The Affordable Care Act’s Tax on Innovation – Issues Confronting Medical Device and other Manufacturers

Why Time-to-Market and Total Cost of Development will become issues of Paramount Importance

Jerry Krasner, PhD., Chief Analyst

January 2013
The Affordable Care Act’s Tax on Innovation – Issues Confronting Medical Device and Other Manufacturers

Why Time-to-Market and Total Cost of Development will become Issues of Paramount Importance

Jerry Krasner, Ph.D., MBA

January 2013
EMF is the premier market intelligence and advisory firm in the embedded technology industry. Embedded technology refers to the ubiquitous class of products which use some type of processor as a controller. These products include guided missiles, radars, and avionics as well as robots, automobiles, telecom gear, and medical electronics.

EMF has been conducting research into the embedded market for more than a decade. EMF survey work is recognized as the most comprehensive and statistically accurate set of measures in the embedded market space by LSA, IBM, Microsoft and a number of other firms. Using research discipline from medical inquiry, EMF has developed a series of survey questions for developers which provide insight into the following areas:

- Trends in Integrated Development Environments (IDEs) and Real Time Operating Systems (RTOS)
- Trends in host processors (both standard microprocessors and Digital Signal Processors (DSPs)
- Trends in Interfaces and Trends in Bus and Board Standards
- Trends in Systems Engineering and Systems Architecture
- Trends in Software Languages
- Trends in simulation
- Trends in testing
- Trends in product life cycle management
- Trends in product development performance, practices and management

Embedded Market Forecasters (EMF) is the market research division of American Technology International, Inc. EMF clients range from startups to Global 100 companies worldwide. Founded by Dr. Jerry Krasner, a recognized authority on electronics markets, product development and channel distribution; EMF is headquartered in Ashland, Massachusetts.

About the author:

Jerry Krasner, Ph.D., MBA is Vice President of Embedded Market Forecasters and its parent company, American Technology International. A recognized authority with over 30 years of embedded industry experience, Dr. Krasner was formerly Chairman of Biomedical Engineering at Boston University, and Chairman of Electrical and Computer Engineering at Wentworth Institute of Technology and Bunker Hill Community College. In addition to his academic appointments, Dr. Krasner served as President of Biocybernetics, Inc. and CLINCO, Inc., Executive Vice President of Plasmedics, Inc. and Clinical Development Corporation, and Director of Medical Sciences for the Carnegie-Mellon Institute of Research. Earlier, he was Senior Engineer at the MIT Instrumentation Laboratory. Dr. Krasner earned BSEE and MSEE degrees from Washington University, a Ph.D. in Medical Physiology / Biophysics from Boston University and an MBA from Nichols College. He is a visiting professor at the Universidad de Las Palmas (Spain) where he was recognized for his work in neurosciences and computer technology.
Regarding the Data in this report

The data that is referred to in this report is statistically accurate and authentic and is based on:

- A statistically generated comprehensive and detailed survey of embedded developers and managers who reported on their design results (number of developers per project, vertical market of their design, time to market, percent of designs completed behind schedule or cancelled, closeness of final design outcomes to pre-design expectations, testing outcomes, etc.), the tools they used (development, modeling, Java, Eclipse, and other development tools), their choice of OS, IDE, communication middleware, processors used as well as where they go to learn about new products, tools and concepts.

- An EMF Dashboard – a unique tool that allows the user to simultaneously compare similar products (vendors can do competitive comparative analysis); that marketing executives can use for sales promo and strategic planning; that allows developers beginning a project to compare the experiences of hundreds of fellow developers that undertook similar projects to gain insights before making a commitment; and that allows CFOs and senior managers to look at what tools and processes resulted in the greatest cost savings.

For the interested reader, the following link demonstrates the power of the Dashboard and how we used it in developing the data that is presented herein:

The Affordable Care Act’s Tax on Innovation – Issues Confronting Medical Device Manufacturers

What’s at Issue?

