

# SELECTING AN EMBEDDED RTOS



## FEATURED INTERVIEW:

EXCERPTED FROM [WWW.EG3.COM](http://WWW.EG3.COM)

*Prepared by:*

eg3.com

Jason McDonald, Senior Editor

eg3.com

tel : 510.713.2150

email : [info@eg3.com](mailto:info@eg3.com)

web : <http://www.eg3.com>



## MICRO DIGITAL: TECHNICAL CRITERIA FOR MAKING THE RTOS CHOICE

15 December 2008: Technical Criteria for Making the RTOS Choice

INTERVIEWEE. RALPH MOORE  
PRESIDENT  
TEL. 714 427 7333  
EMAIL. RALPHM@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. I have over 30 years experience in the embedded systems industry starting as a logic designer, then a programmer/consultant, ending as founder and President of Micro Digital. I architected the SMX<sup>®</sup> RTOS kernel and wrote some of the code for it in 1989. Since then I have progressively done more sales, marketing, and business management and less development work.

**Q. As you know, the 2009 Guide has two sorts of interviews - product Q&A interviews and interviews like this one on “key topics.” But just to help our readership out, tell us in a paragraph or two what Micro Digital offers in the RTOS space and what your unique “value proposition” is as a company. (Interested parties can refer to your product Q&A interview in the guide or your website for more details.)**

A. SMX is a modular RTOS consisting of the smx hard real-time multitasking kernel, TCP/IP stack with numerous protocols, USB host and device stacks with numerous class and function drivers, FAT and flash file systems with support for numerous media, GUI, and other products. We focus primarily on ARM, ColdFire, PowerPC, and x86 processors. For these, we endeavor to offer out-of-the-box solutions that are fully integrated with popular tools and run immediately on processor vendor evaluation boards. We offer a very large selection of free evaluation kits and demos that allow prospective users to verify our value proposition of giving them full integration and a quick start. To make out-of-the-box solutions possible, we offer numerous BSPs, controller drivers, well-written manuals, and good support.

**Q. We are talking today about the “technical criteria” that one should use in making the RTOS choice. What are the most important factors in your opinion from a technical perspective?**

A. In the current environment of rapid technical advancement and increasing product feature demand, the RTOS is becoming the foundation that holds everything together. Embedded system programmers need a reliable base to build upon. It is no longer feasible to be expert in all facets of a project. TCP/IP networking is complicated, USB is complicated, file systems are complicated, user interfaces are complicated, etc.; there is too much to learn. At the same time, new processors and new revisions of tools have bugs. Many things just do not work right, and we have to compensate for them..

However, commercial RTOSs are a prime example of code reuse. For example, our USB stacks have been shipped to nearly 100 customers. In their current state, they reflect not only the work of our staff, but also the work of customers who found things that did not work – in our code and elsewhere. These bugs have been fixed, workarounds have been found, or we advise not using certain hardware. All of this is what the RTOS customer is getting for his money.

**Q. All that brings up the big issue of how a developer should go about “differentiating” their various RTOS choices. What sorts of questions would you suggest he ask the RTOS vendors on his short list, as he goes about trying to figure out how they are different and how their differences might help his own application?**

A. Most RTOS shoppers start with a wish list of features and a short list of processors. Their first step is to make a short list of RTOSs that fit within these boundaries. If an RTOS vendor truly has what is needed, it should be possible to get an evaluation kit and demos that demonstrate this. Some programmers, or their bosses, think that they do not have time to evaluate RTOSs – it is faster to fill in spreadsheets. I think that this is a big mistake. I always try to get prospective customers to try our evaluation kits and demos. Running an evaluation kit will give you the “feel” of it, and it is a great learning opportunity. It is a chance to learn not only about the RTOS, but also about the tools, and even the processor, itself. We humans tend not to learn unless we are forced to learn by actually doing. Things learned in this early phase of a project may change the course of the project and the time allotted for it. Hence, I recommend that programmers spend more time doing than asking.

**Q. Isn't there a dilemma in choosing an RTOS that you don't really know how it works until you *actually* use it? How do you recommend a developer go about “trying before buying” a few of the RTOSes that make his short list? Any particular technical tests that you think he should conduct?**

A. The foregoing answers the main question. As far as what to test, I recommend that a developer actually write some of his application code and debug it so he can determine if he is comfortable with the processor, tools, and RTOS. Now is the time to pick something else, if not. Another thing I recommend is that he tests performance. Most of us are accustomed to 3 GHz PC's. Hence we do not realize how inefficient are TCP/IP, USB, and other protocols. These protocols were not written for bitty embedded processors running at glacial 50 MHz rates! Many prospective customers, whom I encounter, have unrealistic performance expectations. A third area to check is RAM. RAM costs 4 to 6 times the silicon area as flash. Hence processor vendors are very miserly with it. They frequently load processors with too many peripherals for the on-chip RAM. Again, the common protocols were not designed for efficient RAM usage.

