

Shoot Out at the RTOS Corral

**How Design Outcome Data Analysis can be used to
assist Medical Device Developers in Selecting the OS
Most Appropriate to Their Product Design Requirements**

Jerry Krasner, Ph.D., MBA

March 2009

Embedded Market Forecasters

American Technology International, Inc.

Embedded Market Forecasters
Research and Consulting
for Embedded Products,
Markets and Channels



About EMF: www.embeddedforecast.com

508-881-1850

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.

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 Framingham, Mass.

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 has extensive clinical research and medical industrial experience, including the successful filing of more than a dozen 510k submissions.

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. He has been the principal investigator of several NIH funded clinical research programs.

Dr. Krasner was formerly Chairman of Biomedical Engineering at Boston University, and Chairman of Electrical and Computer Engineering at Wentworth Institute of Technology.

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.

Copyright 2009 by Embedded Market Forecasters, a division of American Technology International, Inc, 1257 Worcester Road #500, Framingham, MA 01701. All rights reserved. No part of this document covered by copyright hereon may be reproduced or copied without expressed permission. Every effort has been made to provide accurate data. To the best of the editor's knowledge, data is reliable and complete, but no warranty is made for this.

Shoot Out at the RTOS Corral

Jerry Krasner, Ph.D.

March 2009

Table of Contents

Executive Summary	4
As RTOS Vendors vie for the Growing Medical Device Market, How can Developers sort through the claims and counterclaims? ..	4
There Isn't a Bad Operating System in the Bunch – So which one best fits your needs?	6
RTOS Selection Challenges	7
Data Analysis of RTOS-based Design Outcomes	9
Making Sense of Software Certification Standards – and Deciding Which, if any to Employ	11
Communications Security – Insuring Patient Privacy	14
Summary	15

Executive Summary

Current economic conditions have resulted in a slowdown in mil/aero and avionics embedded markets. RTOS vendors looking to shore up their revenues are exploring other verticals that might place a premium on their operating system offerings. Vendors that have certified their OSES to the extreme mission critical demands imposed for avionics applications, for example, have launched campaigns that suggest that these stringent standards need to be required for medical device applications.

The difference between patient monitoring and data gathering applications and those concerned with keeping time critical multiple systems functioning required for mil/aero applications raises an important question. Are such certified OSES truly necessary for all medical applications?

It's like a shoot out between the various OS vendors each looking to outdo the other, but in the process adding more confusion than enlightenment. Add to this the likely legislation working its way through the US Senate that would impose criminal and financial penalties on companies that don't document and accurately report on their software development processes and testing.

What should medical device developers do in order to keep their CEOs out of jail?

In this report, EMF presents a clarification of the needs, and an independent assessment of the challenges facing embedded medical device developers. The OSES presented herein are all excellent in their own right, but may or may not be applicable to the specific application at hand.

As RTOS Vendors vie for the Growing Medical Device Market, how can Developers sort through the claims and counterclaims?

There is an ancient Chinese curse that says "may you live in interesting times". How much more interesting can the times be? With the US economy in recession, banks facing bankruptcy, a new liberally dominated presidency and Congress historically opposed to military growth, and companies hedging their bets for expansion and growth (assuming that they can find lines of credit), the embedded industry is facing an uncertain future. When it comes to securing traditional revenue sources, companies (particularly those in Aerospace and Defense) are looking to embark in new markets in order to compensate for anticipated losses.

So it comes as no surprise that many RTOS vendors are looking to the medical device marketplace and are looking for short cuts to garnering market share, while making claims that best serve their purposes. Surprisingly, some of the major RTOS vendors have positioned their products without fully understanding the nature of how Class I, II, and III device requirements are differentiated.

So what is the typical medical device developer to think? Are OSES, which are certified for operation in extreme and potentially catastrophic applications, the best OSES for patient monitoring? What does certification mean and what types of certification are appropriate or inappropriate for a medical device?

