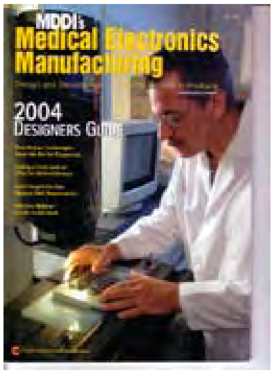
Intertech
ENGINEERING ASSOCIATES, INC.

Communications Protocol Testing:
Balancing Technology, Planning, and Compliance



by David A. Vogel, Ph. D. and David S. Bernazzani
both from Intertech Engineering Associates, Inc.

as published in Medical Electronics Manufacturing,
Fall 2004

The design and validation of communications protocols for intra-device communications and for external communications must be handled with the same care as other device software. This article addresses some of the design, validation, and regulatory compliance challenges related to communications protocols.

Testing the increasing number of microprocessors that need to communicate reliably with each other within medical devices poses not one but three different testing challenges: technical, planning (resources, schedule, budgets, etc.), and compliance. It's a rare electronic medical device that does not make use of microprocessors in its design. Increasingly, designs are taking advantage of low-cost microprocessors to partition functionality. This is a change from earlier designs in which more functionality was centralized into a more powerful general-purpose processor that relied on an array of external devices.

The benefits of a partitioned design philosophy include:

- Keeping hardware and software simpler and easier to develop and test.
- Keeping functionality separate (fire-walling) as a protective measure from random software defects.
- Facilitating parallel development and testing activities.

But, decentralized design philosophies also introduce new challenges to instrument designers, implementers, and testers. Challenges are especially apparent when the microprocessors in a device are to communicate directly with each other. When the integration of microprocessor-based subsystems are inadequately planned, implemented, and tested, it often leads to discovery of defects after instruments are released to

COMPANY PROFILE

Intertech Engineering Associates, Inc.

Address:	100 Lowder Brook Avenue Suite 2500 Westwood, MA 02090 www.inea.com - (781) 801-1100
Industry:	(Electro)Medical Devices
Services:	Assessments Training Consulting Hands-on Engineering
Skills:	Product Design Risk Management Requirements Engineering Electronics Development Software Development Software Verification and Validation Production/Quality System Software Validation

the field. The integration testing of large multiprocessor systems should focus on interfaces, many of which are communications interfaces.

Compliance Challenges

FDA's guidance, "General Principles of Software Validation," mentions communications or interfaces several times under the section relating to "Activities and Tasks – Typical Tasks Supporting Validation."[1]

The guidance recommends that "typical software requirements specify … the definition of all external and user interfaces, as well as any internal software-to-system interfaces." It points out that "external interfaces are those with other software (including operating system software), system hardware, and the users." This definition covers quite a bit of territory. This article focuses on communications between microprocessor-control subsystems within a device and digital communications with external devices or computers.

As device development progresses from requirements definitions to design, the guidance recommends that "the software design specification should include … communication links (links among internal modules of the software, links with the supporting software, links with the hardware, and links with the user)."

Furthermore, as designs are evaluated by peer team members and reviewed with management, the guidance recommends that "an analysis of communication links should be conducted to evaluate the proposed design with respect to hardware, user, and related software requirements."

The guidance also covers testing of communications within a device under integration-level testing. It suggests that "integration-level testing focuses on the transfer of data and control across a program's internal and external interfaces." Further, the guidance says that "these tests should provide a thorough and rigorous examination of the software product's compliance with its functional, performance, and interface definitions and requirements."

It is important to understand and address FDA's suggestions for the validation of communications interfaces. But, is that the only reason to undertake a serious validation test effort? Defects in communications software are notorious for escaping detection under laboratory test conditions only to surface shortly after release of a device to the marketplace.

Some field defects are benign and often are not even identified as defects. Others, however, can become a nuisance to users, and still others can create safety issues. The effort expended on validation should be proportionate to the complexity of the communications software (i.e., likelihood of failure) and the severity of the consequences of its failure. If the consequences include a risk of harm to a user, the validation effort should be substantial. A substantial communications software validation effort might also be justified if a communications software failure could trigger a device recall, resulting in financial harm to the device manufacturer.

Planning Challenges

Beyond the quality planning suggested by FDA's guidance, considerable effort should be put into the technical planning for development and testing of device communications.

Communications software is unique in that the software for both sides of the communication needs to be working to fully exercise the software on either side (unless some type of simulation is used). Planning must establish the order in which the communicating pieces of software are to be brought up and whether communications simulators are needed to help debug the software.

