA development suite. These core technologies were designed specifically to allow both hardware designers and their customers to use either a small footprint...

Web. http://embedded.censoft.com/

Score.

Concurrent Computer Corporation

Concurrent is a leading provider of high-performance, real-time Linux software and solutions for commercial and government markets. For 40 years Concurrent's best-of-breed products have enabled a range of time-critical solutions including: modeling and simulation, high speed data acquisition, visual imaging, low latency transaction processing and on-demand television. [COTS]

Web. http://www.ccur.com/

Score. 3

COSMO, Inc.

Professional system software company founded on the basis of 20 years experiences of developing large-scale digital switching systems in Korea. Software solutions for real-time embedded systems. Products include * Integration Development Environment for Embedded System * High Reliability Duplex System * High Reliability Realtime Operating System based on Linux * Ultrahigh speed, Small-size, Realtime DBMS [RTOS, Linux, StrongARM, PowerPC, Sparc]

Web. http://www.cosmo.re.kr/

Score. 3

DRDOS

As a desktop solution or an embedded application DR DOS is the ideal DOS system, designed for straight forward out-of-the-box implementations. Small, easy, well understood and priced right. Embedded Linux with the simplicity of DR DOS. DRLX allows the ability to load and run Linux applications from DR DOS.

Web. http://www.drdos.com/

Score. 2

Empower Technologies

Working alongside Texas Instruments (TI), Empower Technologies has developed LEOs® (Linux Embedded Operating System) configured for TI's OMAP5910 and OMAP5912 dual processor. The versatility of LEOs® and the power of OMAP initiate a new level of performance expectations for intelligent devices, with increased headroom for applications and expansion capabilities to best suit the needs of developers, manufacturers and consumers.

Web. http://www.empowertechnologies.com/

Score. 3

Evidence

Erika Enterprise: the minimal RTOS for single and multicore embedded systems that solves the major problems related with the development of multicore applications. RT-Druid: the configuration and schedulability analysis tool for modelling, analyzing, and simulating the timing behaviour of embedded real-time systems. FLEX: embedded board based on Microchip dsPIC. Embedded Linux technical support and consulting. [Multicore, Linux Services]

Web. http://www.evidence.eu.com/

Score. 3

HighTec EDV-Systeme GmbH

The Portable eXtendible real time operating system PXROS is a realtime operating system which was developed by HighTec. It offers hard real time with complete avoidance of disabling interrupts or interrupt latency and is extremely robust with the hightest event rates. Besides PXROS core other components can be added if this is required. HighTec GNU C/C++-Compiler. [x86, compiler, Linux support]

Web. http://www.hightec-rt.com/

Score. 3

Lineo Solutions, Inc.

Lineo Solutions, Inc. offers wide range of embedded solutions, including embedded Linux OS, development tools, sophisticated-and-robust software and reliable professional services. Our solutions make customers' product development process much easier, and bring faster time-to-market. Having 20-year experiences of embedded software development, Lineo has a high reputation for our technical capabilities, meeting customers' various...

Web. http://www.lineo.co.jp/eng/index.html

Score. 2

LynuxWorks, Inc.

LynuxWorks is a world leader in the embedded software market, providing operating systems, software development products and consulting services for the world's most successful communications, aerospace/defense, and consumer products companies. The company is a technology leader in the real-time operating systems (RTOS) industry, and a founding member of the Embedded Linux Consortium. [POSIX, LynxOS, BlueCat Linux]

Web. http://www.lynuxworks.com/

Score. 5

MIZI Research, Inc.

Numerous applications such as Hangul Word Processor for Linux and MIZI Prizm (a desktop Linux distro that has been localized and optimized for use in Korea). Recently, the focus of company's operations has shifted to developing embedded Linux solutions for various convergence devices like smartphones, PMPs, e-Book readers, vehicle telematics systems and videophones. Prizm, developed and distributed by MIZI Research, is a stable...

Web. http://www.mizi.com/

Score. 2

MontaVista Software

MontaVista is the leading provider of Linux for intelligent devices and telecommunications infrastructure. MontaVista delivers a commercial-quality Linux OS, plus time-saving development tools, expert support, and design & migration services. MontaVista embedded Linux runs on more processors and boards than any other. Over

2,000 companies develop with MontaVista to add functionality, increase reliability, reduce costs, and accelerate development.
Web. http://www.mvista.com/
Score. 5

RidgeRun (formerly Cadenux)

RidgeRun is your complete software integration products and services provider. We offer reference applications and rich software development kits (SDK) for different target platforms. "Free" SDK packages, so you can test out EVM boards. See our download center area for more information on RidgeRun Free BSP and open source efforts. [Linux]
Web. http://www.ridgerun.com/
Score. 2

SandorLABS-Visualize

SandorLABS is dedicated to create applications that are simple to use. Embedded Linux Solutions: * Want to switch from VxWorks, or another RTOS, to Linux. * Need Linux running on your embedded hardware. * Creating a System On Chip and need a Linux BSP. * Have existing hardware and don't know if Linux will run on it. SandorLABS has all the Linux solutions for you, your product and your company.
Web. http://www.sandorlabs.com/
Score. 2

SYSGO AG

SYSGO is specialized in development, implementation and customization of system software for embedded systems with strong focus on the OS environment. ELinOS is a distribution and toolchain for embedded Linux. PikeOS [RTOS] is a new multi-OS for secure combination of various OS in separate partitions on one hardware. SYSGO also offers consulting, support and trainings. The team has become synonymous to reliable high quality products and services.
Web. http://www.sysgo.com/
Score. 5

TimeSys Corporation

TimeSys provides hardware-optimized, ready-to-run Linux Development Kits (LDKs) for over 90 development boards, plus TimeStorm development and testing tools for all Linux distributions and TimeSys' hosted environment that provides everything that a developer or board vendor needs to get their products to market fast. Thousands of embedded Linux developers have chosen TimeSys solutions to streamline the development of Linux-based embedded systems.
Web. http://www.timesys.com/
Score. 4

Wind River

Wind River is the global leader in Device Software Optimization (DSO). Wind River enables companies to develop, run, and manage device software faster, better, at lower cost, and more reliably. [RTOS, VxWorks, In-Circuit Emulator, IDE, Linux]
Web. http://www.windriver.com/
Score. 5

Yellow Dog Linux for PowerPC

An open source, Linux operating system for home, office, server, and cluster users. Built upon the Fedora Core, Terra Soft has since the spring of 1999 developed and maintained YDL for the Power architecture family of CPUs. This focus and dedication has lead to the world's leading Linux for Power OS. (Desktop / server oriented, but some applicability to embedded systems).

Web. http://www.terrasoftsolutions.com/products/ydl/
Score. 3

Appendix E: Open Source RTOSes (Non-Linux)

In this Appendix, we identify and summarize commercial real-time operating systems or products that are "open source" but non-Linux. Even though they target the "free" or "open source" market, in all cases there is a strategy to make money by charging for services, additional software products, or other features. In most cases, they have good tools and technical support and seek to address a wide range of embedded systems needs.

eCosCentric

eCosCentric specializes in products and services for the eCos open source real-time operating system. Products include eCosPro Developer's Kits - industrial strength distributions of eCos, the award winning RedBoot bootloader, and a supporting range of middleware. Services offered consist of custom engineering, porting, support, training, and consultancy. eCosCentric's offerings provide a total eCos product development solution. [IDE, RTOS]
Web. http://www.ecoscentric.com/
Score. 4

OAR Corporation

OAR is the developer of RTEMS, the Real-Time Executive for Multiprocessor Systems. We proudly provide a unique set of services including support, training, and custom development. RTEMS is available free of charge with full source code provided. No royalties are required for applications that use RTEMS. OAR also provides GNAT/RTEMS, an Ada95 Compiler System suitable for cross development efforts. [RTOSemulation/simulation]
Web. http://www.oarcorp.com/
Score. 2

Open Kernel Labs

OK's open-source technology, OKL4, is the world's best-performing microkernel operating system. We are collaborating closely with NICTA, Australia's prestigious Center of Excellence in ICT research, on developing the first fully verified, proven bug-free operating systems kernel within two years. OKL4 users will have an upgrade path to a system of unprecedented security and trustworthiness. [RTOS, Virtualization]
Web. http://www.ok-labs.com/
Score. 3

Wasabi Systems

Wasabi Certified BSD is a tested, certified, and enhanced BSD-family operating system designed for advanced networked devices. WIth wide platform support and guaranteed performance, Wasabi Certified BSD (WCB) is the ideal choice for OEMs seeking to obtain the benefits of an open source operating system without the costs of the GPL license. [BSD RTOS]
Web. http://www.wasabisystems.com/
Score. 3

Appendix F: RTOS Misc & Tools

In this Appendix, we identify and summarize tools that work with another RTOS and/or allow you to generate your own "custom" RTOS as well as virtualization or hypervisor products. As such they do not fit into an easy category, but are nonetheless relevant in selecting an embedded RTOS. Most, if not all, are commercial in the sense that they are for sale but a few are non-commercial "free" tools available on the Internet.

ALT Software

Embedded Graphics Products Reliable graphic subsystems drivers and hardware targeting RTOS environments Expertise in avionics, automotive, medical imaging, handheld devices and industrial control systems Quality-tested OpenGL, OpenGL ES, and Mobile Direct3D device drivers, numerous RTOS, CPUs, safety standards, and ASICs
Web. http://www.altsoftware.com/
Score. 3

Applied Dynamics International

Provides state-of-the art software and hardware tools to the automotive, aerospace, and defense industries to design and test embedded control systems. Today, ADI's Advantage product line leverages all of our experience with commercial-off-the-shelf (COTS) computer processors, industry standard interface devices, and popular software applications. Our products are uniquely qualified to meet [RTOS]
Web. http://www.adi.com/
Score. 4

Concurrent Computer Corporation

Concurrent is a leading provider of high-performance, real-time Linux software and solutions for commercial and government markets. For 40 years Concurrent's best-of-breed products have enabled a range of time-critical solutions including: modeling and simulation, high speed data acquisition, visual imaging, low latency transaction processing and on-demand television. [COTS]
Web. http://www.ccur.com/
Score. 3

DDC-I, Inc.

DDC-I is a global supplier of software development tools, custom software development services, and legacy software system modernization. DDC-I's customer base is an impressive 'who's who' in the commercial, military, aerospace, and safety-critical industries. Tools include compiler systems and run-time systems for C, Embedded C++, Ada, Fortran and JOVIAL application development. [Java, JVM, powerpc, rtos, tools for embedded application]
Web. http://www.ddci.com/
Score. 3

Eridon Corp.

Eridon is defining the emerging field of Embedded Computing Objects where self-integrating subsystems consisting of circuit designs, supporting electronic logic, and software drivers become simple units that are easily and automatically assembled into production-ready designs. Eridon's UnifiedLogic framework dramatically reduces time-to-market and development risk through its ... [FPGA, PCB, Board Design, RTOS]
Web. http://www.eridon.com/
Score. 5

MapuSoft Technologies

MapuSoft's OS Changer solutions are available for porting code from pSOS, VxWorks, and Nucleus to another OS. Our OS Abstractor solution allows you to develop code independent of the underlying OS and reuse POSIX code on a different OS. Our OS PAL (OS Porting and Abstraction Lab) solution integrates OS Changer and OS Abstractor in an Eclipse based IDE for host development and includes a code generator to produce optimized code for a target OS.
Web. http://www.mapusoft.com/
Score. 3

Open Kernel Labs

OK's open-source technology, OKL4, is the world's best-performing microkernel operating system. We are collaborating closely with NICTA, Australia's prestigious Center of Excellence in ICT research, on developing the first fully verified, proven bug-free operating systems kernel within two years. OKL4 users will have an upgrade path to a system of unprecedented security and trustworthiness. [RTOS, Virtualization]
Web. http://www.ok-labs.com/
Score. 3

Professional Software Associates, Inc. (PSA)

Software engineering company which develops software products and provides comprehensive development services.PSA development services include specialized embedded graphics (GUI) development, display drivers, custom fonts, and custom printer drivers. To handle printing for embedded appliances, PSA's VxPDDK product is a royalty free printer driver development kit for VxWorks (contact PSA for other RTOS'es supported). VxPDDK takes advantage of...
Web. http://www.psa-software.com/
Score.

RoweBots Research Inc.