Should developers be using in-house or open source software rather than commercially available operating systems – and what about Linux? There are additional questions regarding security (and what JCAHO and HIPAA may have to say) and how these considerations play into the expected Drug and Medical Device Accountability Act which can hold companies and their executives criminally accountable for development choices and development documentation?

In order to clear the confusion and provide information of importance to medical device developers, EMF presents thoughtful guidance regarding these issues to help developers decide on the OS most appropriate to the medical device under development and to provide a detailed analysis regarding design outcomes that have been derived from years of comprehensive embedded developer surveys.

The CDRH's software forensic group reported that in 1996, 10% of medical device recalls were software-related issues; by June 2006 that number had grown to 21% of all recalls. The CDRH believes that no manufacturer wants to be in a position in which the FDA finds bugs in their software – so medical device developers need to choose wisely.

Let the shoot out begin.

There Isn't a Bad Operating System in the Bunch – so which one best fits your needs?

Operating systems can range from the very modest, to midrange, to open source and in-house OSES, to powerful RTOSes on steroids. Each offers particular advantages – given the application at hand – and each carries similar disadvantages (footprint, power consumption, cost, processors required, etc.).

To say that one is better than another requires the context of the application.

Let's look at the characteristics of patient monitoring, charting and display applications.

- 1) Heart rate monitoring is done via ECG or pulse (one can have an ECG and no cardiac contraction – called electromechanical dissociation - hence the expanded use of finger pressure transducers). The ECG has a bandpass of 0.5 Hz to 100 Hz (this derives from Einthoven's galvanometer which was used in the late 1890's). The database for ECG measurements involves billions of recordings upon which were predicated current interpretations. If one were to give a physician an ECG with a 3 KHz bandpass, they would certainly consign the unfortunate person to the cardiac intensive unit. The ECG is measured with an A/D interface and the display is filtered to present the expected waveform. Arrhythmias are of a lower frequency than the typical QRS complex
- 2) The pressure pulse used for heart rate is a slowly rising and declining waveform that lasts around 450 msec. As it represents an actual cardiac contraction and is a cheaper device to build and sell, it is used more frequently
- 3) Respiration can be measured by a chest strain gauge or an impedance plethysmograph. The normal range of breathing is 3 to 20 breaths per minute. An actual breath can take many seconds to complete
- 4) Muscle potentials have a base frequency of 3 KHz and are FM/FM modulated. In the past, muscle physiologists used the amplifiers that were most plentiful in the laboratory – unfortunately these were the ECG amplifiers and many books were filled with recordings (and many Ph.D. thesis were based on such recordings) that were skewed based on the amplitude and frequency consequences of using a band limited amplifier
- 5) EEG signals have a frequency range of 3 Hz to 100+ Hz. A/D sampling is usually done at 512 Hz

The question that should be posed to medical developers is “do you really need an OS on steroids to achieve the desired monitoring requirements”? In other words, since most RTOSes can meet the response requirements of such applications, why pay for unnecessary bells and whistles and incur the overhead of the required high power processor when the design parameters for patient monitoring require the responsiveness that small, efficient RTOSes can provide?

A smaller OS makes for an easier 510k conformance process, not to mention faster performance, smaller memory requirements, and lower power consumption.

RTOS Selection Challenges

Research from Embedded Market Forecasters (EMF) indicates that time to market should be one of severable key selection factors in picking a real time operating system (RTOS). Although in the past RTOS selection was determined by an assessment of the functional characteristics of the candidate systems, the increased complexity and the changes in the embedded market means that these factors are no longer sufficient to provide successful results. With the emphasis on seizing windows of market opportunity, factors which affect time to market, design outcomes including “designs completed ahead or behind schedule” and “proximity between pre-design expectations and final design results (for performance, systems functionality, and features and schedule)” have become much more important.

According to EMF survey results, some operating systems consistently outperform others in terms of helping developers get their projects to market on time or even ahead of schedule. This paper examines data that shows the impact of RTOS selection on whether the project was completed on or ahead of schedule. Since time to market is a significant factor in the profitability of most projects, it's clear that developers should look closely at these figures and make development speed a critical consideration in operating system selection.