Let’s state from the onset that manufacturers (and developers) of medical devices in the USA while confronting this new tax (which given the “fiscal cliff” crisis is unlikely to be repealed) will be forced to cut costs to remain solvent. Aside from layoffs and an abandonment of some ongoing developments, choices of tools and operating systems that enhance time to market design outcomes and require fewer developers per project (and incur less cost of development) will become necessary strategic considerations. Politics being what it is, we don’t expect medical devices to be the only technology marketplace that will be affected – and we caution manufacturers and developers alike that it is good business practice to assume the worst and to strategically approach savings as medical device manufacturers must do.

Breaking News

Before releasing this paper, EMF was expecting that there might be reconsideration by the Obama administration and the US Congress regarding what many see as a punitive tax on one of the few industries that the USA dominates. A recent statement by the president shows that, notwithstanding the opposition to this tax by 18 Democratic senators and senators-elect the president remains unmoved. The president stated:

"The health care bill is going to provide those health care companies, 30 million new customers. It’s going to be great for business and they’re doing really well right now and they’re going to get 30 million more customers as a consequence, so this additional tax essentially comes back to them as new customers.

The idea is that when you have 30 million more people coming in, you’re going to make money; you can do a little more to help facilitate and make sure people are getting the health care they need.

Unfortunately, a million more people clearly do not and can not translate into every segment of healthcare. When one says 30 million more "customers" what you really mean is 30 million more people who are simply insured now. Medical device companies will not see “millions of new customers.” EMF quotes the president for the purpose of making clear that embedded developers/managers should take the time to refocus on time and cost saving initiatives."
Background

Beginning in January 2013, medical device manufacturers in the U.S. must pay a 2.3% tax on gross sales of products, or roughly the equivalent of 15% of net sales. The tax is intended to bring in at least $28.5 billion over ten years to fund the Affordable Care Act. Remember that this tax will be paid before even one employee is paid and that there has never before been a company in America that had to pay a tax on their gross sales?

To be clear, the medical device excise tax was added as new code Section 4191 under chapter 32, which governs manufacturers’ excise taxes. The existing rules for these excise taxes apply to the medical device excise tax. These excise taxes are imposed on the sales of a manufacturer or importer, but are not be imposed on sales to a purchaser for further manufacture (or for resale to a second purchaser for further manufacture) or for export (or resale to a second purchaser for export).

Trade lobby AdvaMed released a study prepared by Ernst & Young LLP that argued the tax would raise companies' federal tax liability by an amount equal to 29 percent of their current federal income tax payments. The results, according to the manufacturers, will be job losses and a drying up of R&D funding. Already U.S. based companies are competing in a global arena - our competition, for example, includes companies based in Switzerland (3 percent tax) and Ireland (12.5 percent tax rate) that manufacture in Costa Rica and sell to U.S. healthcare organizations. Those firms already have an immense price advantage.

Device companies negotiate three to five-year deals with hospital systems, Integrated Delivery Networks and Group Purchasing Organizations. There is no way out of those deals and no way to simply raise prices. With 40 percent of hospitals losing money last year, and hospitals facing their own ACA problems, there's no way hospitals are going to allow price hikes. Unless this tax is repealed, companies are going to have to eat this tax, reduce headcount, shift work to low-tax nations or cut research and development spending – in addition to rethinking their development strategies.

What this means to the Embedded Technology Marketplace

It’s not clear why the medical device industry was singled out and other embedded markets have not been subjected to the same tax (yet). But the sage CEO/CFO should take all steps now to reduce development costs, focus on meeting time to market and target windows of opportunity, and how to maximize developments using fewer employees.

Central to this discussion is the choice of an operating system that is most appropriate to the application at hand; one that satisfies application and power requirements and one that has a track record (at least over the past 3 years) of superior time to market, the lowest cost of development and using fewer developers per project. The analysis should also show how close the final design approximates the pre-design expectations for Performance and for Systems Functionality.
Investor groups as well as Angels will want to see company due diligence to be comfortable that their investments have a good chance of succeeding.

To put matters into perspective, we also provide a discussion of factors that contribute to why certain OSes outperform others.