Embedded DSP development products for OEMs: host and algorithm development or deeply embedded development products which extend to distributed, multicore and multiprocessor platforms. iZoom! eliminates C/C++ bottlenecks. DSPnano RTOS provides the ultimate environment for developing embedded DSP systems including: algorithm development, Eclipse IDE for C/C++, DSP RTOS, DSP libraries. Hardware independent - many DSP processors supported.
Web. http://www.rowebots.com/
Score. 3

Trango Virtual Processors

TRANGO technology enables processor secured virtualization for major 32/64 bit RISC architectures : ARM, MIPS, PowerPC, SH, configurable cores and FPGA soft-cores. It is based on a para-virtualization approach (no instruction emulation): it ensures full predictability and close-to-native performance. TRANGO hypervisors are based on TRANGO technology and specialized for each CPU architecture.
Web. http://www.trango-vp.com/
Score. 2

Trolltech

Trolltech is a software company with two flagship products: Qt and Qtopia. Qt is a multi-platform C++ application framework that lets developers write single-source applications that run natively on Windows, Linux, Unix, Mac OS X and embedded Linux. Qt has been used to build thousands of successful commercial applications worldwide. Qtopia is the first comprehensive application environment built for embedded Linux.
Web. http://www.trolltech.com/
Score. 3

VirtualLogix

VirtualLogix™ is the global leader in Real-Time Virtualization™ technology for connected devices. VirtualLogix VLX enables multiple operating system environments to run concurrently on shared hardware and provides a range of performance, fault tolerance and security options to address specific market requirements.
Web. http://www.virtuallogix.com/
Score. 5

Visuality Systems

Visuality Systems provides VisualityNQ (network quick), a CIFS file sharing middleware protocol for embedded real time operating systems. Using VisualityNQ, embedded devices based on VxWorks, Windows CE, Linux and other RTOS can be quickly connected to Windows centric networks while providing full file sharing capabilities.
Web. http://www.visualitynq.com/
Score. 2

Vmware, Inc.

Virtualization solutions and services. VMware solutions separate the operating system and application software from the underlying hardware, delivering significant improvements in efficiency, availability, flexibility and manageability. With a customer base of 20,000+ organizations of all sizes, VMware delivers technology designed to substantially lower IT costs, provide more flexibility in choosing operating systems... [ESX Server, GSX Server]
Web. http://www.vmware.com/
Score. 2

Zeidman Technologies

Develops of software tools for the design of embedded systems. Successful designs include a flat panel display, an ATM router, a multifunction printer/copier/fax, an Ethernet analysis system, a medical imaging system, and a parallel processor for simulation acceleration. The company has created a new tool called SynthOS that uses patented software synthesis technology to automatically generate a reliable, optimized, real-time operating system.
Web. http://www.zeidman.biz/
Score. 2

Appendix G: Non-Commercial Linux

In this Appendix, we identify "non-commercial" Linux distributions. Many of these are university or non-maintained, but they are an alternative to http://www.kernel.org/ *as well as to the "commercial" Linux distributions of companies like MontaVista or Wind River.*

linux

2-Disk X window embedded Linux
2-Disk X window embedded Linux is a tiny net-centric Linux that aims at portable secure remote system usage. It contains many utilities including: X Windows, vncviewer, rdesktop, a Web browser, a file manager, a text editor, a terminal, a window manager, a menu system, a dialog system, X scripting facilities, and many others. It aims to work from 1 or 2 floppy disks in any remote location.

Web. http://freshmeat.net/projects/natld/
Score. 3

linux

BlueCat Linux Evaluation Download
LynuxWorks offers an unsupported evaluation version of BlueCat Linux 4.0 for non-commercial use on x86 targets. To purchase a supported, complete version of BlueCat Linux, visit here.

Web. http://www.lynuxworks.com/products/bluecat/download.php3
Score. 4

linux

CentOS
CentOS is an Enterprise-class Linux Distribution derived from sources freely provided to the public by a prominent North American Enterprise Linux vendor. CentOS conforms fully with the upstream vendors redistribution policy and aims to be 100% binary compatible. (CentOS mainly changes packages to remove upstream vendor branding and artwork.) CentOS is free.

Web. http://www.centos.org/
Score. 3

linux

CRUX Linux for PowerPC
CRUX PPC is a port of CRUX for the Power Architecture platform. It's a GNU system with a Linux kernel and runs on Apple NewWorld (both 32 and 64bit) PowerPC, Genesi PegasosII workstations and EFIKA embedded boards, IBM RS/6000 CHRP and pSeries servers, and is almost completely ported to ACube Sam440ep modular board. CRUX PPC includes support for laptop-specific features, extended hardware support and server tools.

Web. http://cruxppc.sunsite.dk/wp/index.php
Score. 3

linux

CRUX Linux
CRUX is a lightweight, i686-optimized Linux distribution targeted at experienced Linux users. The primary focus of this distribution is keep it simple, which is reflected in a straightforward tar.gz-based package system, BSD-style initscripts, and a relatively small collection of trimmed packages. The secondary focus is utilization of new Linux features and recent tools and libraries. CRUX also has a ports system which makes it easy to...

Web. http://www.crux.nu/
Score. 3

linux

ELKS - Embedded Linux Kernel Subset

ELKS is the Embeddable Linux Kernel Subset, a project to build a small kernel subset of Linux (which will provide more or less UNIX V7 functionality within the kernel) that can run on machines with limited processor and memory resources. More information on the background, goals and current status of the project can be found at the ELKS home page. The initial proposed targets are the Intel 8086 and eventually the 286's 16-bit protected mode.

Web. http://elks.sourceforge.net/
Score. 3

linux

Embedded Linux/Microcontroller Project
Port of Linux to systems without a Memory Management Unit (MMU). Pronounced 'you-see-linux', currently supports Motorola 68K, ColdFire, ADI BlackFin, ARM, i960, Xilinx Microblaze, Renesas H8,

Web. http://www.uclinux.org/
Score. 4

linux

emKnoppix
emKnoppix is a distribution of Knoppix tailored for use in embedded systems. Idea for such a distribution arose when I was playing with Knoppix for remastering and building an embedded Linux platform at the same time Features As of this release, emKnoppix has just ssh server. But you can still use it as a firewall, router or for NAT. This is the first release and is considered as a technology demonstrator. Future releases will contain...

Web. http://emknoppix.sarovar.org/
Score. 3

linux

Fedora Core
Fedora Core is a free operating system that offers the best combination of stable and cutting-edge software that exists in the free software world. Modestly put, that is the opening sentence of the Fedora web page. Sponsored by Red Hat.

Web. http://fedora.redhat.com/
Score. 3

linux

Gentoo Linux / BSD
Gentoo is a free operating system based on either Linux or FreeBSD that can be automatically optimized and customized for just about any application or need. Extreme configurability, performance and a top-notch user and developer community are all hallmarks of the Gentoo experience. Can be used in embedded systems.

Web. http://www.gentoo.org/
Score. 4

linux

IMediaLinux
Small Embedded Linux distribution for mini-ITX / x86 systems. iMedia Linux is a hybrid between a small embedded linux distribution and full featured linux distributions. Linux embedded systems are used in ATMs, Kiosks, vending machines, industrial computers, Point of Sale, mutimedia system, Internet appliances and dozens of other applications.

VIA C3, VIA C7, Geode, i586, i686 CPU targets Eyecandy desktop with full XOrg X Server Complete...

Web. http://www.imedialinux.com/
Score. 4

linux

KURT-Linux: Kansas University Real-Time Linux
Outdated and unmaintained 'real-time' Linux from the University of Kansas.

Web. http://www.ittc.ku.edu/kurt/
Score. 1

linux

L4Linux
L4Linux is a port of Linux to the L4 µ-kernel. L4Linux runs as an L4 server in user-mode, side-by-side with other L4 applications (e.g. real-time components). It is currently running on x86 and ARM and it is binary compatible with the native Linux kernels.

Web. http://www.l4hq.org/
Score. 4

linux

Linux MIPS port
Linux/MIPS is a port of the widespread UNIX clone Linux to the MIPS architecture. Linux/MIPS runs on a large number of technically very different systems ranging from small embedded systems to large desktop machines and servers from SGI and DEC.

Web. http://www.linux-mips.org/
Score. 4

linux

Linux/M32R Linux
Linux/M32R is Linux for M32R, which is a 32-bit RISC processor of Renesas Technology. This project was funded and supported by NEDO (New Energy and Industrial Technology Development Organization) of Japan.

Web. http://www.linux-m32r.org/
Score. 3

linux

Linux/RK
Linux /RK stands for Linux/Resource Kernel, which incorporates real-time extensions to the Linux kernel to support the abstractions of a resource kernel. A resource kernel is a real-time kernel (operating system) that provides timely, guaranteed and enforced access to system resources for applications.

Web. http://www.cs.cmu.edu/%7Erajkumar/linux-rk.html
Score. 1

linux

LinuxSH Wiki
This is the LinuxSH wiki. LinuxSH is a port of the Linux kernel to the SuperH and SH-Mobile family of processors from Renesas Technology, STMicroelectronics, the former SuperH, Inc., and legacy parts from Hitachi.

Web. http://www.linux-sh.org/
Score. 3

linux

PeeWee Linux
PeeWeeLinux is an ongoing development effort to provide an environment that makes the configuration and installation of a Linux operating system on an embedded platform as easy and painless as possible. Some of the key features of PeeWeeLinux are:

Web. http://embedded.sourceforge.net/
Score. 1

linux

PeeWeeLinux
PeeWeeLinux is an ongoing development effort to provide an environment that makes the configuration and installation of a Linux operating system on an embedded platform as easy and painless as possible. Some of the key features of PeeWeeLinux are: Developed on a RedHat 6.2 platform Packages build and maintained using rpm Packages are customized to minimize memory footprint Ncurses driven graphical configuration and installation...

Web. http://www.peeweelinux.com/
Score. 1

linux

Red-Linux Real-Time Kernel
The search for a powerful, flexible and open real-time operating systems continues! The real-time research group at the University of California, Irvine has built a real-time kernel based on the popular Linux kernel. One of the most important reasons for us to choose Linux as the foundation of our real-time kernel project is the large user population.

Web. http://linux.ece.uci.edu/RED-Linux/
Score. 3

linux

RTnet - Hard Real-Time Networking for Real-Time Linux
RTnet is an Open Soure hard real-time network protocol stack for Xenomai and RTAI (real-time Linux extensions). It makes use of standard Ethernet hardware and supports several popular NIC chip sets, including Gigabit Ethernet. Moreover, Ethernet-over-1394 support is available based on the RT-FireWire protocol stack. RTnet implements UDP/IP, ICMP and ARP in a deterministic way. It provides a POSIX socket API to real-time user space...

Web. http://www.rtnet.org/
Score. 3

linux

STLinux distribution
This is a Linux distribution targetted at STMicroelectronics Consumer electronics parts which are based around the ST40 or ST200 CPUs.

Web. http://www.stlinux.com/
Score. 3

linux

T2 System Development Environment

T2 SDE is not just a common Linux distribution - it is a flexible open source System Development Environment or Distribution Build Kit (some might even name it Meta Distribution ...). T2 allows the creation of custom distributions with bleeding edge technology, up-to-date packages and integrated support for cross compilation. Currently the Linux kernel is normally used - but we are expanding to Minix, Hurd, OpenDarwin and OpenBSD - more to come

Web. http://www.t2-project.org/
Score. 3

linux

TA-Linux

Free Linux distribution, that targets Linux power users. It's main goal is to have a small base installation that the end-user can expand to include the software he needs. The secondary goal is to support as many different architectures as possible.
At this time i386, Alpha, PPC, Sparc and MIPS are fully supported with PA-RISC around the corner. Work is also underway to support some more exotic and old platforms like the m68k Macintosh and Sun3.

Web. http://talinux.tal.org/
Score. 3

linux

The Linux Kernel Archives

This is the primary site for the Linux kernel source, but it has much more than just Linux kernels. It is not, of course, a site for real-time or embedded Linux but it is useful if you are intent on taking Linux from the source and reformulating it to meet the needs of your embedded and/or realtime project.

Web. http://www.kernel.org/
Score. 4

linux

uClinux Blackfin processor Project

This web site is designed to be the central repository and open source workspace for non-commercial software and hardware projects targetted for use with Analog Devices' family of Blackfin processors. In addition to a wide range of applications, this workspace also focuses on supporting open source hardware and software tools for the Blackfin processor

Web. http://blackfin.uclinux.org/
Score. 4

linux

UltraLinux

UltraLinux is the name given to the port of Linux the SPARC family of processors most commonly found in Sun workstations and clones. The port has been developed over the past few years and is currently very stable. It supports most workstations including the older 32bit SPARC processors and the newer 64bit UltraSPARC based workstations.

Web. http://www.ultralinux.org/
Score. 3

linux

Xenomai: Real-Time framework for Linux

Xenomai is a real-time development framework cooperating with the Linux kernel, in order to provide a pervasive, interface-agnostic, hard real-time support to user-space applications, seamlessly integrated into the GNU/Linux environment.

Web. http://www.xenomai.org/
Score. 3

linux

Yellow Dog Linux for PowerPC

An open source, Linux operating system for home, office, server, and cluster users. Built upon the Fedora Core, Terra Soft has since the spring of 1999 developed and maintained YDL for the Power architecture family of CPUs. This focus and dedication has lead to the world's leading Linux for Power OS. (Desktop / server oriented, but some applicability to embedded systems).