**Q. Designers and pundits often fixate on “real-time performance” yet there is a lot more than that to the RTOS choice. Beyond “hard” real-time, what sorts of other software areas do you think can be critically important? What about “development tools” in particular?**

A. I think it is very important that the RTOS do a lot of error checking for the user. This helps to find application programming errors quickly. Since C is strongly pointer-oriented, it also helps if the RTOS protects pointers by embedding them in control blocks and range-checking them before using. Good kernel awareness is very helpful during

system debugging. It should not only present convenient views of system objects and provide task-sensitive breakpoints, but it should also present graphical event timelines and stack and processor usage graphs. We have had many users tell us that timeline graphs (which show what is happening, when, and why) have saved them countless hours of debugging. A stack usage graph alerts a user if a task is overflowing or about to overflow – a common problem in multitasking systems. Processor usage identifies tasks that are hogging the machine. We have recently begun expanding kernel awareness to include network, USB, and file system awareness. The importance of these is that it is often difficult to deduce the cause of systemic errors at the code debug level. (Of course, systemic errors are the ones that are most obvious to customers and bosses.)

**Q. What about “middleware” and Internet networking/networking software like TCP/IP stacks. Many of the RTOS vendors offer these types of software as well. What should a developer look for? How should he compare and contrast the offerings of different vendors?**

A. It is important to look for complete, integrated solutions. Integrating pieces from different vendors, though sometimes necessary, can be disastrous. For this reason, we and most other RTOS vendors try to offer the most complete solutions that we can. With regard to networking, USB, file systems, and GUIs, the breadth of the offering is a good indication of how seasoned the products are. For example, if a TCP/IP stack is supported by numerous Ethernet controller drivers, runs on many processors, and has many supportive protocols, such as FTP, HTTP, SNMP, etc, and then it is probably a well-proven, reliable stack. Attempting to cut corners on middleware, such as using free or cheap stacks, can cause irreparable harm to a project – possibly even failure. Of course compliance with industry standards is also essential for interoperability with the rest of the world – lack of which is another product killer.

**Q. What about GUI (Graphical User Interface) development tools? I noticed that Micro Digital offers the *PEG Family*, for instance.**

A. The *PEG family*, which is produced by Swell Software, is a prime example of the foregoing. The individual *PEG* products are well seasoned and well accepted in the embedded systems industry. They offer extensive display and input device drivers and come with tools that make designing user interfaces much easier. Though prices have been going up with success, *PEG* products are still very good buys relative to the cost of programming labor and the cost of coming to market with an inferior user interface due to using an inferior tool. *SMX* was the first RTOS supported by *PEG*. We are proud to have been partners with Swell for over 10 years. Our companies work closely together to assure a good experience for our joint customers.

**Q. Speaking from your own experience at Micro Digital, can you tell us some other ways how you differentiate your RTOS, *SMX*®, from other commercial competitors? In particular you mentioned in the pre-interview, that *LSRs* (Link Service Routines) and “one shot tasks” were areas of differentiation.**

A. *SMX*, as opposed to the many generic multitasking kernels flooding the market, is a feature-rich kernel. There is an unfortunate belief, in our industry, that all RTOS kernels are equivalent. Thus, one might as well pick up a free kernel as to pay for a good one. I think that people who espouse this belief are ignoring the fact that what is not in the kernel ends up in the application as green code. Since every new line of code is expensive to produce and to debug, this makes very little sense. It is greatly to the benefit of a

project to use proven code rather than green code. Most commercial RTOSs have been proven in hundreds of applications. Hence, I think it makes sense for a potential user to carefully evaluate each feature of a prospective kernel in order to ascertain how it might help him with the work at hand.

The LSRs and one-shot task support of smx are good cases in point. A primary goal of smx was to minimize interrupt latency. Unlike simple kernels which disable interrupts in every system call, smx disables interrupts in only a few short places in the scheduler. LSRs are a key to this. One-shot task support allows many tasks to share a few stacks from a common pool. This can greatly reduce RAM usage for systems with mutually exclusive tasks (e.g. a state machine). These and other special smx features are discussed in white papers on our website.

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