For similar reasons, testing communications software should also be planned carefully. It is quite common to design special connections and switches into hardware to facilitate development and testing. In some cases, special equipment must be purchased, rented, or custom-designed to fully test a communications interface. Unexpected costs and delays in the development and test schedules may result if this is not planned adequately in advance.

Communications testing can also require hardware modifications to allow access to the communications signals. Part of the planning for communications testing should consider this to ensure that required switches, connections, and other monitoring and control access points are factored into the hardware and software development plans.

Perhaps most important, communications testing is like all other forms of verification and validation testing in the sense that software cannot really be tested if the test engineer doesn't know how it is supposed to work. As with most validation efforts, detailed, testable requirements are perhaps the most important ingredient for success. Useful communications testing requires that a detailed protocol requirements specification be in place before initiating testing activities. These specifications are also called *interface requirements specifications, communication protocol specifications,* or *interprocessor communications specifications.* These specifications should include high-level descriptions of:

- Communication timing requirements
- Message initiation and acknowledgement sequencing
- Details on message structure (envelope, synchronization patterns, error checking and correction, addressing, sequence numbers, etc.)
- Handling of unexpected or out-of-sequence messages
- Command groups
- Error detection, reporting, and recovery
- Resynchronization details.

Additionally, a command dictionary should include details on:

- The structure of each command
- Data elements within each command (i.e., data types, valid ranges, etc.) and handling for out-of-range data elements
- System events that trigger each command (e.g., alarm messages versus routine heartbeat messages triggered every *n*th second)
- Acknowledgement details (ACKs) and recovery from negative acknowledgements (NAKs).
- Command-specific requirements for timing.

Finally, a good communications specification also includes use of case-based data-flow diagrams that show typical communications sequences that are anticipated in a normally functioning system, as well as in adverse system conditions such as dropped communications, nonsense message receipt, alarm conditions, system power up and down, and restart.