Web. http://www.terrasoftsolutions.com/products/ydl/
Score. 3

Appendix H: "Free" RTOSes

In this Appendix, we identify "free" or "open source" non-Linux RTOSes. Many of these are university or non-maintained, but a few are extremely popular as nothing sells like free. But beware of licensing restrictions that curtail commercial use. *eg3.com's index to this category is at http://www.eg3.com/rtos-free.htm.*

rtos

avr-rtos
A Real-Time Operating System kernel for the Atmel AVR microcontroller, incorporating tasks, memory management, semaphores, queues, and other features

Web. http://sourceforge.net/projects/avr-rtos/
Score. 3

rtos

AvrX Real Time Kernel
AvrX is a Real Time Multitasking Kernel written for the Atmel AVR series of micro controllers. AvrX contains approximately 40 API in the following Six categories: Tasking * Semaphores * Timer Management * Message Queues * Single Step Debugging support * Byte FIFO support with synchronization. The Kernel is written in assembly. Total kernel size varies from ~500 to 700 words depending upon which version is being used.

Web. http://www.barello.net/
Score. 5

rtos

C Kernel
The C Kernel is a real-time preemptive kernel specially written for embedded applications running on various micro controller environments. Supports x86, 8051, C166, ZiLOG, Fujitsu F2MC-16 Since The C Kernel is written in C it is easy to make new bindings (ports) to other compilers and micro controllers or processors. GNU Lesser GPL License.

Web. http://home.hetnet.nl/~p_vd_vlugt/
Score. 4

rtos

CapROS: The Capability-based Reliable Operating System
CapROS is a new operating system for the x86 architecture that merges some very old ideas about capabilities with some newer ideas about performance and resource management. The result is a small, secure, real-time operating system that provides orthogonal persistence. CapROS is an experimental capability-based operating system, for the Intel architecture, based on EROS, KeyKOS, and Gnosis.

Web. http://www.capros.org/
Score. 4

rtos

Chimera Real-Time Operating System
A next generation multiprocessor real-time operating system (RTOS) designed especially to support the development of dynamically reconfigurable software for robotic and automation systems. Chimera is a VMEbus-based operating system which supports multiple general and special purpose processors. General purpose processors come in the form of single-board-computers (currently MC680x0 family of processors supported) which we call Real-Time...

Web. http://www-2.cs.cmu.edu/~aml/chimera/chimera.html
Score. 1

rtos

Coyotos Secure Operating System
Coyotos is a secure, microkernel-based operating system that builds on the ideas and experiences of the EROS project. Much of the code developed for EROS will migrate

directly to Coyotos. Coyotos is being developed on AMD-64 and Pentium platforms. Once we have a baseline kernel working, we would welcome help getting it running on PowerPC as well.

Web. http://www.coyotos.org/
Score. 4

rtos

CubTrix - Real-time Kernel
CubTrix is a graphical user interface with support for real-time applications. Built on top of a small, fast and efficient real-time microkernel. Download CubTrix a Cubical Real-Time Kernel Project Admins: pettersson Operating System: (None Listed) License: BSD License Category: Operating System Kernels

Web. https://sourceforge.net/projects/cubtrix
Score. 3

rtos

E.R.I.K.A. (ERIKA) - Open Source RTOS for Automotive
ERIKA Educational is a a full-fledged and fully functional RTOS distributed under the GNU GPL license. It has been designed to be an effective and attractive Educational platform for real-time programming or embedded systems courses. A first possibility for students and instructors is to use ERIKA educational as a programming environment to develop small control applications. Supports Hitachi H8/300.

Web. http://erika.sssup.it/
Score. 4

rtos

eCos home page
Open source, royalty-free, real-time operating system intended for embedded applications. The highly configurable nature of eCos allows the operating system to be customised to precise application requirements, delivering the best possible run-time performance and an optimised hardware resource footprint. Supports ARM, Xscale, Renesas H8, x86, 68K, MIPS, PowerPC, Sparc, SuperH, and many others.

Web. http://ecos.sourceware.org/
Score. 4

rtos

Fiasco
Fiasco is a preemptible real-time kernel supporting hard priorities. It uses non-blocking synchronization for its kernel objects. This guarantees priority inheritance and makes sure that runnable high-priority processes never block waiting for lower-priority processes. Supports x86 architecture.

Web. http://os.inf.tu-dresden.de/fiasco/
Score. 4

rtos

Free Dos
The goal of the FreeDOS Project is to create a completely <B>FREE</B> MS-DOS(tm) compatible operating system. FreeDOS will run on all DOS capable platforms, from XT's to Pentium Pro's.

Web. http://www.freedos.org/

Score. 5

rtos

FreeRTOS - Realtime Scheduler
Portable, open source, mini Real Time Kernel - a free to download and royalty free RTOS that can be used in commercial applications. FreeRTOS is licensed under a modified GPL and can be used in commercial applications under this license. (Supports ARM Cortex-M3, ARM7, ARM9, HCS12, H8S, MSP430, Microblaze, Coldfire, AVR, x86, 8051, PIC24 & dsPIC.)

Web. http://www.freertos.org/
Score. 5

rtos

Hartik - The HArd Real TIme Kernel
HARTIK is a hard real-time kernel designed to help the development of real-time applications, ranging from critical control systems to soft multimedia, distributed systems. It was developed at the RETIS Lab as a research project to explore the applicability of real-time theory to real world applications. HARTIK Applications: Monitoring and control systems, Advanced robotics, Multimedia applications, Scheduling Simulator.

Web. http://hartik.sssup.it/
Score. 4

rtos

IVMUK RTOS
Basic OS for small MCU's. Supports Microchip's PIC18F252 chip.

Web. http://ivmuk-os.sourceforge.net/
Score.

rtos

Katix project
Katix real time operating system (Katix RTOS) and Katix Embedded Linux (Black Cat Linux). Katix Embedded Linux distribution is available for Motorola MPC5200 microprocessor family.

Web. http://www.katix.org/
Score. 2

rtos

L4 µ-Kernel Family
Originally, L4 is the name of a second-generation µ-kernel (microkernel) designed and implemented by Jochen Liedtke, running on i486 and Pentium CPUs. However, there are now numerous implementations of the L4 API (application programming interface) on several hardware architectures.

Web. http://os.inf.tu-dresden.de/L4/
Score. 2

rtos

L4Ka::Pistachio RTOS
Vision is a microkernel technology that can be and is used advantageously for constructing any general or customized operating system including pervasive systems,

deep-computing systems, and huge servers. Supports Alpha (21164, 21264) AMD64 (Opteron 242, Simics) ARM (SA1100, XScale, ARM925T) IA32 (Pentium and higher) IA64 (Itanium1, Itanium2, Ski) MIPS 64bit (R4000, R5000) PowerPC 32bit (IBM 750) PowerPC 64bit (Power3, Power4).

Web. http://www.l4ka.org/
Score. 4

rtos

MaRTE OS - Minimal Real-Time Operating System for Embedded Applications
MaRTE OS is a real-time kernel for embedded applications that follows the Minimal Real-Time POSIX.13 subset. Most of its code is written in Ada with some C and assembler parts. It allows software cross-development of Ada and C applications using the GNU compilers Gnat and Gcc. Supports x86.

Web. http://marte.unican.es/
Score. 5

rtos

Maruti Project, University of Maryland
Maruti 3.0 is an embeddable hard real-time runtime system for distributed and single-node systems. Also includes a development environment. Not updated since 1996.

Web. http://www.cs.umd.edu/projects/maruti/
Score. 1

rtos

MenuetOS
Hobby Operating System for the PC written entirely in 64bit assembly language, and released under License. It supports 64 and 32 bit x86 assembly programming for smaller, faster and less resource hungry applications. Menuet has no roots within unix or the posix standards, nor is it based on any particular operating system. The design goal has been to remove the extra layers between different parts of an OS, which normally complicates...

Web. http://www.menuetos.net/
Score. 3

rtos

NicheTask 'Free' RTOS
The royalty-free 'C' source code is being made freely available to all device developers. Key Features: ANSI 'C' Source Code. Low memory footprint Portable to any CPU Network protocol stacks available from InterNiche Easy to upgrade to a preemptive RTOS without adding overhead Supports simple migration from one RTOS to another Multiple tasks, limited only by memory availability. Sponsored by InterNiche.

Web. http://www.freertos.com/
Score. 5

rtos

Open Ravenscar Real-Time Kernel
Open-source real-time kernel of reduced size and complexity, wich can be used to develop high-integrity real-time applications in a subset of the Ada 95 language compatible with the Ravenscar Profile. The Ravenscar Profile defines a subset of the Ada tasking features which can be implemented using a small reliable kernel. Supports x86 and SPARC.

Web. http://polaris.dit.upm.es/~ork/
Score. 1

rtos

Openmoko - Integrated Open Source Mobile Communications Platform
OpenMoko is an attempt to create the world's first completely open mobile phone software stack. The OpenMoko Wiki is our centralized, collaborative effort to collect and maintain all kinds of information and documentation on the OpenMoko software platform as well as the supported devices (Neo1973).

Web. http://www.openmoko.org/
Score. 3

rtos

openOSEK
Q: What is openOSEK? A: openOSEK is an open source and cross-platform operating system framework, which aims for full conformance with the OSEK/VDX(TM) derived specification (ISO 17356). Q: Why OSEK? A: Clearly there are some excellent open source RTOS alternatives (FreeRTOS, eCos, Jaluna, etc). The problem with these operating systems, from an automotive perspective, is: Most of these alternatives are quite bulky and their real-time...

Web. http://www.openosek.org/
Score. 4

rtos

OS Kit
The OSKit is a framework and a set of 31 component libraries oriented to operating systems, together with extensive documentation. The OSKit's goal is to lower the barrier to entry to OS R&D and to lower its costs. The OSKit makes it vastly easier to create a new OS, port an existing OS to the x86.

Web. http://www.cs.utah.edu/projects/flux/oskit/
Score. 1

rtos

Phoenix-RTOS
Phoenix-RTOS is the realtime operating system intended for embedded systems. The main goal of the Phoenix-RTOS project is to develop free, portable, small and well-designed realtime operating system for embedded hardware platforms like SBC (Single Board Computers), SOM (System on Module) and SoC (System On Chip). Support for · IA32 · PowerPC · ARM

Web. http://www.phoenix-rtos.org/
Score. 4

rtos

pico]OS
pico]OS is a highly configurable and very fast real time operating system (RTOS). It targets a wide range of architectures, from very small 8 bit processors and microcontrollers up to very huge platforms. Ports are available for 6502, 80x86, PPC and AVR.

Web. http://picoos.sourceforge.net/
Score. 4

rtos

PICOS18 - Operating Systems for PIC Microcontrollers
This web site presents PICOS18, an operating system based on OSEK/VDX, an open industry standard. The OSEK standard is well suited for the low computing, power, small memory embedded controllers used in automotive or robotique applications. PICOS18 designed by Pragmatec Inc. is an operating system for the PICmicro microcontrollers from the Microchip PIC18xxx family. It is totaly free and is distributed under the GPL licence.

Web. http://www.picos18.com/index_us.htm
Score. 3

rtos

PORTOS - Real Time Operating System
PORTOS is a new kind of RTOS based on the concepts of priority functions and priority objects. The main goal behind PORTOS is to simplify the programming of embedded systems, to enable cleaner designs, and to improve performance. PORTOS is essentially a library of routines that can be either used stand-alone or integrated in any existing Real Time Operating System. A license is required for any commercial use.

Web. http://www.portos.org/
Score. 4

rtos

Prex project
The Prex project is an open source initiative to provide a portable real-time operating system for embedded systems. This project originally started to make a royalty-free OS for mobile phones and PDA in order to spread open source movement to such proprietary OS fields. And now, the project target has been shifted to the wider area of real-time and embedded systems. Prex is designed specifically for small memory footprint platforms. It...

Web. http://prex.sourceforge.net/
Score. 3

rtos

proc Real-Time Kernel
Produces FREE proc Real-Time Kernel: Compact and RTOS for embedded systems. Also for DOS. proc runs on several different architectures. Source included. Other products include The Message Router, NMX Message Exchange and the nemon Boot Monitor. Supported architectures: x86 / PC 68HC11, (also banked) H8, (8 and 16 bits) 68k / 6833x ColdFire 80C196 C167 AVR i960Jx MSP430 Z180

Web. http://www.nilsenelektronikk.no/neprod.html
Score. 4

rtos

RDOS Operating System
RDOS Operating System License RDOS source code is distributed as GPL, except for commercial usage in embedded systems. * Serial Network Protocol for embedded systems. FREE only for non-commercial use. Commercial use requires a LICENSE. Supports x86.

Web. http://www.rdos.net/rdos/index.htm
Score. 2

rtos

RTEMS - The Real-Time Executive for Multiprocessor Systems
Commercial grade RTOS designed for deeply embedded systems. It is a free open source solution (GPL) that supports multi-processor systems. RTEMS is designed to support applications with the most stringent real-time requirements while being compatible with open standards. Development hosts include both MS-Windows and Unix (GNU/Linux, FreeBSD, Solaris, MacOS X, etc.) platforms. Supports ARM, TI DSPs, Renesas, x86, i960, 68K, MIPS, PowerPC, SPARC.