Without question, the traditional criteria still matter: the RTOS must also be capable of meeting the functional requirements of the application. EMF observes that using an RTOS that is overqualified for the application may have a negative impact on time-to-market, design cost, and product support. This is because additional capabilities, beyond what is required by the needs of the application, make these RTOSes more complicated and harder to use.

Medical device developers are confronted by greater design complexities and limited windows of opportunity – in addition to the demands of the Class II and III pre-market application process. Medical device developers usually don't have the time to research operating system alternatives and make decisions that can have a major impact on development and deployment processes.

The following data is derived from the detailed 2008 annual EMF survey of embedded developers from which such factors as time-to-market, the percentages of designs completed ahead or behind schedule, and the closeness of final design outcomes to pre-design expectations, are developed for commercial and open source RTOSes. It is presented to provide medical device developers reference information that they otherwise could not have found.

Commercial operating systems form a continuum of functionality, performance, and price. These operating systems range from those that offer a basic preemptive scheduler and a few key system services, which are usually inexpensive and come with modifiable source code and are royalty free, to those more sophisticated operating systems that typically include a lot of functionality beyond the basic scheduler and can be quite expensive. With such a variety of operating systems and features to choose from, it can be difficult to decide which is best for a given project. Many developers make their decision based on performance, functionality, or compatibility with their choice of compiler, debugger, and other development tools. Many use integrated development

environments (IDEs) that enable them to develop over a wider range of RTOSes and to use Eclipse-based tools.

When faced with all the traditional selection criteria, choosing an RTOS appears to be a toss-up, with no RTOS standing head-and-shoulders above the others in technical merit. However certain OSES seem to produce better results, year over year. Without statistically accurate design outcome information for developers to consider, RTOS selection has been primarily a matter of preference and convenience. Development teams tend to resist change in the absence of perceived benefits of change. And, since many RTOSes now offer similar technical capabilities and a broad range of software and hardware ports, there are few technically compelling reasons to change.

EMF cautions the reader that the following data must be carefully interpreted. Any comparison between RTOSes must be made within the context of the application.

The following considerations should be a part of your critical thinking.

- If it fits your application, a smaller OS makes for faster performance, smaller memory requirements, and lower power consumption. Unused features are useless, add dead weight, and complicate proper selection of the features that are needed
- To certify or not – if you feel that certification of your application software is an advantage, there are formal methods available (e.g., SCADE) that can produce the same certification level as DO-178B Level A. Some RTOSes (e.g., from Green Hills, Wind River and LynuxWorks) have received DO-178 B Level A certification for their OS. Other vendors (e.g., Express Logic) have had their OS certified to DO-178 B Level A by customers
- Certain OSES have been targeted to mission critical applications in which software failure would have catastrophic consequences. OSES provided by Green Hills, Wind River and LynuxWorks are examples for which DO-178B Level A and ARINC 653 certifications have been achieved and are required for avionics on-board software
- Other OSES provided by Express Logic (ThreadX) and Monta Vista (Linux) aren't marketed to those markets and hence have not undergone the time and expense that such certifications require. These OSES, characterized by small footprint, small memory requirements and power consumption, have been deployed in hundreds of millions of applications without problems. For many applications they may be a better fit for medical devices
- Should I use an Open Source OS rather than a commercial OS? There is no free lunch and while open source can provide certain benefits, there is a down side as well. Things to consider – design outcomes; designs completed ahead or behind schedule; time to market considerations; how close is your final design to your pre-design expectations?

In the next section we will explore design outcomes from the survey responses from embedded developers. EMF conducts detailed and comprehensive surveys of embedded developers. Data from these surveys can be cross tabbed to determine time to market, percent of designs completed ahead of or behind schedule, and how close one's final design approximated pre-design expectations.