**How important to Cost Control is the Choice of an Operating System?**

567 embedded developers were surveyed. They responded to over 90 questions that inquired as to their activities including the number of developers on a project (SW and HW), the vertical market to which the design was intended, how long it took from design start to shipment, the percent of design cancelled or completed behind schedule and how long the project ran before cancellation or final shipment, chips used, OSes used and tools used.

We also inquired as to how close the final design came to its pre-design expectation (within 20% is considered a good design outcome). Using the EMF Dashboard (a unique tool that permits the simultaneous display of multiple filters) users of Integrity, LynxOS, VxWorks and ThreadX were selectively filtered from the database and simultaneously displayed (see www.embeddedforecast.com for a demo of the Dashboard) so that comparisons could be made.

Table I presents a comparison of the activities of the users of Integrity, VxWorks, LynxOS and ThreadX. Integrity, VxWorks and LynxOS are considered to be the major operating systems and have been wrongly suggested as the ideal mission critical operating system for patient monitoring.

Given that the maximum frequency response for patient monitoring is 100 Hz, one can see that this idea is preposterous. Nonetheless, it is interesting to see in Table I how an alternative OS (one with over one billion apps shipped) compares with the traditional OSes.
<table>
<thead>
<tr>
<th></th>
<th>Ind ave</th>
<th>VxWorks</th>
<th>LynxOS</th>
<th>Integrity</th>
<th>ThreadX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Devel time Months - Start to Ship</td>
<td>13.9</td>
<td>16.0</td>
<td>13.3</td>
<td>17.9</td>
<td>12.9</td>
</tr>
<tr>
<td>% behind schedule</td>
<td>47.0%</td>
<td>50.0%</td>
<td>35.5%</td>
<td>43.5%</td>
<td>36.9%</td>
</tr>
<tr>
<td>Months behind</td>
<td>3.8</td>
<td>3.1</td>
<td>2.3</td>
<td>3.4</td>
<td>3.0</td>
</tr>
<tr>
<td>% cancelled</td>
<td>11.2%</td>
<td>12.3%</td>
<td>9.3%</td>
<td>17.5%</td>
<td>13.4%</td>
</tr>
<tr>
<td>Months before cancellation</td>
<td>4.4</td>
<td>4.9</td>
<td>3.6</td>
<td>5.4</td>
<td>4.6</td>
</tr>
<tr>
<td>SW Developers/project</td>
<td>14.7</td>
<td>8.1</td>
<td>7.7</td>
<td>7.3</td>
<td>6.8</td>
</tr>
<tr>
<td>Average Developer months/project</td>
<td>204.3</td>
<td>129.6</td>
<td>102.4</td>
<td>130.7</td>
<td>87.7</td>
</tr>
<tr>
<td>Developer months lost to schedule</td>
<td>26.3</td>
<td>12.6</td>
<td>6.3</td>
<td>10.8</td>
<td>7.5</td>
</tr>
<tr>
<td>Developer months lost to cancellation</td>
<td>7.2</td>
<td>4.9</td>
<td>2.6</td>
<td>6.9</td>
<td>4.2</td>
</tr>
<tr>
<td><strong>Total developer months/project</strong></td>
<td><strong>237.8</strong></td>
<td><strong>147.0</strong></td>
<td><strong>111.3</strong></td>
<td><strong>148.4</strong></td>
<td><strong>99.4</strong></td>
</tr>
<tr>
<td><strong>At $10,000/developer month</strong></td>
<td><strong>$2,043,300</strong></td>
<td><strong>$1,296,000</strong></td>
<td><strong>$1,024,100</strong></td>
<td><strong>$1,306,700</strong></td>
<td><strong>$877,200</strong></td>
</tr>
<tr>
<td>Average developer cost/project</td>
<td><strong>$262,542</strong></td>
<td><strong>$125,550</strong></td>
<td><strong>$62,871</strong></td>
<td><strong>$107,967</strong></td>
<td><strong>$75,276</strong></td>
</tr>
<tr>
<td>Average cost to delay</td>
<td><strong>$72,442</strong></td>
<td><strong>$48,819</strong></td>
<td><strong>$25,780</strong></td>
<td><strong>$68,985</strong></td>
<td><strong>$41,915</strong></td>
</tr>
<tr>
<td>Average cost to cancellation</td>
<td><strong>Total developer cost/project</strong></td>
<td><strong>$2,378,284</strong></td>
<td><strong>$1,470,369</strong></td>
<td><strong>$1,112,750</strong></td>
<td><strong>$1,483,652</strong></td>
</tr>
<tr>
<td>Lines of Code/Project * 1000</td>
<td>785.7</td>
<td>393.1</td>
<td>268.3</td>
<td>690.1</td>
<td>536.2</td>
</tr>
</tbody>
</table>