Web. http://www.rtems.com/
Score. 5

rtos

rtmk - real-time microkernel
rtmk is free real-time microkernel developed by Johan Rydberg. It shared alot of ideas with the Mach microkernel developed by CMU in the late 80's and early 90's (project stopped in -94).

Web. http://rtmk.sourceforge.net/
Score. 3

rtos

RTOS UH: Real-Time Operating System for Process Automation
Programming system PEARL RTOS-UH was designed for the special requirements found in process automation and automatic control and is based on the processor families MC 68xxx, MC 683xx as well as on the PowerPC. RTOS-UH offers the same programming and real-time environment regardless which hardware is used - from the smallest single board MC 68008 up to the high-end PowerPC 604 VME-bus system.

Web. http://www.irt.uni-hannover.de/rtos/rtos-gb.html
Score. 4

rtos

S.Ha.R.K. (Soft Hard Real-Time Kernel)
S.Ha.R.K. is a dynamic configurable kernel architecture designed for supporting hard, soft, and non real-time applications with interchangeable scheduling algorithms. The kernel is fully modular in terms of scheduling policies, aperiodic servers, and concurrency control protocols, which typically are not modular in traditional operating systems.

Web. http://shark.sssup.it/
Score. 4

rtos

Scout Operating System
Scout is a communication-oriented operating system targeted at network appliances (e.g., network-attached devices, set-top boxes, hand-held devices, and so on). Scout currently runs stand-alone on Digital Alpha and Intel Pentium processors. It is hosted from Linux using GNU tools. Scout includes source code derived from GNU libraries, Linux device drivers, Bellcore's MGR window manager, and the x-kernel.

Web. http://www.cs.arizona.edu/scout/index.html
Score. 1

rtos

T-Engine Project
T-Engine is a new standardized development platform into which the eTRON architecture has been installed. The eTRON architecture enables the secure transfer of information and rights through insecure network channels such as the Internet. By using the T-Engine platform, network devices such as mobile phones, portable devices, and information devices can be efficiently developed. Source code available for download.

Web. http://www.t-engine.org/
Score. 3

rtos

The Contiki Operating System
Contiki is an open source, highly portable, networked, multi-tasking operating system for memory-constrained systems. Contiki runs on a variety of tiny systems. Supports MSP430, x86, AVR, and various old computer formats. Code footprint is on the order of kilobytes and memory usage can be configured to be as low as tens of bytes.

Web. http://www.sics.se/~adam/contiki/
Score. 3

rtos

The ROME Operating System
ROME is a lightweight, modular, multitasking, embedded operating system which has been developed and used for multiple research projects within the Computer & Communications Research Laboratory (CCRL) of NEC USA, Inc. in Princeton, NJ. ROME OS has been released under the GNU General Public License (GPL). Supports Intel x86, i960, and MIPS r4000.

Web. http://rome.sourceforge.net/
Score. 1

rtos

TinyOS
TinyOS is an open-source operating system designed for wireless embedded sensor networks. It features a component-based architecture which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks.

Web. http://www.tinyos.net/
Score. 5

rtos

TNKernel RTOS
Compact and very fast real-time kernel for embedded 32/16 bits microprocessors. TNKernel performs preemptive priority-based scheduling with round-robin scheduling ability for tasks with identical priority. Current version of TNKernel includes semaphores, mutexes, data queues, event flags and fixed-sized memory pool operation. System's functions calls in interrupts are supported. Supports ARM architecture. FreeBSD-like license.

Web. http://www.tnkernel.com/
Score. 5

rtos

USIX Operating System

The USIX Operating System is designed to provide source and binary compatibility with SVR4/386 while offering guaranteed response time to interrupts in real-time applications, multiprocessor kernel architecture, better resource utilization, a more robust file system and the ability to dynamically extend the kernel without re-linking.

Web. http://usix.rosweb.ru/
Score. 1

rtos

uSmartX RTOS

Non-preemptive, multitasking, priority based RTOS. It features necessary mechanisms for inter-task communication and basic task and time control functions. uSmartX is targeted for small foot-print embedded designs. Being a non-preemptive kernel it consumes little memory and it is very simple to use. Supports ARM, AVR, Renesas H8-300H.

Web. http://usmartx.sourceforge.net/
Score. 3

rtos

WhatOS (RTOS Creation Tool)

WhatOS is a free open source embedded system development solution. It provides a complete set of tools for creating high-quality, reliable embedded systems. These include: a real-time operating system (RTOS) generator, a simulator for testing and debugging generated systems, and tools for interacting with systems remotely after they have been embedded.

Web. http://www.sticlete.com/whatos/
Score. 5

rtos

XMK - eXtreme Minimal Kernel

XMK is a preemptive multi-tasking kernel for 8bit microcontrollers. Its goal is to provide a bare bones RTOS with a small enough footprint (RAM+ROM) to run on 8bit microcontrollers. Open Source RTOS for 8Bit microcontrollers - Including Hitachi H8 and Atmel's AVR chips.

Web. http://www.shift-right.com/xmk/
Score. 3

rtos

Xoberon: Hard Real-Time Operating System for Mechatronics

XOberon/PowerPC is a hard real-time operating system (HRTOS), deployed for high-end mechatronic products control. It has been developed at the Institute of Robotics, Swiss Federal Institute of Technology, Zurich. XOberon is loosely based on the Oberon System, and it is written in the Oberon-2 programming language.

Web. http://www.ifr.mavt.ethz.ch/research/xoberon/index.html
Score. 3

Appendix I: Key RTOS Resources

In this Appendix, we identify the "best" or "most useful" resources for learning about RTOSes. It's a long list, and be sure to use the online full version at http://www.eg3.com/rtos. When you are learning about RTOSes, there are many good websites, conferences, publications, seminars, and other venues via which to begin your own "learning" process.

portal

All Linux Devices

All Linux Devices is a resource for professionals interested in maintaining the highest level of awareness pertaining to Linux and the Open Source (TM) community news. This Zine covers Linux and Embedded or Realtime Linux.

Web. http://alllinuxdevices.com/
Score. 5

overview

An Introduction to Real-time Operating Systems

A real-time operating system or RTOS (sometimes known as a real-time executive or kernel) is a library of functions that implements rules and policies concerning time-critical allocation of a computer system's resources. The intent of this document is to provide a top-level overview of real-time operating systems.

Web. http://www.quadros.com/pdf/article_Quadros-RTOS_2006-09.pdf
Score. 5

project

Carrier Grade Linux

Carrier Grade Linux provides the availability, scalability, and service response characteristics required by carrier-grade applications. Carrier Grade Linux complies to recognized standards such as the Linux Standard Base. The effort is aligned with existing forums and communities such as the Service Availability Forum.

Web. http://www.linux-foundation.org/en/Carrier_Grade_Linux
Score. 5

newsgroup

comp.os.linux.embedded

This is a relatively new newsgroup (discussion board) devoted to the bad boy of all real-time operating systems, the spoiler of commercial royalties, and the love child of nerd everywhere: embedded Linux.

Web. news:comp.os.linux.embedded
Score. 5

newsgroup

comp.realtime

Devoted to realtime computing, both RTOS and realtime issues.

Web. news:comp.realtime
Score. 5

newsgroup

comp.realtime

Devoted to realtime computing, both RTOS and realtime issues.

Web. news:comp.realtime
Score. 5

portal

Dedicated Systems - Realtime Encyclopedia

Meta-resource on the Web, located in Europe, and focused on identifying and indexing the best resource for realtime systems -- both RTOS and board-level sorts of issues. Has some great 'buyers guide' of available RTOS's and tools, as well as reviews of Windows CE, QNX, etc.

Web. http://www.dedicated-systems.com/encyc/
Score. 5

portal

Dedicated Systems - Realtime Encyclopedia

Meta-resource on the Web, located in Europe, and focused on identifying and indexing the best resource for realtime systems -- both RTOS and board-level sorts of issues. Has some great 'buyers guide' of available RTOS's and tools, as well as reviews of Windows CE, QNX, etc.

Web. http://www.dedicated-systems.com/encyc/
Score. 5

overview

Embedded Linux @ Wikipedia

Embedded Linux is the designation for Linux-based operating systems that are used as embedded operating systems in cell phones, personal digital assistants, media player handsets and other consumer electronics devices. Linux is also suitable for other embedded applications such as networking equipment, machine control, industrial automation, navigation equipment, and medical instruments. Embedded Linux can be characterized as different...

Web. http://en.wikipedia.org/wiki/Embedded_Linux
Score. 5

resource

Foundry27 (for QNX)

Foundry27 provides the infrastructure for members of the QNX community to share source code and binaries, to support each other through their development experience, to suggest improvements, to develop software in a transparent manner in projects, and to promote technologies that support or leverage the QNX software platform.

Web. http://www.foundry27.com/
Score. 5

buyers guide

Free RTOS Buyers Guides and Reports

The RTOS Buyer's Guide A comprehensive list of commercial RTOS. The information is provided and maintained by the RTOS vendors themselves. Vendors who's RTOS is not yet listed, can add it at their convenience. Free download RTOS evaluation reports In this section you can download several RTOS evaluation reports, as well as some accompanying documents and white papers.

Web. http://www.dedicated-systems.com/encyc/buyersguide/rtos/rtosmenu.htm
Score. 5

buyers guide

Free RTOS Buyers Guides and Reports

The RTOS Buyer's Guide A comprehensive list of commercial RTOS. The information is provided and maintained by the RTOS vendors themselves. Vendors who's RTOS is not

yet listed, can add it at their convenience. Free download RTOS evaluation reports In this section you can download several RTOS evaluation reports, as well as some accompanying documents and white papers.

Web. http://www.dedicated-systems.com/encyc/buyersguide/rtos/rtosmenu.htm
Score. 5

buyers guide

Free RTOS Buyers Guides and Reports
The RTOS Buyer's Guide A comprehensive list of commercial RTOS. The information is provided and maintained by the RTOS vendors themselves. Vendors who's RTOS is not yet listed, can add it at their convenience. Free download RTOS evaluation reports In this section you can download several RTOS evaluation reports, as well as some accompanying documents and white papers.

Web. http://www.dedicated-systems.com/encyc/buyersguide/rtos/rtosmenu.htm
Score. 5

paper

How to Use Real-Time Multitasking Kernels In Embedded System
The purpose of this booklet is to familiarize you with the functions provided by a typical commercial kernel and how to utilize them. Once you have a better understanding of these, the benefits of using a kernel will become apparent.

Web. http://www.smxinfo.com/howto/howto.htm
Score. 5

hot list

IEEE-CS Technical Committee on Real-Time Systems
TC-RTS Archives, Real-Time Research Groups at Universities, Archives of Real-Time Conferences and Workshops, List of Commercially Available Real-Time Products. A very good page if you are ACADEMICALLY inclined but the page does not list to commercial resources, or anything that is remotely practical per the philosophy of its moderator.

Web. http://cs-www.bu.edu/pub/ieee-rts/Home.html
Score. 5

project

ITRON Project Archive
The ITRON Project creates standards for real-time operating system specifications for embedded systems and related specifications. Many products have been developed based on the ITRON specifications, and now ITRON is a de-fact standard operating system specification for small-scale embedded systems in Japan. The ITRON Project is promoted by the ITRON Specification Group in the TRON Association, as one of the subproject of the TRON Project.

Web. http://www.assoc.tron.org/
Score. 5

project

ITRON Project Home Page
We have published 'Protection Extension of µITRON4.0 Specification (µITRON4.0/PX Specification; Japanese version)', which is to add access protection functions of kernel objects including memory area. The detailed information on the specification is here. We

have published the English version of the µITRON4.0 Specification (Ver. 4.00.00). The detailed information on the specification is here.

Web. http://www.ertl.jp/ITRON/home-e.html
Score. 5

project

Linux High Availability Project
The basic goal of the High Availability Linux project is to: Provide a high-availability (clustering) solution for Linux which promotes reliability, availability, and serviceability (RAS) through a community development effort.

Web. http://www.linux-ha.org/
Score. 5

portal

LinuxDevices The Embedded Linux Portal
Our goal is to provide the highest possible concentration of quality information regarding the use of Linux in embedded applications. We intend to accomplish this by enabling YOU, the real Linux experts, to share information with others in a free and open manner. We will depend on the motivation and integrity of the embedded Linux developer community to make this vision a reality.

Web. http://www.linuxdevices.com/
Score. 5

personal page

Mikehall's Embedded WebLog
A look at Windows CE, Windows XP Embedded (and anything else that's cool or interesting) by one of the 'insiders' at Microsoft on the Windows XP Embedded team.

Web. http://blogs.msdn.com/mikehall/default.aspx
Score. 5

organization

Official TRON Project Home
This page for TRON contains news, information on events, and general information related to TRON.

Web. http://www.tron.org/
Score. 5

portal

OpenQNX.com
Major portal for QNX users and community - not clear who is the sponsor. QNX?