Data Analysis of RTOS-based Design Outcomes

Embedded development engineers were interviewed via a comprehensive survey designed to elicit information regarding current and anticipated tool usage, design starts, completions and cancellations, development (host) and target platforms, microprocessors used, desirable and undesirable product features, vendor evaluation criteria and purchasing decision processes, among other important information. Responses from 455 embedded developers comprised the EMF 2008 comprehensive survey, which explored the attitudes, preferences and values of embedded developers to the current and projected use of embedded technologies. The survey was constructed such that the responses could be evaluated from many perspectives, including total response, specific job title of the respondent, architecture employed in embedded design, processor family, and each of ten embedded vertical market application (e.g., telecom, industrial controls, etc.).

Table 1 presents design outcomes derived from the 2008 EMF survey of embedded developers for smaller OSEs including open source (which in some cases might be larger). These outcomes are for reference as there are other factors that reflect different reasons for selecting an appropriate OS for a medical device.

	Industry Ave	Medical Ave.	ThreadX	Nucleus	MS-CE	Open Source
Months to Market - design start to ship	13.6	15.0	11.1	15.1	14.2	13.8
Designs completed ahead of schedule %	21.8%	28.7%	27.8%	17.3%	24.2%	24.1%
Designs completed behind schedule %	43.4%	47.1%	24.0%	40.2%	41.8%	41.4%

Table I: Sample Design Outcomes

Table II presents design outcomes from the major OS vendors. Note that the major vendors provide a number of OSEs designed to different operating environments, but we have selected their principal operating system.

	Monta Vista	Integrity	LynxOS	XPE	VxWorks
Months to Market - design start to ship	15.3	15.3	13.7	12.8	16.6
Designs completed ahead of schedule %	20.3%	16.8%	24.6%	29.3%	16.3%
Designs completed behind schedule %	49.3%	41.4%	38.7%	31.2%	48.0%

Table II: Additional Sample Design Outcomes

Table III presents a comparison of the percent of final design outcomes that are within 30% of pre-design expectations.

	Industry Ave	Medical Ave.	ThreadX	Nucleus	MS-CE	Open Source
Performance	74.5%	78.7%	94.2%	69.3%	72.3%	77.9%
Systems Functionality	74.2%	75.8%	88.2%	53.9%	73.9%	78.9%
Features & Schedule	69.6%	63.6%	94.0%	61.6%	71.8%	77.0%

Table III: Comparison of Pre-Design Expectations to Final Design Results

Table IV continues these comparisons.

	Monta Vista	Integrity	LynxOS	XPE	VxWorks	WRS Linux
Performance	84.4%	85.0%	76.0%	74.2%	78.8%	79.4%
Systems Functionality	88.3%	75.0%	68.0%	76.4%	81.8%	79.3%
Features & Schedule	81.9%	75.0%	70.9%	74.1%	77.3%	73.9%

Table IV: Additional Comparisons of Pre-Design Expectations to Final Design Results

What is interesting in looking at this data is that it is not dependent upon the complexity of the design or the particular application. Developers reported factors that are under their control and how their designs wound up.

One can argue that “time-to-market”, measured from design start to shipment, can be subject to the complexity of the development and the magnitude of the project. Whereas this is a fair observation, it is nonetheless of value to look at average “time-to-market” information among OSEs that usually involve the same types of developments. In many cases, mid-design “surprises” can affect time-to-market results, as well as designs completed on or ahead of schedule. It is certainly fair to use comparative data for designs completed ahead of or behind schedule, as well as comparisons of final design results with pre-design expectation as these are not dependent on design complexities and applications.

Let’s now look at other considerations that can influence the choice of an appropriate OS to meet the design requirements of a medical device.

Making Sense of Software Certification Standards – and Deciding which, if any to Employ

Medical device developers are being bombarded with claims from embedded RTOS vendors regarding their OSES and why they believe their “certified” OSES should be used.

EMF has addressed some of the issues regarding whether a certified OS is desirable (or required) and whether application software should also be developed by a certification process. The following is a description of certification standards that are usually referred to in OS promotional literature.

For the aerospace industry, the DO-178B specification describes production of software for airborne systems and equipment. The objective of the guidelines is to ensure that software performs its intended function with a level of confidence in safety that complies with airworthiness requirements.

