

RTOS: Buy or Roll Your Own?

The embedded systems industry is as strong as ever. More and more organizations are designing and implementing an ever-expanding range of creative and useful products that require microprocessors to function. As market demand increases, engineering departments are facing important decisions regarding the real time software tools they will use to implement designs quickly and efficiently. Occasionally, a question will arise concerning the implementation of a Real Time Operating System (RTOS). Should we buy or roll our own? This paper attempts to clearly put forth the reasons why an organization should never consider re-inventing the wheel, no matter how exciting such a challenge may be (incorrectly) perceived to be.

What are you looking for in an RTOS ?

Generally, engineers would agree that any RTOS they use should have the following characteristics:

- Availability – the RTOS should run on the target processor(s) and work seamlessly with the associated tool suites that will be used.
- Efficiency – the RTOS should require little overhead and be blazingly fast.
- Reliability – the RTOS should be tested and proven over hundreds of designs.
- Functionality – the RTOS should offer the functions that are required for the application now and in the foreseeable future.
- Sustainability – the RTOS should be meticulously maintained to fix current and future problems and/or needs.
- Economy – the RTOS should not put burdensome costs on the project.
- Quality – the RTOS should be well documented and fully supported to enable engineers to extract the most value from the system.

What is most interesting about the above list is that an in-house developed system will almost certainly never satisfy all of these important requirements. Only a commercially

produced and supported RTOS can offer such breadth of features. This will be more fully explored below.

Is an RTOS different from other engineering tools?

Would your department consider developing its own CAD or C Compiler programs? Of course not, there are too many established vendors producing better code than any single product team could ever hope to match. This is exactly the case with an RTOS, as well. Vendors have been producing *and refining* powerful real time software tools for tens of thousands of clients over many, many years.

Do you recall reading any recent engineering journals where it is recommended that engineers create their own tools? Of course not, because the engineering software market has matured well beyond the “do-it-yourself” stage. That’s why there are regular articles in most technical journals regarding the use of commercially purchased tools – everyone has recognized the benefit of *using* software tools, not *writing* them.

Can you save money by rolling your own?

An RTOS like CMX-RTX™ is composed of sophisticated functionality that took a full-time staff of professional developers (with a specially selected knowledge base of hardware, software, computer science, programming, etc.) over three years to develop and test for commercial release. Millions of dollars and many man-years of investment costs have been poured into the software since its inception. Is your department preparing to expend similar resources for this undertaking? Buying an RTOS allows you to share (with thousands of the vendor’s other clients) in the amortization of the software research, development, testing, and support costs.

Further, have you ever seen a complex software project of this scope finish on time and under budget at your organization? It is impossible to design, develop, implement, test, and support a worthwhile RTOS for even close to the price of a commercial, off-the-shelf software system. It is even more impossible to begin using it as quickly (most vendors can ship the same day an order is received.) Are you prepared to pay more, wait longer and receive less?

Your project is too important to risk!

Engineering development is the lifeblood of a product company. If your design and development team is handicapped with less than the best tools or distracted by non-core competency programming work, the entire organization will suffer from reduced competitiveness. Your job is to design and develop your company’s products and in today’s markets you must remain tightly focused on that. To do less is to imperil the company and to ignore your fiduciary responsibility. You must be sure that you are using the best technology you can afford, and not settle for less. A small increase in

productivity and resultant faster time to market (from better and more sophisticated software tools) will quite easily justify the price of a commercial RTOS like CMX-RTX.

And what if the in-house RTOS development project fails? It is a little known fact that *most* software development projects fail. Can you and your company risk the consequences of project failure stemming from a decision to roll your own RTOS?

Learn from others.

Right at this moment, probably no one knows better than your team what is needed, in terms of software tools, to increase productivity and efficiency; however, what will your requirements be next year and beyond when the product lines take a dramatic new turn? The CMX Company's large and diverse user base (which ranges from small design shops to large multi-national corporations) positions us with a unique view of the embedded systems world that is difficult to be matched. Input from this rich variety of users not only makes our RTOS extremely productive and efficient, but allows for cross-fertilization of ideas (through user suggested improvements, etc.) that will never occur in an isolated engineering department struggling to keep its in-house software running and products moving out the door.

This wealth of experience has been engineered into CMX-RTX over the years and allows our users to benefit from the collective wisdom of our clients.

Benefit from functionality.

CMX-RTX provides a wealth of functionality to our users including, task management, timer management, memory management, resource management, event management, system management, message management, queue management, and semaphore management. Of course, these top-level categories offer numerous command functions to suit the needs of even the most demanding applications. But don't worry about our code size. All of our functions are stored in a library so that only those functions you require for the particular application will be compiled in to the actual code.

Beyond this functionality, however, CMX offers additional modules such as, CMXBug (interactive debugger), CMXTracker (real time flow analyzer), CMX TCP/IP (networking stack), PCProto-RTX (PC-based development platform), and CMX-Tiny+ (a special implementation for 8-bit processors utilizing only onboard RAM.) The system also supports dozens of compilers and hundreds of target microprocessors.

Which of the above will you be giving up with the in-house developed system? Are you really sure that you will *never* need any of these important capabilities? Are you sure that your target or compiler will never change? Are you prepared to pass up all of the enhancements and completely new functionality CMX has planned for the next few years? CMX-RTX is a dynamic, growing product with ambitious plans for its state-of-the-art technology.

What about the future?

What percentage of your budget for this project will be devoted to research and development of new RTOS features and capabilities over the next few years? Had you developed this program a few years ago, would you have recognized the importance of TCP/IP back then? Would you now have the funds left over to add any capability to the system? Or are your plans to develop the system as quickly as possible now and subsequently let it stagnate for years?