Web. http://www.openqnx.com/
Score. 5

resource

OSDev.org - Operating System Development
Hello, my name is Chase and I'd like to welcome you to osdev. This site has become one of the major starting places for those brave few that want to venture into the realm of Operating System programming. Check out the links and OS development projects. Add

new sites that aren't listed and rate the current sites. Try to add the OS projects to the OSDev Ring and the useful resources/tutorials to the links page.

Web. http://www.osdev.org/
Score. 5

project

OSEK
RTOS for automobile industry. OSEK has been founded as a joint project in the German automotive industry aiming at an industry standard for an open-ended architecture for distributed control units in vehicles. OSEK is an abbreviation for the German term "Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug" (English "Open Systems and the Corresponding Interfaces for Automotive Electronics").

Web. http://www.osek-vdx.org/
Score. 5

organization

Portable Application Standards Committee
Official web site of the IEEE's Portable Application Standards Committee (PASC). PASC is the group that has and continues to develop the POSIX family of standards.

Web. http://www.pasc.org/
Score. 5

standards

POSIX Certified by the IEEE and The Open Group
Voluntary program jointly administered by the IEEE and The Open Group, open to any product meeting the conformance requirements. The program is now open for certification of products meeting the 2003 Edition of IEEE 1003.1-2001, 'Standard for Information Technology--Portable Operating System Interface (POSIX)' which incorporates the IEEE 1003.1 Corrigendum.

Web. http://posixcertified.ieee.org/
Score. 5

hot list

Programming POSIX threads
Provide a forum to discuss and collect information about programming using POSIX Threads, especially using free implementations like Patched MIT Pthreads and LinuxThreads.

Web. http://www.humanfactor.com/pthreads/
Score. 5

overview

QNX @ Wikipedia
QNX (pronounced either Q-N-X or Q-nix) is a commercial POSIX-compliant Unix-like real-time operating system, aimed primarily at the embedded systems market. As of the 12th of September 2007, the source of the QNX kernel has been released for non-commercial use. Contents [hide] 1 Description 2 History 3 Technology 4 Foundry27 5 Competitors 6 References 7 Bibliography 8 External links [edit] Description As a...

Web. http://en.wikipedia.org/wiki/QNX
Score. 5

association

QNX Developers Netrwork

If you're tired of white papers and would really prefer sample source or want to dive straight to our manuals, you've come to the right place. We recommend that you have a look at our articles for a wide range of topics from graphics to IPC. New articles appear weekly and yes, source code and tutorials are on their way.

Web. http://www.qnx.com/developers/
Score. 5

overview

Real Time and Embedded HOWTO

This HOWTO is about the fundamentals of (i) real-time and embedded operating systems (focusing mostly on their differences with general purpose operating systems such as Linux), and (ii) real-time programming. Everything is illustrated with Open Source examples: RTLinux, RTAI, eCos, RT-EMS, uCLinux, ...

Web. http://www.mech.kuleuven.ac.be/~bruyninc/rthowto/
Score. 5

conference

Real-time and Embedded Computing Conference

Focused on embedded applications with moderate to high performance computing needs in OEM markets. These include medical, defense/aerospace, gaming, industrial control/automation, vehicular control, vending machine, instrumentation/data acquisition and many more.

Web. http://www.rtecc.com/
Score. 5

project

RTAI - the RealTime Application Interface for Linux from DIAPM

Then RTAI might be the realtime extension of your choice! The Realtime Application Interface consists mainly of two parts: A patch to the Linux kernel which introduces a hardware abstraction layer A broad variety of services which make realtime programmers' lifes easier RTAI is a true community project.RTAI supports several architectures: x86 (with and without FPU and TSC) x86_64 (beta) PowerPC (recovering) ARM...

Web. https://www.rtai.org/
Score. 5

overview

RTOS @ Wikipedia

A real-time operating system (RTOS) is a multitasking operating system intended for real-time applications. Such applications include embedded systems (programmable thermostats, household appliance controllers, mobile telephones), industrial robots, spacecraft, industrial control (see SCADA), and scientific research equipment. An RTOS facilitates the creation of a real-time system, but does not guarantee the final result will be...

Web. http://en.wikipedia.org/wiki/RTOS
Score. 5

overview

RTOS @ Wikipedia

A real-time operating system (RTOS) is a multitasking operating system intended for real-time applications. Such applications include embedded systems (programmable thermostats, household appliance controllers, mobile telephones), industrial robots, spacecraft, industrial control (see SCADA), and scientific research equipment. An RTOS facilitates the creation of a real-time system, but does not guarantee the final result will be...

Web. http://en.wikipedia.org/wiki/RTOS
Score. 5

overview

RTOS @ Wikipedia

A real-time operating system (RTOS) is a multitasking operating system intended for real-time applications. Such applications include embedded systems (programmable thermostats, household appliance controllers, mobile telephones), industrial robots, spacecraft, industrial control (see SCADA), and scientific research equipment. An RTOS facilitates the creation of a real-time system, but does not guarantee the final result will be...

Web. http://en.wikipedia.org/wiki/RTOS
Score. 5

tutorial

RTOS Report: Selecting an Embedded RTOS

Many, if not all, embedded projects require a real-time operating system. As part of eg3.com's research program, this report details the various options: free RTOSes, embedded Linux, paid RTOSes (commercial) as well as in-depth survey results, overviews, and datasheets. Due to generous vendor sponsorship, this report is availble for FREE for a limited time.

Web. http://www.rtos-report.org/
Score. 5

personal page

Windows XP Embedded Team Blog

This posting is provided 'AS IS' with no warranties, and confers no rights. Any downloads provided by the Embedded team are at your own risk and should be considered in BETA state until they're available from the Microsoft.com downloads site for XPe.

Web. http://blogs.msdn.com/embedded/
Score. 5

overview

A Selection Methodology for the RTOS Market

This paper compares commercially available RTOSs that are suitable for space missions requiring hard realtime capabilities.

Web. http://www.qnx.com/download/feature.html?programid=8089
Score. 4

overview

A Survey of Contemporary Real-time Operating Systems

A real-time operating system (RTOS) supports applications that must meet deadlines in addition to providing logically correct results. This paper reviews pre-requisites for an

RTOS to be POSIX 1003.1b compliant and discusses memory management and scheduling in RTOS. We survey the prominent commercial and research RTOSs and outline steps in system implementation with an RTOS. We select a popular commercial RTOS within each category of...

Web. http://ai.ijs.si/informatica/PDF/29-2/12_Baskiyar-A%20Survey%20of%20Contemporary...pdf
Score. 4

project

Adeos Project
The purpose of Adeos is to provide a flexible environment for sharing hardware resources among multiple operating systems, or among multiple instances of a single OS. To this end, Adeos enables multiple prioritized domains to exist simultaneously on the same hardware. For instance, we have successfully inserted the Adeos nanokernel beneath the Linux kernel, opening a full range of new possibilities, notably in the fields of SMP...

Web. http://home.gna.org/adeos/
Score. 4

organization

American Society of Safety Engineers
The American Society of Safety Engineers (ASSE) is the world's oldest and largest professional safety organization. ASSE's mission is to foster the technical, scientific, managerial and ethical knowledge, skills and competency of safety, health and environmental professionals for the protection of people, property and the environment, and to advance the status and promote the advancement of the safety profession.

Web. http://www.ASSE.org/
Score. 4

paper

Basic Concepts of Real-Time Operating Systems
To most people, embedded systems are not recognizable as computers. Instead, they are hidden inside everyday objects that surround us and help us in our lives. Embedded systems typically do not interface with the outside world through familiar personal computer interface devices such as a mouse, keyboard and graphic user interface.

Web. http://www.kalinskyassociates.com/Wpaper1.html
Score. 4

resource

Beyond Logic
Resource for topics such as USB, TCP/IP, and Embedded Linux. Very detailed and useful.

Web. http://www.beyondlogic.org/
Score. 4

seminar

Bootcamp for Beginners - MontaVista Linux
We are pleased to offer Bootcamp for Beginners to anyone interested in Embedded Linux Developing. The five course tutorial sessions are in order and being offered on Monday, October 8th only. Tutorial Sessions Getting Started on Your First Embedded Linux Project Speaker: Mike Anderson This presentation will present the steps required to configure a standard Linux desktop system for cross development using an example embedded Linux...

Web. http://www.mvista.com/vision/boot.html
Score. 4

seminar

Bootcamp for Beginners - MontaVista Linux
We are pleased to offer Bootcamp for Beginners to anyone interested in Embedded Linux Developing. The five course tutorial sessions are in order and being offered on Monday, October 8th only. Tutorial Sessions Getting Started on Your First Embedded Linux Project Speaker: Mike Anderson This presentation will present the steps required to configure a standard Linux desktop system for cross development using an example embedded Linux...

Web. http://www.mvista.com/vision/boot.html
Score. 4

tool

BusyBox: The Swiss Army Knife of Embedded Linux
BusyBox combines tiny versions of many common UNIX utilities into a single small executable. It provides replacements for most of the utilities you usually find in GNU fileutils, shellutils, etc. The utilities in BusyBox generally have fewer options than their full-featured GNU cousins; however, the options that are included provide the expected functionality and behave very much like their GNU counterparts. BusyBox provides a fairly...

Web. http://www.busybox.net/
Score. 4

freeware

Cheddar: a free real time scheduling analyzer
Cheddar is a free real time scheduling tool. Cheddar is designed for checking task temporal constraints of a real time application/system. It can help you for quick prototyping of real time schedulers. It can also be used for educational purpose. Cheddar is developed and maintened by the LISYC Team, University of Brest.

Web. http://beru.univ-brest.fr/~singhoff/cheddar/
Score. 4

newsgroup

comp.os.lynx
Specific to the LYNX OS.

Web. news:comp.os.lynx
Score. 4

newsgroup

comp.os.qnx
Specific to QNX.

Web. news:comp.os.qnx
Score. 4

newsgroup

comp.os.qnx
Specific to QNX.

Web. news:comp.os.qnx
Score. 4

newsgroup

comp.os.vxworks
Specific to VxWorks from Wind River Systems.

Web. news:comp.os.vxworks
Score. 4

newsgroup

comp.os.vxworks
Specific to VxWorks from Wind River Systems.

Web. news:comp.os.vxworks
Score. 4

publication

Crosstalk - The Journal of Defense Software Engineering
CrossTalk, The Journal of Defense Software Engineering is an approved Department of Defense journal. CrossTalk's mission is to encourage the engineering development of software in order to improve the reliability, sustainability, and responsiveness of our warfighting capability and to inform and educate readers on up-to-date policy decisions and new software engineering technologies.

Web. http://www.stsc.hill.af.mil/crosstalk/about.html
Score. 4

archive

D. Kalinsky Associates White Papers
A Survey of Task Schedulers * Basic Concepts of Real-Time Operating Systems * * Mutexes Battle Priority Inversions * Architecture of Device I/O Drivers * Architecture of Safety Critical Systems'

Web. http://www.kalinskyassociates.com/EmbeddedResources.html
Score. 4

archive

D. Kalinsky Associates White Papers
A Survey of Task Schedulers * Basic Concepts of Real-Time Operating Systems * * Mutexes Battle Priority Inversions * Architecture of Device I/O Drivers * Architecture of Safety Critical Systems'

Web. http://www.kalinskyassociates.com/EmbeddedResources.html
Score. 4

publication

Dedicated Systems Magazine
European magazine on realtime issues - Bus (VME and non-VME), Real-Time Operating Systems, Tools (CASE, Bus Analysers, Debuggers, Monitors, Simulators, Compilers, Emulators,...), Systems (I/O), etc.

Web. http://www.dedicated-systems.com/magazine/magazine.htm
Score. 4

project

DIAPM RTAI - Realtime Application Interface

This is the homepage of RTAI - the Realtime Linux Application Interface for Linux - which lets you write applications with strict timing constraints for your favourite operating system. Like Linux itself this software is a community effort. If you are interested in what it does just join our mailing list and help our team!

Web. http://www.aero.polimi.it/~rtai/
Score. 4

cd-rom

eCos CD-ROM

As a service to the open source community eCosCentric provides a CD-ROM containing the official eCos 2.0 Beta distribution. These are available at a cost to cover just the postage and packing. The release was produced by the eCos maintainers and eCosCentric with contributions from the eCos community. The CD-ROM contains the full set of eCos and RedBoot sources.

Web. http://www.ecoscentric.com/ecos/order.shtml
Score. 4

paper

Embedded Linux Survey

This survey presents the state of embedded Linux as applied to consumer electronics devices, from wristwatches to PDAs to cellular handsets. We have attempted to address important decision areas for embedded Linux developers, namely sorting through the various kernel variants, distributions, C libraries, and GUI options in creating a coherent software architecture.

Web. http://www.bluemug.com/research/els/els.pdf
Score. 4

seminar

Embedded Linux World Tour

The Embedded Linux World Tour 2007 seminars will provide free presentations and demos in 10 cities in the U.S. and Canada and 13 cities in Europe. Admission to the seminars is free, courtesy of MontaVista Software, Freescale, Avnet, and EBV Elektronik. To register now, click on your choice of city in the left column. The seminars will include live demonstrations of Eclipse-based development tools for the Freescale PowerQUICC® II Pro...