**Table I: Comparative Cost of Development between Operating Systems**

Note that all OSes are more cost effective than the average embedded design. Nonetheless, ThreadX users get to market faster, have fewer designs completed behind schedule, and have the lowest average cost of development.

What is of interest to EMF is that these have been the finding for the past five years (year over year). Now it is well known that the vendors of the major OSes have their own IDEs Even though they each are OS agnostic – that they can be used to develop with any OS – users of VxWorks tend to use Wind River’s Workbench; users of Integrity use Green Hills’ MULTI and LynxOS users tend to use Luminosity.

Of interest is that until recently, Express Logic didn’t have their own IDE for ThreadX developments. When we look at user data we see that ThreadX developers use a wide range of IDEs. Since ThreadX developers have the cost and times to market advantage yet don’t use the same IDE, the OS itself must be responsible.

We hear a lot about using open source software for medical applications (no matter what they claim it’s not free). Table II presents data from developers that indicated that they use open source software (without being specific as to the OS); that they use commercial Linux (e.g., MontaVista, Blue Cat or Wind River Linux); or that they use non-commercial Linux (free download).
Table II: Comparative Cost of Development between Open Source Operating Systems

For those that insist that using open source software is the gateway to inexpensive product development should take Table II to heart. There are reasons of course why unsupported software would be more expensive to use. But this is a topic for another discussion. As we have observed similar outcomes, year-over-year for the past five years we are confident in the data.

Developers were asked how close their final design results approximated their pre-design expectations. They were given response choices of:

- Within 10%
- Within 20%
- Within 30%
- Within 40%
- Within 50%
- Not within 50%

EMF considers final design outcomes within 30% of pre-design expectations to be a good design outcome. Designs within 20% are considered to be excellent outcomes.
Table III presents the comparison between final design outcomes that were within 30% of pre-design expectations for commercial RTOSes. Table IV presents the data for open source software.

<table>
<thead>
<tr>
<th></th>
<th>Ind Ave</th>
<th>ThreadX</th>
<th>Integrity</th>
<th>VxWorks</th>
<th>LynxOS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Within 30% of Expectation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expectations in Performance</td>
<td>69.1%</td>
<td>88.9%</td>
<td>78.8%</td>
<td>70.6%</td>
<td>73.7%</td>
</tr>
<tr>
<td>Expectations in Systems Functionality</td>
<td>67.5%</td>
<td>83.4%</td>
<td>68.8%</td>
<td>68.6%</td>
<td>73.7%</td>
</tr>
</tbody>
</table>

**Table III: Comparative Design Outcomes for Commercial RTOSes**

<table>
<thead>
<tr>
<th></th>
<th>Ind</th>
<th>Open</th>
<th>Comm</th>
<th>Linux non-comm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Within 30% of Expectation</strong></td>
<td>Ave</td>
<td>Source</td>
<td>Linux</td>
<td></td>
</tr>
<tr>
<td>Expectations in Performance</td>
<td>69.1%</td>
<td>70.0%</td>
<td>75.2%</td>
<td>69.6%</td>
</tr>
<tr>
<td>Expectations in Systems Functionality</td>
<td>67.5%</td>
<td>75.3%</td>
<td>78.0%</td>
<td>69.7%</td>
</tr>
</tbody>
</table>

**Table IV: Comparative Design Outcomes for Open Source Software**

Once again the comparison between commercial RTOSes and the use of open source software is dramatic. The ThreadX results again show that there are alternative choices to the major OSes that all developers should consider in making their project choice.