The DO-178B standard defines five “Development Assurance Levels” ranging from no effect on the safe operation of the airplane (Level E) to catastrophic failure for the airplane (Level A). The higher the level the stricter the requirements for the design process and code. Green Hills, Wind River and LynuxWorks have certified certain of their operating systems to meet DO-178B Level A, while certain of Express Logic’s customers have certified ThreadX to the same DO-178 B level.

ARINC 653 (Avionics Application Standard Software Interface) is a software specification for space and time partitioning. It defines an API for software of avionics, following the architecture of Integrated Modular Avionics. It is part of ARINC 600-Series Standards for Digital Aircraft & Flight Simulators. ARINC-653-1 Application/EXecutive (APEX) interface provides a recognized standard interface between the operating system of an avionics computer resource (ACR) and the application software. Many companies fully support ARINC-653-1 while complying with DO-178B Level A, thereby providing a COTS baseline avionics operating environment that meets standards already adopted and accepted by the commercial avionics industry for Integrated Modular Avionics.

The Evaluation Assurance Level (EAL1 through EAL7) of an IT product or system is a numerical grade assigned following the completion of a Common Criteria security evaluation, an international standard in effect since 1999. Increasing assurance levels reflect added assurance requirements that must be met to achieve Common Criteria certification. The intent of the higher levels is to provide a greater confidence that the system's principal security features are reliably implemented. The EAL level does not measure the security of the system itself, it simply states at what level the system was tested to see if it meets all the requirements of its Protection Profile.

To achieve a particular EAL, the computer system must meet specific *assurance requirements*. Most of these requirements involve design documentation, design analysis, functional testing, or penetration testing. The higher EALs involve more detailed documentation, analysis, and testing than the lower ones. Achieving a higher EAL certification generally costs more money and takes more time than achieving a lower one. The EAL number assigned to a certified system indicates that the system completed all requirements for that level.

Although every product and system must fulfill the same *assurance* requirements to achieve a particular level, they do not have to fulfill the same *functional* requirements. The functional features for each certified product are established in the *Security Target* document tailored for that product's evaluation. Therefore, a product with a higher EAL is not necessarily "more secure" in a particular application than one with a lower EAL, since they may have very different lists of functional features in their Security Targets. The NSA stipulates that OSEs at EAL levels 4 or lower can never be upgraded to EAL levels 5 through 7.

A product's fitness for a particular security application depends on how well the features listed in the product's Security Target fulfill the application's security requirements. If the Security Targets for two products both contain the necessary security features, then the higher EAL *should* indicate the more trustworthy product for that application. EAL security is concerned with maintaining the internal security of the application to determine the degree to which the system is isolated from external attacks. Currently, Green Hills is the only vendor to achieve an EAL 6+ certification.

For developers that want to have their application software certified to the DO-178 B standard irrespective of the OS employed, Esterel's Safety Critical Application Development Environment (SCADE) is capable of generating certifiable code to the highest level, namely the A level. For this they supply a code generator capable of generating C and Ada with several options for optimization and configuration.

SCADE certification relies on both an external view of the software (does it obey its requirements) and also on an internal examination of the precise way it was developed (this is what Esterel calls "design assurance level"). If an OS has source code available, the OS design documents, the verification reports etc., then the OS can be similarly certified for a specific development.

Since SCADE is based on a formal language and the SCADE products have been qualified as development tools under DO-178B to Level A, including a qualified code generator (KCG), SCADE developed software can be ported to target hardware without having to perform code review.

What is the medical device developer to take from this information?

EMF recommends that developers follow the likelihood that the Drug and Medical Device Accountability Act will make it through the Senate this year, and make sure that they meet the requirements that should be followed to keep your CEO out of jail and keep the company in business.

Your OS selection should be compatible with and provide audit trails and documentation for requirements management and software control management tools (that include version control documentation, validation and verification tools, and testing/software verification tools). Importantly, your OS selection should be compatible with system modeling tools that permit an ability to visualize code execution (and the visual mapping of requirements onto code within the model) and the ability to automatically upgrade legacy code to new hardware requirements. IBM Rational's Rhapsody is a serious suggestion. MathWorks' Simulink and SCADE have links to Rhapsody and other

mentioned necessary tools that enabled their modeling tools to take advantage of broader capabilities.