Web. http://www.mvista.com/tour/
Score. 4

seminar

Embedded Linux World Tour

The Embedded Linux World Tour 2007 seminars will provide free presentations and demos in 10 cities in the U.S. and Canada and 13 cities in Europe. Admission to the seminars is free, courtesy of MontaVista Software, Freescale, Avnet, and EBV Elektronik. To register now, click on your choice of city in the left column. The seminars will include live demonstrations of Eclipse-based development tools for the Freescale PowerQUICC® II Pro...

Web. http://www.mvista.com/tour/
Score. 4

resource

Embedded Operating System Development

This section of the MSDN Library is devoted to Microsoft Windows Embedded technical content, such as technical articles and product documentation. Windows Embedded consists of a set of operating systems and tools that allow you to build your embedded device with a rich set of componentized technologies. Windows Embedded consists of three products: * Windows CE * Windows XP Embedded * Windows 2000 with Server Appliance Kit

Web. http://msdn2.microsoft.com/en-us/library/aa286493.aspx
Score. 4

resource

Embedded Operating System Development

This section of the MSDN Library is devoted to Microsoft Windows Embedded technical content, such as technical articles and product documentation. Windows Embedded consists of a set of operating systems and tools that allow you to build your embedded device with a rich set of componentized technologies. Windows Embedded consists of three products: * Windows CE * Windows XP Embedded * Windows 2000 with Server Appliance Kit

Web. http://msdn2.microsoft.com/en-us/library/aa286493.aspx
Score. 4

personal page

embeddedTUX.org

embeddedTUX.org is the companion site to Karim Yaghmour's Building Embedded Linux Systems. Along with the book, this site is meant to provide embedded systems developers with all the information they need to build embedded systems based on the Linux kernel using only freely available open source and free software packages.

Web. http://www.embeddedtux.org/
Score. 4

resource

Free OS - the site for Free Operating Systems

FreeOS.com is a resource center for Free Operating Systems. It provides Articles, Documentation, Download Sites, News, Links and anything related to all Free Operating Sys

Web. http://www.freeos.com/
Score. 4

hot list

Freebyte's Guide to free and other Operating Systems

Introduction Free Linux Operating Systems Free BSD Operating Systems Other free Unix Operating Systems Free BeOS Windows Free DOS clones Emulators Commercial Operating Systems

Web. http://www.freebyte.com/operatingsystems/
Score. 4

project

Gnome Mobile

GNOME Mobile will advance the use, development and commercialization of GNOME components as a mobile and embedded user experience platform. It brings together industry leaders, expert consultants, key developers and the community and industry organizations they represent. [Linux]

Web. http://www.gnome.org/mobile/
Score. 4

archive

Green Hills Critique of Linux: The Archive
Green Hills has consolidated their white papers criticizing Linux into a nice easy-to-find single site. White Papers Part I- FAA Safety-critical Certified Operating Systems Deliver The Reliability and Security Required by Defense Systems; Linux Does Not Part II - 'Many Eyes' - No Assurance Against Many Spies Part III - Linux Security: Unfit for Retrofit Part IV - Linux in Defense: Free Software is Just Too...

Web. http://www.ghs.com/linux.html
Score. 4

organization

IEEE POSIX® Certification Authority
Offers validation service for the National Institute of Standards and Technology (NIST) POSIX® (Portable Operating System Interface), Federal Information Processing Standards (FIPS) 151-2. FIPS 151-2 adopts ISO/IEC 9945-1: 1990 (IEEE Std. 1003.1: 1990)

Web. http://standards.ieee.org/regauth/posix/index.html
Score. 4

contest

Imagine Cup - Windows Embedded Design Contest
Devices are becoming smaller, more portable and are having a greater impact on our everyday lives. Here is your opportunity to unleash your creativity and to change the world by developing your own embedded device. Formerly the Windows Embedded Student Challenge, this competition challenges you to go beyond the desktop, challenge your creativity, and to build a complete hardware and software solution using Windows CE and the hardware...

Web. http://www.imaginecup.com/
Score. 4

paper

Implementing Device Drivers - Migrating from Linux to a Microkernel OS
Recent market developments around IP ownership have prompted many companies to investigate the possibility of migrating their Linux code to a commercial-grade operating system. This paper describes a case study that examines the scope of the effort involved in migrating device driver code developed for a 'monolithic' OS such as Linux to a microkernel OS such as the QNX Neutrino realtime OS (RTOS). A primary goal of the study was to provide...

Web. http://www.qnx.com/download/feature.html?programid=8086
Score. 4

paper

Implementing Device Drivers - Migrating from Linux to a Microkernel OS

Recent market developments around IP ownership have prompted many companies to investigate the possibility of migrating their Linux code to a commercial-grade operating system. This paper describes a case study that examines the scope of the effort involved in migrating device driver code developed for a 'monolithic' OS such as Linux to a microkernel OS such as the QNX Neutrino realtime OS (RTOS). A primary goal of the study was to provide...

Web. http://www.qnx.com/download/feature.html?programid=8086
Score. 4

overview

Introduction to Linux for Real-Time Control

This report is directed to control engineers and managers. It is intended to provide an overview of real-time operating systems and particularly of the real-time modications to the Linux operating system in enough detail that the reader can make an informed decision whether to commit the resources to evaluate Linux as a real-time operating system.

Web. http://www.aeolean.com/html/RealTimeLinux/RealTimeLinuxReport-2.0.0.pdf
Score. 4

overview

Introduction to Rate Monotonic Scheduling

The purpose of a real-time scheduling algorithm is to ensure that critical timing constraints, such as deadlines and response time, are met. When necessary, decisions are made that favor the most critical timing constraints, even at the cost of violating others. Real-time scheduling is also used to allocate processor time between tasks in soft real-time embedded systems.

Web. http://www.netrino.com/Publications/Glossary/RMA.html
Score. 4

paper

Introduction to VxWorks

An Overview Tutorial of the VxWorks® Real-Time Operating System. VxWorks is a popular real-time multi-tasking operating system for embedded microprocessor systems, designed by Wind River Systems of Alameda, CA. Like Unix and Linux, VxWorks is generally compliant with the IEEE's POSIX (Portable Operating System Interface) standard, version 1003.1b. The current release of VxWorks is version 5.4. VxWorks projects are usually developed in...

Web. http://www.cross-comp.com/pages/embedded/index.php
Score. 4

overview

ITRON Introduction

This is a detailed overview of ITRON, looking at its history, its uses, and its evolution over time. It also looks at its strengths and possible changes for the future.

Web. http://tronweb.super-nova.co.jp/itron.html
Score. 4

overview

LynxOS @ Wikipedia

The LynxOS RTOS is a Unix-like real-time operating system from LynuxWorks (formerly 'Lynx Real-Time Systems'). Sometimes known as the Lynx Operating System, LynxOS features full POSIX conformance and, more recently, Linux compatibility. LynxOS is mostly used in real-time embedded systems, in applications for avionics, aerospace, the military, industrial process control and telecommunications.

Web. http://en.wikipedia.org/wiki/LynxOS
Score. 4

partners program

Microsoft Windows Embedded Partner Program

New Windows Embedded Partner Program (WEP).The WEP was designed to provide a wide range of benefits to companies utilizing the Microsoft Windows Embedded Family of products with the intent of enabling WEP partners to bring their Windows Powered products to market in a more timely fashion.

Web. http://www.mswep.com/
Score. 4

partners program

Microsoft Windows Embedded Partner Program

New Windows Embedded Partner Program (WEP).The WEP was designed to provide a wide range of benefits to companies utilizing the Microsoft Windows Embedded Family of products with the intent of enabling WEP partners to bring their Windows Powered products to market in a more timely fashion.

Web. http://www.mswep.com/
Score. 4

newsgroup

microsoft.public.windowsnt.embedded

Public usenet discussion group devoted exclusively to WindowsXP for Embedded - tell Bill Gates what you think about NT (XP, embedded), post questions, get answers.

Web. news:microsoft.public.windowsnt.embedded
Score. 4

organization

Microsoft: Operating Systems Research Group

Microsoft is busy researching all sorts of stuff about computing. Take a peek at the future of Microsoft-based computing via this URL. Microsoft research: what do you want to monopolize tomorrow? Just kidding, it's actually kind of cool.

Web. http://www.research.microsoft.com/
Score. 4

project

Mobile Linux (MLI)

The Mobile Linux workgroup has as its mission to accelerate adoption of Linux on next-generation mobile handsets and other converged voice/data portable devices and to provide a mobile profile for the LSB. Open Source Projects - The Mobile Linux workgroup does not impose unfunded mandates upon the Linux industry. Rather, it creates requirement specifications based on existing open source (prototype) implementations and works with LF...

Web. http://www.linux-foundation.org/en/Mobile_Linux

Score. 4

conference

MontaVista Vision Linux Conference

You are cordially invited to attend the top embedded Linux developer's conference of the year, hosted by MontaVista Software and sponsored by Freescale Semiconductor, IBM, Intel, and Texas Instruments. You will have the opportunity to choose from more than 30 keynotes and technical sessions. Select the sessions that are best tailored to meet your professional needs. We are also offering Birds-of-a-Feather sessions with top developers,...

Web. http://www.mvista.com/vision/welcome.html
Score. 4

conference

MontaVista Vision Linux Conference

You are cordially invited to attend the top embedded Linux developer's conference of the year, hosted by MontaVista Software and sponsored by Freescale Semiconductor, IBM, Intel, and Texas Instruments. You will have the opportunity to choose from more than 30 keynotes and technical sessions. Select the sessions that are best tailored to meet your professional needs. We are also offering Birds-of-a-Feather sessions with top developers,...

Web. http://www.mvista.com/vision/welcome.html
Score. 4

project

Montavista's Open Source Real-Time Linux Project

The goal of this project is to deliver performance and deterministic real-time responsiveness that is comparable to that of other commercial and proprietary RTOSes. MontaVista expects to lower Linux task preemption latency (worst case) from a range of one to tens of milliseconds, down to tens of microseconds, for time-critical applications. This would be an improvement of at least two orders of magnitude in Linux responsiveness.

Web. http://source.mvista.com/
Score. 4

paper

Moving from a Proprietary RTOS To Embedded Linux

Whether you are planning a move to embedded Linux in the near future or are just considering the level of investment to convert existing applications to run on embedded Linux, this white paper will help you understand the transition process, to assess the challenges and risks involved, and appreciate the benefits realized from such a move. Specifically, this white paper addresses the porting process, API and IPC conversion, and reliability...

Web. http://www.mvista.com/downloads/wp_rtos_to_linux.pdf
Score. 4

paper

Moving from a Proprietary RTOS To Embedded Linux

Whether you are planning a move to embedded Linux in the near future or are just considering the level of investment to convert existing applications to run on embedded Linux, this white paper will help you understand the transition process, to assess the challenges and risks involved, and appreciate the benefits realized from such a move.

Specifically, this white paper addresses the porting process, API and IPC conversion, and reliability...

Web. http://www.mvista.com/downloads/wp_rtos_to_linux.pdf
Score. 4

paper

Moving Legacy Applications to Linux: RTOS Migration Revisited
Whether you are planning a move to embedded Linux or are just considering the investment needed to convert existing applications to run on embedded Linux, this white paper will help you understand the transition process, assess the challenges and risks, and appreciate the benefits realized from such a move. This white paper addresses how to map legacy architectures onto Linux, options for migrated application execution, API and IPC...

Web. http://www.mvista.com/downloads/RTOS_transition.pdf
Score. 4

paper

Moving Legacy Applications to Linux: RTOS Migration Revisited
Whether you are planning a move to embedded Linux or are just considering the investment needed to convert existing applications to run on embedded Linux, this white paper will help you understand the transition process, assess the challenges and risks, and appreciate the benefits realized from such a move. This white paper addresses how to map legacy architectures onto Linux, options for migrated application execution, API and IPC...

Web. http://www.mvista.com/downloads/RTOS_transition.pdf
Score. 4

paper

Moving Legacy Applications to Linux: RTOS Migration Revisited
Whether you are planning a move to embedded Linux or are just considering the investment needed to convert existing applications to run on embedded Linux, this white paper will help you understand the transition process, assess the challenges and risks, and appreciate the benefits realized from such a move. This white paper addresses how to map legacy architectures onto Linux, options for migrated application execution, API and IPC...

Web. http://www.mvista.com/downloads/RTOS_transition.pdf
Score. 4

newsgroup

news:microsoft.public.windowsxp.embedded
Microsoft's official newsgroup on XPe (XP Embedded).

Web. news:microsoft.public.windowsxp.embedded
Score. 4

project

Open Components for Embedded Real-time Applications
OCERA, that stands for Open Components for Embedded Real-time Applications, is an European project that tries to provide Linux with the new real-time functionalities, permiting embedded system developers to access all these benefits. In order to provide these real-time functionalities, the OCERA consortium finally chose the RTLinux executive.