**Comments on Presented Data and what Embedded Developers Can Take Away**

Our year-over-year data consistently points at Express Logic's ThreadX RTOS as offering the best development results in each of the past 6 years. It's a compelling record and we wonder why ThreadX has done so well among all RTOSes used by the developers we surveyed. We approached Bill Lamie, Express Logic's founder and author of the ThreadX RTOS (as well as previously being the author of Nucleus PLUS, now offered by Mentor Graphics), and asked for his assessment of the reasons why ThreadX outperforms the competition so consistently. Bill points to "Ease of Use" as the driving factor, and enumerates several areas that contribute to ease of use in any RTOS, and most effectively, in ThreadX:
1. **User Guide**
   I think the most important element is the User Guide. The ThreadX User Guide is highly organized and designed to be simple and clear. Even the API was somewhat designed for the User Guide - one of the primary reasons the “noun verb” API naming convention was used was to keep the API description in both alphabetic and function order. There is also a ThreadX text book to augment the documentation.

2. **API**
   a. The use of easy to understand naming conventions that includes “noun verb” ordering, and very limited use of abbreviations. For example, in ThreadX, all queue APIs have tx_queue_* in the name. The naming conventions also make ThreadX API names distinct from application code which also helps reduce confusion.
   b. The API should be highly consistent. For example, all services that suspend have an optional time-out parameter that behaves the same way regardless of the API being used. The timeout parameter is always the last parameter in the API. All APIs take the object pointer as the first parameter. Most APIs return a completion status in the form of a UINT. All memory required by the APIs is passed into the API by the application. The idea here is that the user can leverage experience using one ThreadX API when he/she uses another ThreadX API.
   c. All the important functional needs of an RTOS for the 95% use case are met in ThreadX. An example of functional completeness is that all APIs that might need to suspend can optionally suspend at the user's direction. In ThreadX, applications can suspend on queue full conditions as well as memory allocations when there isn't enough memory. Many RTOSes don't have this functionality or are missing whole components like mutexes or memory allocation. Another functional advantage is that many ThreadX APIs can be called from a variety of contexts, e.g., there aren’t separate API's that ISRs must use that are different from those which application threads can use. Another nice feature is having error checking on a per-compilation unit basis rather than the traditional all or nothing approach that other RTOSes have. This allows already debugged code to run without extra error checking while newer code can still use error checking.
   d. The total number of APIs is manageable. Some RTOSes have 300+ API names, which makes it impossible for users to ever "master" their usage. In ThreadX there are 50 core APIs and 84 total APIs which makes it reasonable to "master" the API, and also to comprehend existing code, without constant reference to the User Guide.
3. **Size**
   a. The small footprint of ThreadX makes it easier for the developer to fit everything into the fastest memory in the system, rather than being forced to use external, slower memory.
   b. The user provides the memory to ThreadX, which makes it easier to dynamically configure ThreadX and not rely on worst-case allocations and the complexity that goes along with that.
   c. Best of all, the footprint scales automatically, based on services actually referenced in the application, making that simple for the user as well (no complicated RTOS configuration needed like in some other RTOSes).

4. **Performance**
   a. The performance in ThreadX is constant regardless of system load, number of resources (threads, queues, etc.). This makes validating (testing, debugging) the application much easier.

5. **Source Code**
   a. Having the source code allows users to build the ThreadX library with whatever set of optimizations/debug info they wish.

In addition, the following characteristics help to make any RTOS, and ThreadX in particular, easy to use:

1. Availability of Simple Demo Programs
2. Easy Migration to New Processors
3. Configured for CPU/Tools for “Out-of-the Box” use
4. Integrated Trace Capability
5. Designed As A Linkable Library
6. Single Execution Mode
7. Linear Address Space
8. Additional Complementary Middleware Products
9. Made From The Ground Up For Real-Time, Embedded Applications