If your OS is not compatible with the tools needed to comply with the anticipated Drug and Medical Device Accountability Act, you may be taking on more risk and added design time than it is worth. Be sure to ask your vendor.

Be sure to involve your OS vendor, up front, to insure that their OS is compatible with the following. If your vendor won't respond, get another one.

- 1) Risk assessment and management is a very important part of your filing with the CDRH and can be your best approach for protecting you under the Act. Be exceeding careful in documenting the "Level of Concern" section of your application
- 2) Submit a Device Hazard Analysis for all software devices – include all hazards (hardware and software) associated with the products intended use
- 3) Submit a Software Requirements Specification (SRS) that includes functional, performance, interface, and developmental requirements for the software, including hardware, OS and programming language requirements
- 4) Include an Architecture Design Chart (flowchart or similar illustration) that describes the relationships among the major functional units in the software device. There should be sufficient information to allow for the organization of the software relative to the functionality and intended use of the software device.
- 5) The software design specification should present information to demonstrate that the work performed by the software development engineers was clear and unambiguous, with minimum ad hoc design decisions – make sure that you have the full cooperation of your OS vendor
- 6) Submit a summary of the Life Cycle plan and the Life Cycle processes employed. It will be useful to include an annotated list of the control/baseline documents generated during the software development process, and a list or description of software coding standards.
- 7) Include verification and validation documentation and base it on the claimed Level of Concern. Validate design requirements early. Whenever software is changed, a validation analysis should be conducted to validate the specific change and also to determine the extent to which this change may impact the entire systems operation. Documentation and tracking is essential.
- 8) Include a revision level history of software generated during the course of product development.
- 9) Take advantage of modeling technologies. Choose a development platform that can easily integrate testing and management tracking information, easily upgrade legacy product code when improvements are added, or when underlying hardware has changed. Model Driven Development (MDD) technology is not only effective for meeting this need, but EMF data shows that the expected ROI derived from using MDD is significantly higher than that for development methods that don't employ MDD.
- 10) Traceability must be complete and audited to protect the CEO under the act. This should be considered a good design practice. However, it is important to initiate traceability early in the development process in order to gain management efficiencies. Consider automating traceability as a best practice.
- 11) Document third party code -know what's in your application even it's not yours: Software provided by a third party for which adequate documentation may not

exist is known as Software of Unknown Pedigree (SOUP). As a developer you can follow two paths: explain the origin of the software and the circumstances surrounding the software documentation, or use MDD to import the SOUP and apply rigorous testing and validation analysis to it. Some MDD tools allow SOUP to be either integrated into the product or to be treated as a separate legacy component for which analysis can verify and validate systems operation. Formal code certification tools can also be used to ensure that SOUP code is safe.

Clearly, there is a lot to be done. Many OSEs will be inappropriate to your specific application. If you can use a smaller OS with less memory and processor power requirements, you will not only save on development and product costs, but the steps necessary to comply with the medical accountability act will be less cumbersome and expensive.

Communications Security – Insuring Patient Privacy

President Obama's economic stimulus package contains a specific call-out for \$19 billion to be spent on digitizing medical records nationally.

I'm sure that developers don't need marketing input to see this as a compelling target.

Here are some facts that deserve consideration:

- 1) \$19 billion of the Stimulus bill has been designated for nationally digitizing medical records A major concern – and limitation of current patient record methodologies concerns patient privacy
- 2) These concerns are a focus of HIPAA and JCAHO which don't have a comprehensive security requirement established as yet – but hold considerable clout within hospitals and health care facilities. A solution based upon the NSA's FIPS 140-2 requirement would be an appropriate solution. If it's good enough for national security it should be good enough for medical record privacy concerns
- 3) Until recently, there has not been a suitable solution that can be easily and affordably applied to the communications end of embedded medical devices
- 4) The NSA along with NIST has published the FIPS 140- 2 (now 3) standard that requires communication equipment purchased by a government agency or prime contractor working on government projects to meet this standard
- 5) Even with a NIST and NSA certified encryption module, developers needed to go back to NIST for every new application, to insure that the product's application software didn't change the FIPS 140-2 stack. MDD can be potentially used to address this issue – and to reduce the work load of NIST, as the modeling environment can clearly show that the new application does not compromise the encryption module

