

The DIY RTOS Meets Its Match

John Carbone, Express Logic

The do-it-yourself approach to a real-time operating system appeals to many an engineer's desire to control, schedule, and manage application interrupts. So ten years ago, the number of in-house RTOS applications outstripped all commercial RTOS implementations.

Although the do-it-yourself (DIY) RTOS remains the number-one competitor to commercial RTOSs, industry analyst Embedded Market Forecasters has found developers using a commercial RTOS have a greater likelihood of finishing their development projects on time. Developers commonly have three misconceptions about using a commercial RTOS:

1. Commercial RTOSs require too much memory. In fact, commercial RTOSs require from as few as two kbytes to as many as 500 kbytes. Few applications have such extreme size constraints that a commercial RTOS cannot meet them, particularly as memory costs have fallen and processors clock frequencies have increased. As Linux developers know, scaling down an open-source or DIY RTOS to this extent proves difficult if not impossible.

2. Commercial RTOSs cost too much. Analysts have examined the cost of commercial and DIY approaches to include acquisition costs, support, enhancements, and indemnification in case of legal challenge to the use of proprietary intellectual property in open-source code. When they considered all return-on-investment (ROI) factors, commercial RTOSs become the less-expensive option.

From 2003 to present, Embedded Market Forecasters (EMF) has consistently found that commercial RTOSs dominate ROI results with open-source RTOSs trailing and DIY RTOSs lagging further behind. A recent EMF white paper revealed that commercial RTOSs let developers better meet schedules and get products to market on-time, with 60 percent of projects completed on time or ahead of schedule versus 54 percent among those who took the DIY-RTOS route. (See For further reading.)

Not all commercial RTOSs fared the same. EMF asked hundreds of embedded developers, "Was your last project completed on-time, on-schedule, or behind-schedule?" and, "What operating system did you use?" The results, shown in Figure 1, indicate those who used commercial operating systems completed their projects on-time or ahead of schedule more often than the industry average.

The DIY RTOS Meets Its Match

Published on Electronic Component News (<http://www.ecnmag.com>)

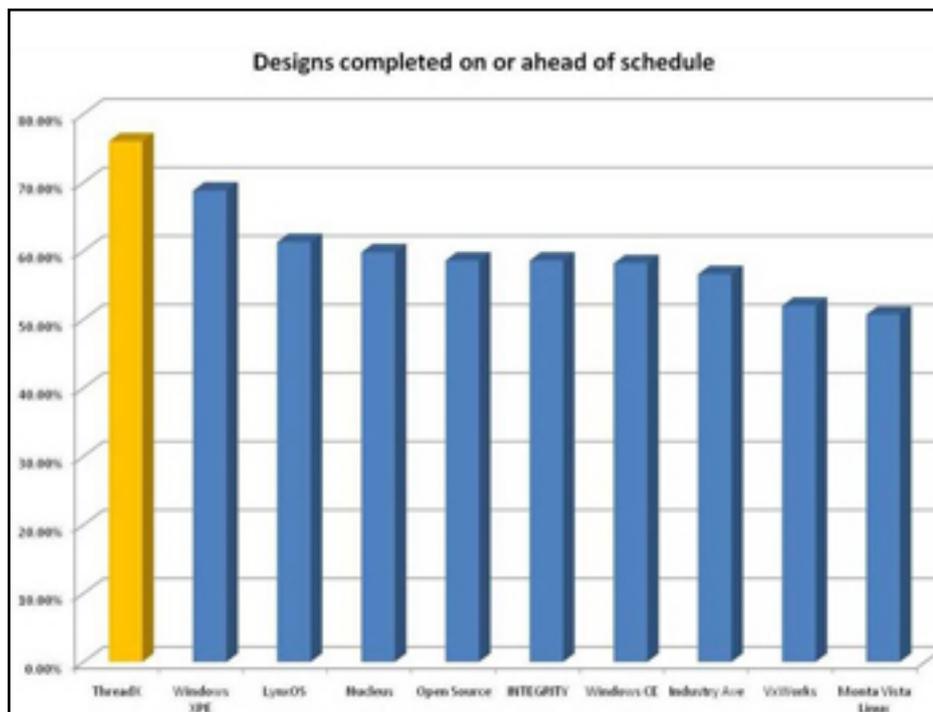


Figure 1. According to an Embedded Market Forecasters survey, respondents reported how well they met schedule, when using a particular RTOS.

In all instances, the trap for DIY and open-source RTOSs lies in the cost for support, product enhancements and ports to new microprocessors. These areas require substantial investments of time from an in-house development team. Commercial RTOS suppliers also invest heavily in these areas, but they amortize costs among many customers.

3. Commercial RTOSs are unsuitable for many applications. Obviously, this no longer holds true or we would see DIY RTOSs used in more successful products. According to EMF, in 1995 only 33 percent of applications included a commercial RTOS. Today, that number has grown to more than 76 percent.

Changes in the market for processor-based products also have driven developers to seek commercial RTOSs for several reasons:

1. Connectivity: Ten years ago, when DIY RTOSs dominated developers' efforts, 90 percent of applications simply needed a stand-alone scheduler to maintain real-time control of a local system. Now, though, 90 percent of applications communicate with something else. So RTOS vendors typically provide TCP/IP and USB stacks as well as secure-wireless options. Few companies can bear the cost to create and maintain this type of software on their own.