Web. http://www.ocera.org/
Score. 4

hot list

Open Directory RTOS List
Hot list of Real-time Operating Systems from the Open Directory project. Pretty good.

Web. http://dmoz.com/Computers/Software/Operating_Systems/Realtime/
Score. 4

paper

Open Source Licensing: What Every OEM Should Know
On this site, you will find a short "Licensing 101" guide to when the GPL requires you to share code, and when it doesn't. You'll find out exactly which business uses the GPL really affects and which it doesn't. You'll learn how the The Sarbanes-Oxley Act changes the open source landscape by making GPL violations a federal crime. And you'll learn how Loadable Kernel Modules may be the riskiest Linux bet of all.

Web. http://www.wasabisystems.com/gpl/
Score. 4

resource

Operating System.org
relevant and compressed information about operating systems and its developer on this independent web site for the archiving of informations. The specialized project to the documentation of operating systems contains many facts and grows with visitor feedback and steadily improvement and enhancement from the main author. Many screenshots and version information complete the articles with an objective description. The historical time...

Web. http://www.operating-system.org/
Score. 4

overview

OS9 @ Wikipedia
OS-9 is a family of real-time, process-based, multitasking, multi-user, Unix-like operating systems, developed in the 1980s, originally by Microware Systems Corporation for the Motorola 6809 microprocessor. It is currently owned by RadiSys Corporation. The OS-9 family was popular for general-purpose computing and remains in use in commercial embedded systems and amongst hobbyists. Today, OS-9 is a product name used by both a Motorola...

Web. http://en.wikipedia.org/wiki/OS9
Score. 4

personal page

OS-9 Al's Info Pages
The very best PERSONAL site from a fan of OS-9.

Web. http://www.os9al.com/index.shtml
Score. 4

personal page

OS-9 Samba File Manager

OS-9 Samba File Manager is a software solution that enables OS-9 (Microware Ê) users to participate in Microsoft networks with file and print services. It provides OS-9 users the ability to mount shared directories or printers on Windows NT (Server and Workstation), Windows 95/98, and Windows for Workgroups computers or compatible systems running TCP/IP.

Web. http://www.os9samba.com/
Score. 4

resource

Paul Yao Windows Technical Papers

* Articles and Book Review - Links to Book Review, articles in MSDN Magazine and other publications on topics of interest to Windows CE and Pocket PC programmers. * Presentations - Download Powerpoint Presentations and presentation sample code from industry conferences. * Tools - Developer tools to make programmers more productive. * White Papers * Links- Links of interest to Windows CE programmers.

Web. http://www.paulyao.com/resources/
Score. 4

overview

POSIX @ Wikipedia

POSIX (IPA: [?p?s?ks]) or 'Portable Operating System Interface'[1] is the collective name of a family of related standards specified by the IEEE to define the application programming interface (API) for software compatible with variants of the Unix operating system. Originally, the name stood for IEEE Std 1003.1-1988, which as the name suggests, was released in 1988. The family of POSIX standards is formally designated as IEEE 1003 and the...

Web. http://en.wikipedia.org/wiki/POSIX
Score. 4

organization

POSIX Certification web site

Welcome to the POSIX Certification web site POSIX® Certified by IEEE and The Open Group is for products meeting the IEEE POSIX standards. The following documents should be read and understood prior to certification, since you will be required to agree to them during that process . . .

Web. http://get.posixcertified.ieee.org/
Score. 4

overview

POSIX conformance is worth more than POSIX compliance

POSIX conformance is what real-time embedded developers are usually looking for. POSIX conformance means that the POSIX.1 standard is supported in its entirety. In the case of the LynxOS real-time operating system, the routines of the POSIX.1b and POSIX.1c subsets are also supported.

Web. http://www.lynuxworks.com/products/posix/posix.php3
Score. 4

resource

QNX Beginners' Pages

This is a collection of examples, tutorials, and notes on administrating and programming QNX Neutrino and Photon. The tutorials are deliberately kept very simple, and focus on the topic being explained.

Web. http://psy.swansea.ac.uk/staff/Carter/QNX/
Score. 4

overview

Rate Monotonic Analysis - Overview
Rate Monotonic Analysis (RMA) is a collection of quantitative methods and algorithms that allows engineers to specify, understand, analyze, and predict the timing behavior of real-time software systems, thus improving their dependability and evolvability. Sections in the paper include technical detail, usage considerations, maturity, costs and limitations, and more.

Web. http://www.sei.cmu.edu/str/descriptions/rma_body.html
Score. 4

tool

RCOS.java
RCOS.java is a tool designed to help people understand the inner workings of an operating system. RCOS.java is an animated, multi-tasking operating system running on simulated hardware.

Web. http://rcosjava.sourceforge.net/
Score. 4

organization

Real Time Linux Foundation, Inc.
The natural consequence of these turns of events was a discussion and decision at the real time Linux workshop in Orlando 2000 to create the Real Time Linux Foundation as a balanced, impartial nonprofit organization which is vendor neutral, but offers an entry point to real time Linux for documentation, discussion, sample applications and vendors of real time Linux related services.

Web. http://www.realtimelinuxfoundation.org/
Score. 4

paper

Real Time or Real Linux? A Realistic Alternative
In this paper, we look at an alternate approach - using a POSIX-based RTOS designed specifically for embedded systems - that not only allows Linux developers to keep their programming model, but also maintains the key advantages of Linux's open source model.

Web. http://www.qnx.com/download/feature.html?programid=8085
Score. 4

paper

Real Time or Real Linux? A Realistic Alternative
In this paper, we look at an alternate approach - using a POSIX-based RTOS designed specifically for embedded systems - that not only allows Linux developers to keep their programming model, but also maintains the key advantages of Linux's open source model.

Web. http://www.qnx.com/download/feature.html?programid=8085

Score. 4

paper

Real Time or Real Linux? A Realistic Alternative
In this paper, we look at an alternate approach - using a POSIX-based RTOS designed specifically for embedded systems - that not only allows Linux developers to keep their programming model, but also maintains the key advantages of Linux's open source model.

Web. http://www.qnx.com/download/feature.html?programid=8085
Score. 4

paper

Realtime and Embedded How To (Linux)
This Guide explains the fundamentals of real-time and embedded programming and operating systems (focusing mostly on their differences with general purpose operating systems such as Linux). Everything is illustrated with free software examples, mainly RTAI.

Web. http://people.mech.kuleuven.ac.be/~bruyninc/rthowto/
Score. 4

organization

Real-time and Embedded Systems Forum, Open Group
The vision of the Real-time and Embedded Systems is to grow the marketplace for standardized real-time and embedded systems, through the deployments of standards and associated certification programs. Focuses: Java, Safety-Critical, Security.

Web. http://www.opengroup.org/rtforum/
Score. 4

project

Real-Time Linux Wiki
Welcome to the RT Wiki, the Wiki Web for the CONFIG_PREEMPT_RT community, and real-time Linux in general. Please support us and help to extend this wiki. Thank you! More about RT Wiki Documentation CONFIG_PREEMPT_RT Patch RT PREEMPT HOWTO High Resolution Timers (hrtimers) Utilities RT Patch Manager RT Watchdog OSADL Real-Time Live Linux CD O(1) Memory Allocator More Utilities Benchmarks, and Test Cases IBM...

Web. http://rt.wiki.kernel.org/index.php/Main_Page
Score. 4

overview

Real-time Mantra
Real-time Mantra highlights the key issues in real-time system design. It also cover real-time design patterns and issues in complex software design. The mantras have been divided into the following categories. Has a nice sections on issues like C and Assembly language programming, object-oriented programming, and TCP/IP.

Web. http://www.eventhelix.com/RealtimeMantra/
Score. 4

university

Real-Time Systems Laboratory - Univ. of Illinois

Established more than 20 years ago, the Real-Time Systems Laboratory has been pioneering the area of real-time computing. For more on the history of the RTSL, download and view rtsl_history.pdf (112KB). The Real-Time Systems Laboratory (RTSL) performs research on all aspects of real-time computing systems.

Web. http://www-rtsl.cs.uiuc.edu/
Score. 4

paper

RTOS Real-Time Performance vs. Ease Of Use
Real-time operating system (RTOS) vendors often make claims of 'fast real-time performance,' and 'rapid real-time response,' to convince developers to use their RTOS for a given application. The implication is that 'faster is better,' and sub-microsecond interrupt response and context switch times are offered as compelling evidence.

Web. http://www.rtos.com/page/imgpage.php?id=208
Score. 4

paper

RTOS Real-Time Performance vs. Ease Of Use
Real-time operating system (RTOS) vendors often make claims of 'fast real-time performance,' and 'rapid real-time response,' to convince developers to use their RTOS for a given application. The implication is that 'faster is better,' and sub-microsecond interrupt response and context switch times are offered as compelling evidence.

Web. http://www.rtos.com/page/imgpage.php?id=208
Score. 4

overview

RTOS State of the Art Analysis (OSEK)
The objective of this workpackage is to make a study of the state of the art of real-time technology which is made available by the research community, and to determine what types of mechanisms actually turn out to be most useful for real-time applications. In concrete, this workpackage will analyse the real-time operating systems (RTOS) features and extract the main characteristics that will be included in the OCERA development.

Web. http://www.mnis.fr/opensource/ocera/rtos/book1.html
Score. 4

overview

RTOS State of the Art Analysis (OSEK)
The objective of this workpackage is to make a study of the state of the art of real-time technology which is made available by the research community, and to determine what types of mechanisms actually turn out to be most useful for real-time applications. In concrete, this workpackage will analyse the real-time operating systems (RTOS) features and extract the main characteristics that will be included in the OCERA development.

Web. http://www.mnis.fr/opensource/ocera/rtos/book1.html
Score. 4

overview

RTOS State of the Art Analysis
The objective of this workpackage is to make a study of the state of the art of real-time technology which is made available by the research community, and to determine what types of mechanisms actually turn out to be most useful for real-time applications. In

concrete, this workpackage will analyse the real-time operating systems (RTOS) features and extract the main characteristics that will be included in the OCERA development.

Web. http://www.mnis.fr/ocera_support/rtos/book1.html
Score. 4

personal page

Sean Liming's Windows XP Embedded Center

Because Windows Embedded is always changing and developers are constantly looking for information, the XPE Center was created to help developers find the needed resources for their XP Embedded project. The XPE Center provides information on the education material, training, consulting services, and resources that are available for XP Embedded.

Web. http://www.seanliming.com/
Score. 4

paper

Selecting the Right RTOS, A Comparative Study

With over 100 different RTOS manufacturers in the market today, how do we know which RTOS is right for our application? This paper seeks to develop a framework for the reader to understand the key differences between the various offerings to aid in this selection process.

Web. http://www.theptrgroup.com/pdf/ESCclass200-220.pdf
Score. 4

personal page

Software Safety Home Page

While I concentrate on Software Safety on this site it is important to note that no software works in isolation. The entire system must be designed to be safe. The system contains the software, hardware, the users, and the environment. All must be given consideration when developing software. All parts of the system must be safe.

Web. http://www.softwaresafety.net/
Score. 4

organization

Software Technology Support Center (U.S. Air Force)

In 1987, the U.S. Air Force selected Ogden Air Logistics Center (OO-ALC), Hill Air Force Base, Utah, to establish and operate its Software Technology Support Center (STSC). It was chartered to be the command focus for proactive application of software technology in weapon, command and control, intelligence and mission-critical systems. Publishes <B>Crosstalk, a monthly journal on software and software security.

Web. http://www.stsc.hill.af.mil/
Score. 4

personal page

Steven Bradley's Home Page

Real-Time and Concurrency This course is taught by Steven Bradley and Zhaohui Luo, but this page deals only with the real-time part given by Steven Bradley. Teaching At the moment I am involved in four modules computer systems programming and data structures Business and Professionalism Real-time computing

Web. http://www.dur.ac.uk/s.p.bradley/
Score. 4

project

The Embedded Linux Workshop

The Embedded Linux Workshop is an open source project put together for the book 'Embedded Linux' by John Lombardo. The workshop makes it easy to build embedded Linux applications quickly and easily.

Web. http://sourceforge.net/projects/elw
Score. 4

project

The gRMA project

. gRMA is a graphical tool for performing Rate Monotonic Analysis. It is being developed by Sam Tregar and will be released under the GNU General Public License for all to use and modify freely. It might even become an official GNU project!

Web. http://www.tregar.com/gRMA/
Score. 4

institution

The IEEE-CS TC-RTS Home Page

This is the WWW Home Page of the IEEE Computer Society, Technical Committee on Real-Time Systems (IEEE-CS TC-RTS).

Web. http://www.cs.bu.edu/pub/ieee-rts/Home.html
Score. 4

association

The Linux Foundation

The Linux Foundation is a nonprofit consortium dedicated to fostering the growth of Linux. The Linux Foundation promotes, protects and standardizes Linux by providing unified resources and services needed for open source to successfully compete with closed platforms...