Unfortunately, many networked embedded systems lack robust encryption to protect sensitive information. This may be due to resource limitations (strong encryption requires substantial processing, memory, and power), cost restrictions, design limitations, or possibly the extension of an internal, legacy, hard-wired system onto an open network such as Ethernet or IP, without considering the associated security implications.

Regardless of the reason, the potentially disastrous results are the same. Intruders or malicious insiders can read, intercept, modify, or remove communications at will. If proprietary wireless RF links are involved, the danger is further amplified, as anyone with suitable equipment can attack the system, potentially from a substantial distance given a high-gain antenna.

The industry is not without powerful encryption algorithms. An algorithm such as AES is unbreakable, but it takes a toll in terms of memory, power and hardware capabilities. Most embedded devices don't have the overhead to employ AES. Even if AES could be easily deployed, secure communications could not be assured unless the system were capable of authenticating exactly with what or whom it is communicating.

EMF has published details of embedded cryptography and how it can be deployed (*Avoiding an Embedded Security Disaster: What vendors, OEMs and developers need to know about embedded security* – www.embeddedforecast.com). We raise the subject to make medical device developers aware of the need for encryption security, particularly for medical devices that share information with medical record repositories, or that enable radiological remote viewing.

As we write this paper, a potential solution – one that avoids the limitations of previous efforts while being consistent with FIPS 140-2/3 – is in progress and may be available in the near future.

Renesas has developed a Board ID Security Stack (BSS) and has partnered with Express Logic and their ThreadX OS. BSS manages the Board ID, a tamper-proof hardware including private encryption keys and the software required to internally authenticate, which encrypts and decrypts independently of the operating system. This relieves the medical device from the high overhead that authentication and encryption usually requires of the system. The system uses ThreadX, with its small footprint and responsiveness, to perform ancillary tasks of importance to the medical data application.

Note that this is different from the OS certifications previously discussed, and should not be confused with internal security characterized under Common Criteria and MILS (Multiple Independent Levels of Security). The BSS/ThreadX device is concerned with insuring a strong authentication between approved medical devices and systems, and a safe and encrypted data communicated between hospitals, health provider agencies and physician's offices, whether data is being sent to or received from an approved medical data repository.

This is the most pressing embedded requirement for assuring the privacy of patient medical information to be achieved before digitizing patient medical records on a national basis.

Summary

In this report, EMF presented a clarification of the needs and an independent assessment of the challenges facing embedded medical device developers. Issues regarding certification, design outcomes, and OS choice criteria were discussed, as well as the demands that will shortly be imposed on developers and their companies by US Senate legislation.

In summary, EMF suggests that:

- Base your OS selection on the level of complexity presented by your design requirements. Make sure that your selection is appropriate to the application
- Decide whether your application requires a mission critical level design, and then choose whether to employ a previously certified OS, or insure the certifiability of your software through formal development methods. If the application permits, a non-certified OS that has been in successful use for millions of devices with similar complexities and design requirements can be used
- Your OS selection should be compatible with development tools that provide audit trails and documentation for requirements management and software control management tools (that include version control documentation, validation and verification tools, and testing/software verification tools). Importantly, your OS selection should be compatible with system modeling tools that permit the ability to visualize code execution (and the visual mapping of requirements onto code within the model) and the ability to be able to automatically upgrade legacy code to new hardware requirements
- Follow the suggested best practices that were presented to ensure that your designs will fall within the requirement of the Drug and Medical Device Accountability Act
- Don't attempt to bring a product to market that is used for transmitting patient information without FIPS 140-2/3 encryption