2. Integrated middleware: Consumers demand electronic products that include connectivity and graphical user interfaces (GUIs). Commercial RTOS products generally include a package of middleware that immediately works with the RTOS. When using a DIY RTOS, developers must select third-party middleware, or develop it themselves, and then integrate it with their RTOS.

The DIY RTOS Meets Its Match

Published on Electronic Component News (<http://www.ecnmag.com>)

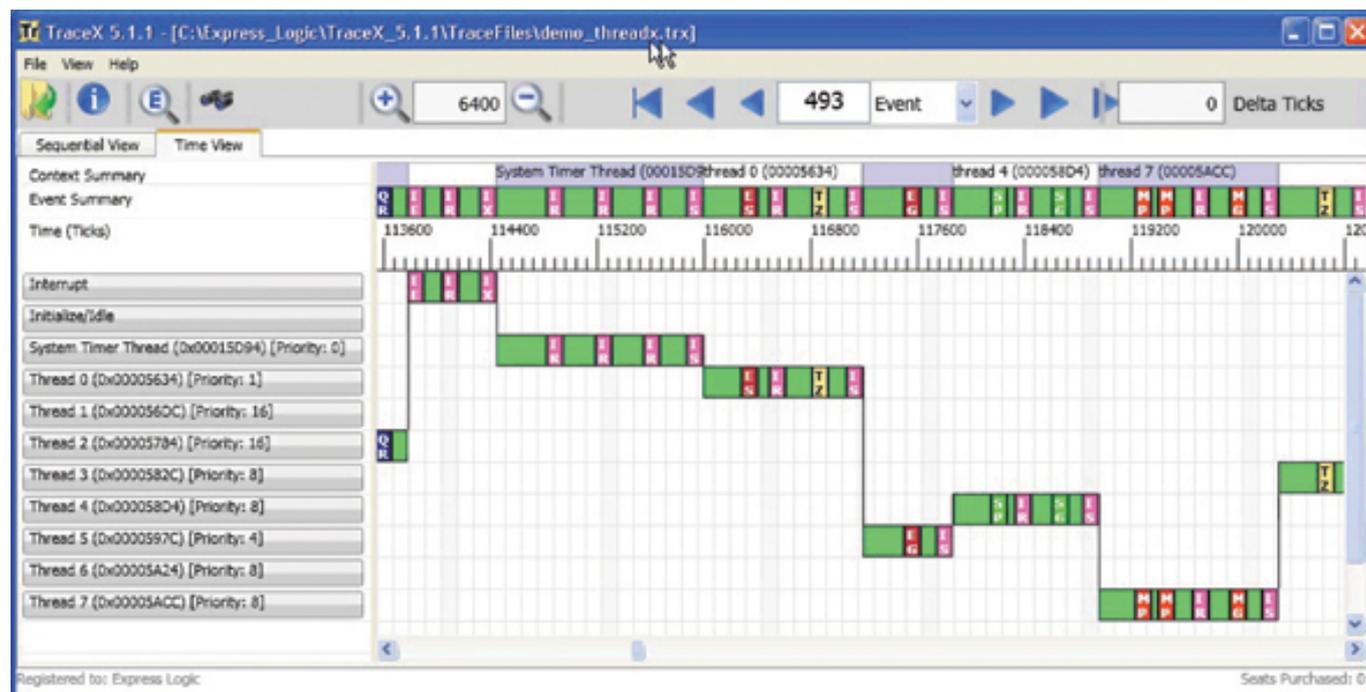


Figure 2. Express Logic's TraceX can identify priority inversion, which, if undetected, can lead to unresponsive operation, or even total system failure.

3. Tight development times: New products get introduced rapidly and companies must ensure reliable operation to avoid costly recalls. Shortened product-development times have increased the need for RTOS-related tools for debugging, profiling, and testing software. Typical debuggers, for example, fail to identify bottlenecks that then drive up development costs in real-time applications. So, commercial-RTOS companies provide compatible tools for application debugging and optimization. System-event analyzers, such as the Express Logic TraceX tool (Figure 2) plot application-thread activity versus time to reveal otherwise-obscure events that might indicate application-code bugs or inefficient operations. Another tool, the Express Logic StackX, determines the maximum stack size needed by each component of a program, so developers avoid the “guess-and-hope” process of assigning memory.

The sophistication of electronic products requires an RTOS in code of even modest complexity. And given the number and variety of RTOSs available commercially, developers have almost no reasons to create their own.

For further reading

“Shoot-Out at the RTOS Corral,” Embedded Market Forecasters.
www.rtos.com/PDFs/ShootoutRTOSCorral-03-2009.pdf [1].

About the author

John A. Carbone is vice president, marketing for Express Logic, Inc. He has 35 years of experience in real-time computer systems and software, ranging from embedded system developer to FAE to vice president of sales and marketing. Prior to joining Express Logic, John served as vice president of marketing for Green Hills Software.

The DIY RTOS Meets Its Match

Published on Electronic Component News (<http://www.ecnmag.com>)

Mr. Carbone has a BS degree in mathematics from Boston College.

Source URL (retrieved on 04/27/2015 - 6:47am):

<http://www.ecnmag.com/articles/2010/08/diy-rtos-meets-its-match>

Links:

[1] <http://www.rtos.com/PDFs/ShootoutRTOSCorral-03-2009.pdf>