Web. http://www.linux-foundation.org/
Score. 4

paper

The Truth About Windows Real-Time Architectures

Ring 0 or Ring 3. This whitepaper discusses the essential differences between the architectures and summarizes the major benefits of Ring 0 based designs. In addition to discussing the obvious benefits of performance and decreased development time with a Ring 0 solution, this white paper will also discuss the issues and downsides of Ring 3 memory protection.

Web. http://www.ardence.com/embedded/products.aspx?ID=539
Score. 4

archive

TimeSys Linux Webinars

TimeSys is a unique vendor in the 'embedded Linux' marketplace with their unique strategy of 'LinuxLink by TimeSys' - a continuously updated, Web-based resource for embedded Linux development, delivering subscription-based access to hundreds of cross-compiled packages ... Click here for online learning and seminars from TimeSys on embedded Linux.

Web. http://www.timesys.com/webinars/index.htm
Score. 4

overview

Tri-Pacific Software: RMA Backgrounder
With RAPID RMA (formerly PERTS) users can study the effects and relationships of various parameters in the system that directly affect the schedulability of the entire task set. If the system is predictable, it can be formally analyzed (See Figure 1). RMA allows a user to determine ahead of time whether the system will meet its timing requirements.

Web. http://www.tripac.com/html/tech-bkgd-rma.html
Score. 4

libraries

uClibc -- a C library for embedded systems
C library for developing embedded Linux systems. It is much smaller then the GNU C Library, but nearly all applications supported by glibc also work perfectly with uClibc. Porting applications from glibc to uClibc typically involves just recompiling the source code. uClibc even supports shared libraries and threading.

Web. http://www.uclibc.org/
Score. 4

archive

UNC DiRT group's Realtime Selected Papers
DiRT stands for DIstributed and Real-Time systems. Our purpose in life is to understand how to better build, you guessed it, distributed real-time systems. This site provides a selection of papers on realtime, object-oriented, multimedia and other topics of interest in the realtime computing area. Most are provided in the PS format.

Web. http://www.cs.unc.edu/Research/dirt/
Score. 4

resource

Unofficial Dr. DOS Site
If you are a DOS beginner, this file might help you: http://docdos.net/install If you have got problems, questions etc. concerning DOS send an e-mail to webmaster@drdos.net, use the DR-DOS Forum (DR-DOS Forum in German language: here) or subscribe to the OpenDOS mailing list (send an e-mail to listserv@delorie.com with the text add opendos or add opendos-digest in the body).

Web. http://www.drdos.net/
Score. 4

paper

Using POSIX for Embedded Systems
More than ever, commercial, government, and military organizations are demanding that developers use POSIX interfaces. The question is, how much of the POSIX standard must your project support? POSIX is so large, and has so many optional components, that few applications need everything it offers. This paper introduces basic concepts of applications portability, and explores the benefits of using POSIX as a standard. . .

Web. http://www.qnx.com/download/feature.html?programid=9870
Score. 4

overview

Virtualization @ Wikipedia

In computing, virtualization is a broad term that refers to the abstraction of computer resources. One useful definition is 'a technique for hiding the physical characteristics of computing resources from the way in which other systems, applications, or end users interact with those resources. This includes making a single physical resource (such as a server, an operating system, an application, or storage device) appear to function as...

Web. http://en.wikipedia.org/wiki/Virtualization
Score. 4

paper

Virtualization for Embedded Systems

Virtualization has been a hot topic in the enterprise space for quite some time, but has recently become an important technology for embedded systems as well. It is therefore important for embedded-systems developers to understand the power and limitations of virtualization in this space, in order to understand what technology is suitable for their products. This white paper presents an introduction to virtualization technology in...

Web. http://portal.ok-labs.com/_assets/downloads/OK_Virtualization.pdf
Score. 4

overview

VxWorks @ Wikipedia

VxWorks is a Unix-like real-time operating system made and sold by Wind River Systems of Alameda, California, USA. Like most RTOSes, VxWorks includes a multitasking kernel with pre-emptive scheduling and fast interrupt response, extensive inter-process communications and synchronization facilities, and a file system. Newer versions of VxWorks now support pSOS system calls since Wind River now owns both RTOSes. Major distinguishing...

Web. http://en.wikipedia.org/wiki/VxWorks
Score. 4

organization

Windows Embedded Developers Interest Group

The Windows Embedded Developers' Interest Group is dedicated to the exchange of ideas around Microsoft Windows embedded and mobile software development. Do you want to get more involved with us? If you would like to chat with other members and share your views, experiences, and ideas about embedded programming, please register with us and click here.

Web. http://www.we-dig.org/
Score. 4

organization

Windows Embedded Developers Interest Group

The Windows Embedded Developers' Interest Group is dedicated to the exchange of ideas around Microsoft Windows embedded and mobile software development. Do you want to get more involved with us? If you would like to chat with other members and share your views, experiences, and ideas about embedded programming, please register with us and click here.

Web. http://www.we-dig.org/
Score. 4

overview

Windows XP Embedded @ Wikipedia

Windows XP Embedded, or XPe, is the componentized version of Microsoft Windows XP Professional. XPe is based on the same binaries as XP Professional, but XPe is marketed towards developers for OEMs, ISVs and IHVs that want the full Win32 API support of Windows but without the overhead of Professional. It runs existing Windows applications and device drivers off-the-shelf on devices with at least 32MB Compact Flash, 32MB RAM and a P-200...

Web. http://en.wikipedia.org/wiki/Windows_XP_Embedded
Score. 4

newsgroup

WindowsCE Development Discussion Group

Discussion of Microsoft Windows CE development issues. Both operating system and application development are appropriate topics for this list. This list is intended for discussion of problems that are specific to Windows CE and is not for help with general Win32 programming problems.

Web. http://groups.yahoo.com/group/windowsce-dev/
Score. 4

portal

WindowsForDevices.com

WindowsForDevices.com is an independent, embedded-oriented, online meeting place and technical news site for the Windows Embedded developer community. The focus of the site is to advance the use of Microsoft operating systems within devices and embedded systems while educating developers about Windows Embedded technologies.

Web. http://www.windowsfordevices.com/
Score. 4

portal

WindowsForDevices.com

WindowsForDevices.com is an independent, embedded-oriented, online meeting place and technical news site for the Windows Embedded developer community. The focus of the site is to advance the use of Microsoft operating systems within devices and embedded systems while educating developers about Windows Embedded technologies.

Web. http://www.windowsfordevices.com/
Score. 4

resource

XPeFiles.com

We are happy to launch this new version of the XPeFiles site. There have been numerous requests to add items like forums, upload management capabilities, automated password retrieval, and news. We are happy to say that it's all here! You'll notice a few oddities as you browse the new site. If you were a contributor on the previous site you still own your components.

Web. http://www.xpefiles.com/
Score. 4

Appendix J: RTOS Books

In this Appendix, we identify interesting and useful books published on RTOSes in the last three years. In a few cases, we list older books that are "classics."

book

Building a Real Time Operating System: RTOS from the Ground Up
Real-time Operating Systems are an increasingly important tool, as integration of networking functionality, reliability, modularity, and complex multitasking become ever-more prominent concerns for embedded developers. This is because real-time operating systems (RTOSs) enable precise timing of multiple tasks on a much stricter schedule than traditional operating systems. This makes them a key ingredient in the successful deployment of...

Web. http://www.amazon.com/exec/obidos/ASIN/0750683791/Eg3communicationA/
Score. 3

book

Embedded Multitasking
Embedded Multitasking is the ability to make an embedded system appear as if it's performing numerous functions at one time. For example, you'd like to be able to check the email on your palm pilot while looking at your schedule while changing the resolution on your screen. In order to accomplish this, real-time interaction between the software and the microcontroller is essential. This is difficult because most normal microcontrollers can...

Web. http://www.amazon.com/exec/obidos/ASIN/0750679182/Eg3communicationA/
Score. 3

book

Embedded Software: The Works
This one book has an amazing breadth of coverage, undertaking all the key subjects embedded engineers need to understand in order to succeed, including Design and Development, Programming, Languages including C/C++, and UML, Real Time Operating Systems Considerations, Networking, Programmable Logic and much more. For those in the field who are looking to broaden their professional skill-sets in order to advance . . .

Web. http://www.amazon.com/exec/obidos/ASIN/0750679549/Eg3communicationA/
Score. 3

book

Handbook of Real-Time and Embedded Systems
Real-time and embedded systems are essential to our lives, from controlling car engines and regulating traffic lights to monitoring plane takeoffs and landings to providing up-to-the-minute stock quotes. Bringing together researchers from both academia and industry, the Handbook of Real-Time and Embedded Systems provides comprehensive coverage of the most advanced and timely topics in the field. The book focuses on several major areas of...

Web. http://www.amazon.com/exec/obidos/ASIN/1584886781/Eg3communicationA/
Score. 3

book-classic

Introduction to Real Time Systems : From Design to Multitasking With C/C
This text introduces the nature of real-time, concurrent, distributed systems, presenting a specific set of techniques for designing and implementing such systems. It develops a

"systems way of thinking" about software that is intended to serve readers throughout their careers.

Web. http://www.amazon.com/exec/obidos/ASIN/0136060706/Eg3communicationA/
Score. 3

book-classic

MicroC/OS-II

Learn the inner workings of an RTOS with MicroC/OS-II - a completely portable, ROMable, preemptive real-time kernel. Since 1992 MicroC/OS has been used by thousands of developers worldwide in applications ranging from cameras to ATMs. You can use it in your applications, too! newly revised - June, 2002!

Web. http://www.amazon.com/exec/obidos/ASIN/1578201039/Eg3communicationA/
Score. 5

book

Real-Time Embedded Components and Systems (Computer Engineering)

Due to the rapidly expanding market for digital media services and systems, there is a growing interest in real-time systems. Real-Time Embedded Systems and Components is a much-needed resource addressing this field for practicing engineers and students, particularly engineers moving from best-effort applications to hard or soft real-time applications. The book is written to teach practicing engineers how to apply real-time theory to the...

Web. http://www.amazon.com/exec/obidos/ASIN/1584504684/Eg3communicationA/
Score. 3

book

Real-Time Systems Development

Real-time Systems Development is a text for computing students who want to understand more about the development of software for real-time applications, involving concurrent programming, multi-tasking, data i/o and embedded processors. The book has been written to cover single semester final year undergraduate options or MSc modules in the area of real-time systems design and implementation. Assuming a certain level of general systems...

Web. http://www.amazon.com/exec/obidos/ASIN/0750664711/Eg3communicationA/
Score. 3

book-classic

Real-Time Systems

A textbook written for a technical elective for seniors and graduate students in computer science or computer engineering. Liu (electrical engineering, Massachusetts Institute of Technology) builds on other operating systems knowledge and covers techniques for scheduling, resource access control, and validation that are used in real-time computer or communication systems.

Web. http://www.amazon.com/exec/obidos/ASIN/0130996513/Eg3communicationA/
Score. 2

book

Simple Real-time Operating System: A Kernel Inside View for a Beginner
The first motivation for writing this book is to teach my colleagues about RTOS (Real-Time Operating System) concepts and the proprietary RTOS developed for our project by me. I realized that the same material with some extension can be useful to most embedded engineers who are beginners in learning about RTOS. I came across many embedded engineers who are not comfortable to use real-time operating system. A significant fraction of these...

Web. http://www.amazon.com/exec/obidos/ASIN/1425117821/Eg3communicationA/
Score. 3

book

Symbian OS Internals : Real-time Kernel Programming
Take a look inside Symbian OS with an under-the-hood view of Symbian's revolutionary new real-time smartphone kernel Describes the functioning of the new real-time kernel, which will become ubiquitious on Symbian OS phones in the next 5-10 years Will benefit the base-porting engineer by providing a more solid understanding of the OS being ported Contains an in-depth explanation of how Symbian OS drivers work. Device drivers have changed...

Web. http://www.amazon.com/exec/obidos/ASIN/0470025247/Eg3communicationA/
Score. 3

book

Symbian OS Platform Security : Software Development Using the Symbian OS Security Architecture
Symbian OS is an advanced, customizable operating system, which is licensed by the world's leading mobile phone manufacturers. The latest versions incorporate an enhanced security architecture designed to protect the interests of consumers, network operators and software developers. The new security architecture of Symbian OS v9 is relevant to all security practitioners and will influence the decisions made by every developer that uses...

Web. http://www.amazon.com/exec/obidos/ASIN/0470018828/Eg3communicationA/
Score. 2

book

What Every Engineer Should Know About Developing Real-Time Embedded Products
You can find them in your wristwatch or MP3 player; they perform specific functions in washing machines, traffic lights, and even pacemakers. Embedded systems are pervasive, ubiquitous, and widespread throughout our daily lives. Developing these real-time embedded products requires an understanding of the interactions between different disciplines, such as circuit design, power, cooling, packaging, software, and human interface. This volume...

Web. http://www.amazon.com/exec/obidos/ASIN/0849379598/Eg3communicationA/
Score. 3