A commercial software vendor like CMX devotes a significant portion of its budget to ongoing research and development. Our very survival in a competitive market depends upon it. That's why CMX-RTX has so many unique features now (like our TCP/IP module for 16-bit processors) and so many exciting projects underway (e.g., our *Ethernet in a Box* hardware/software solution). And that is also why an in-house developed system will never be as rich in functionality, ease of use, flexibility, comprehensiveness, etc., and will not keep pace as the years progress - there are no competitive pressures finely honing the product every day.

Staying power.

A typical problem with in-house developed software is that all of the programming work falls to a primary individual who is a great resource to have around during the early stages of the product and knows the program inside and out. What happens when that person unexpectedly leaves or is transferred to the next exciting project? Often, it is the beginning of the end of the RTOS. New technical people brought on board have an NIH (Not Invented Here) attitude about the "old code" and are unable or unwilling to figure out its most intricate workings and problems. Getting any help with the system gets harder and harder and eventually the engineering group realizes that it is losing critical productivity and efficiency *because* of the software tool. At this point, "Too much time and money was invested in the old system to do anything for years," and your team is stuck with an unsupported, unchanging, and unwieldy tool.

CMX has been implementing and supporting RTOS's since 1990. It is the only thing we do and we plan to be around for a long time doing it. You never have to worry about not having someone to call with a question or problem - we will gladly be there to address it.

Documentation

Typically, both hardcopy documentation and on-line Help files for an in-house developed RTOS are shoddy at best (usually produced as an afterthought or completely neglected when the budget runs out at the end of the project). This means that the early users, who are theoretically trained at the outset, are the only ones who will be able to most effectively use the software and new users will have to rely on word-of-mouth or whatever

information they can glean from the system itself. This translates to fairly ineffective use of the system and yields less productivity and efficiency.

CMX offers complete documentation, including a comprehensive user's guide and online, context-sensitive help files.

Training

Training, at all levels, is critical to the successful use of any sophisticated software package. To provide users with a powerful tool and not teach them how to use it is a recipe for inefficiency and frustration. Who will be assigned to regularly train new and old users alike? Will the trainer actively prepare for each training session and have a course plan for each different section of the software? Or will training be relegated to the "we'll get to that when we have the chance" category?

CMX offers all of its clients the option of receiving training from its staff of experts when required.

Maintenance

What happens when your target changes in a year or two? Will you have the budget and time for a complete re-write of the RTOS and the application? Are you prepared to discover that the new microprocessor is completely different and will require significantly different functionality from the RTOS for best performance?

Even if you have budgeted and planned for such an occurrence, many institutions have later discovered - much to their chagrin - that their technical staff views such maintenance tasks as sheer drudgery and would much rather spend their time in more "challenging" pursuits. This means that these upgrades will be performed with as little effort as possible and always as a last priority (which is not a formula for a successful upgrade).

As a commercial software vendor, we are contractually obligated and fully prepared to provide meticulous service and support to all of our clients throughout their use of the system. We specifically hire staff members who take great pride in and, in fact, enjoy such work. To be successful, we must closely monitor and follow all technology trends so that our response can be quick when the latest platform or technology becomes available. That's why we are often a test site for the most popular hardware and compiler manufacturers and maintain close, working relationships with them.

Customization

"Customization" is often the mantra intoned by those who would argue for designing, programming, testing, implementing, and supporting your own RTOS. However, with CMX-RTX the ability to customize the system to your specific needs has been designed in from the beginning. Full source code is provided with every purchase for just this reason.

Also, all of the RTOS's functions are stored in a library so that only those functions that are called from a program are compiled into the code.

Engineer?

Your company would never consider having its accountants write their own spreadsheet program before beginning their work. It also would not consider having its secretaries create and develop their own word processing program prior to sending out letters. It just would not make economic sense. Why are engineers any different? One could even make the compelling argument that embedded systems engineers are a scarce resource these days and they should be focusing on those tasks that most leverage their skills and abilities.

Who should benefit from this decision?

Why do some people push for the obsolete idea of in-house software development when the rest of the world is buying commercially supported, off-the-shelf tools? An RTOS development project will certainly be a new challenge and a fun distraction from regular programming tasks, but is it the right business decision? From a business perspective, shouldn't your organization focus on what it does best: producing its core products on a timely and cost-effective basis? Isn't it in the best long-term, interest of all employees that the right business decisions be made for your company?

In Conclusion

Proponents of rolling your own RTOS will make statements that sound quite compelling, until they are answered with sound business facts:

Our application is too different/special/customized for a commercial tool to have the functionality we need. The same thing was said in the 1980's about word processing software. A commercial RTOS like CMX-RTX is right now being used by thousands of users producing hundreds of applications in industries ranging from appliances to medical equipment to cell phones to satellites to automobiles. While there may be some applications that our system has not been used for, the odds are that we have worked in your environment already. It is even more likely that we have all the functionality that you will ever need and it is tested, proven, and ready to go.

We don't like working with in a "black box" environment. An RTOS like CMX-RTX provides source code for all of its functionality. And it is well-documented – something that cannot always be said of in-house developed software.

A commercial RTOS is too expensive. CMX-RTX has a low initial cost and does not charge royalties for deployed products. It would be impossible to roll your own RTOS for less than the cost of this system.

Finally, if you wanted to fly coast-to-coast, you would not begin the trip by building the airplane. Nor would you feel that you must understand the complex intricacies of the wing rudder dynamics to arrive safely. Such efforts would be a waste of time and a complete distraction from your actual goals. Embedded systems engineers are too valuable a resource to waste their time and/or distract with non-core competency projects. Get your products completed on time and within budget by giving your team the tools it needs to get the job done now. Buy the RTOS!