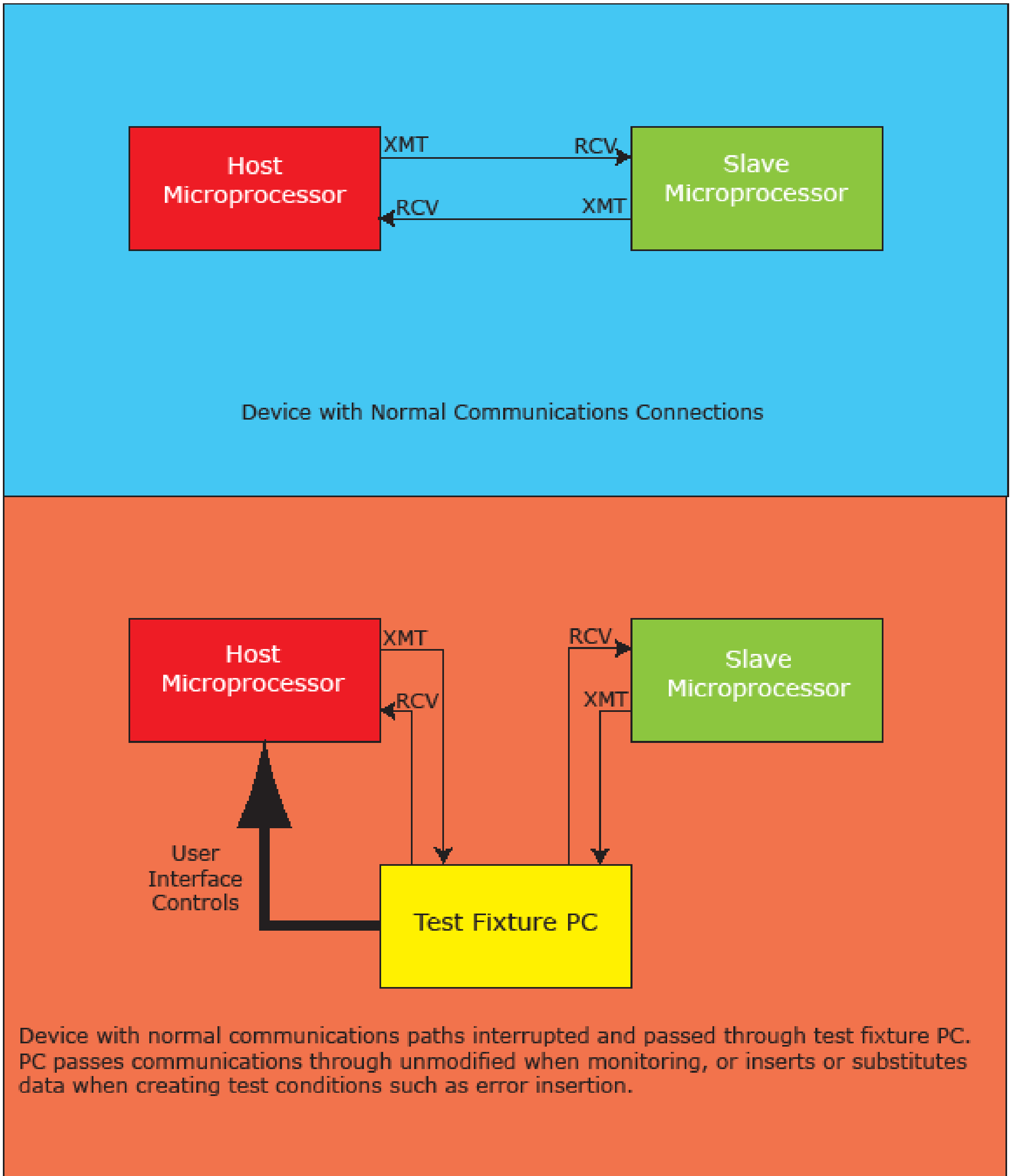
Technology Challenges

Communications via Ethernet, RS-232, or RS-422, etc., with external devices or computers can be monitored and sometimes tested with commercial off-the-shelf test equipment. More often than not, these devices are communications monitors, and, while useful for examining the communications traffic on the communications medium, they are typically limited in their ability to insert errors to test the handling on both sides of the communications.

Development systems, specifically debuggers and in-circuit emulators, are often used to test communications software. The procedure is to set breakpoints in the communications software to stop the system so that the test engineer can view the contents of a buffer that is full of information received or about to be transmitted. This approach is useful for some testing; however, it is labor-intensive and time-consuming, and it requires a skilled tester. This approach also is limited in its ability to test a system in full operation at its stressed timing limits.

Most communications protocols require rapid sequencing of messages and timely responses that make testing difficult when using debugger breakpoints. Often the testing is confined not only by the technical limitations of the method, but also by the difficulty of setting up test scenarios, thus making it difficult to capture and examine sufficient data. Test engineers often try to get by with as few test scenarios as possible.

A more general approach to communications testing that works well for external communications – as well as intra-device inter-processor communications – is to insert a test computer (usually a PC) between the two



Device with Normal Communications Connections

Device with normal communications paths interrupted and passed through test fixture PC. PC passes communications through unmodified when monitoring, or inserts or substitutes data when creating test conditions such as error insertion.

FIGURE 1

processors of a given communication. Figure 1 shows this configuration for two processors that communicate bidirectionally on separate transmit (XMT) and receive (RCV) lines. The test computer intercepts transmissions from each microprocessor and relays them to the intended receiving processors.

This configuration gives the test computer the opportunity to check the validity of the communication against the expected result, or to insert errors to test the error-recovery requirements. Generally, the test computer can intercept and relay communications data quickly and keep the protocol timing within specifications. Additionally, timing delays can be inserted into the communications exchanges to test the software at its specified timing bounds. The test computer in the communications path can easily simulate dropped messages, out-of-sequence messages, unexpected messages, etc.

The wide arrow in Figure 1 labeled "user interface controls" indicates that the test computer functionality is often extended to enable it to control – and even to monitor – the user interface of the device under test. This control makes it possible to create scripted tests that run automatically as directed by the test computer.

Instrumented Communications Testing

The overhead involved in instrumenting a device, developing test computer software, and designing test protocols is significant. Why would one want to go to so much trouble to test communications? A manufacturer might not always want to. If the communications are simple and are for noncritical, non-safety-related functions, a manufacturer might not need to incur these overhead costs. The resulting benefits may not justify the investment.

However, in most applications, intradevice communications are critical to the safe operation of the device and probably warrant the testing techniques described in this article. Major advantages include:

- Testing that can't be done without instrumentation. For example, some tests may require complex sequences to set up for the actual test situation. Using a debugger to set breakpoints and force values for the setups would violate timing specifications, thereby making it impossible to test the interface software untouched. The test computer can be scripted to automate the setups in a real-time environment with fully featured communications software that does honor-timing specifications.
- Timing analyses. Some timing analyses can be done using an off-the-shelf communications monitor; however, a test computer can be scripted not only to set up the scenarios for testing key timing specifications, but also to log the maximum and minimum delays encountered during a test run. The test computer can insert out-of-specification timing delays into a system to test the system's response to the delay and its ability to recover from such a delay.
- Regression Testing. Testing communications by manual means is time-consuming. There is a temptation to cut back on communications tests on regression runs simply because they take so much time. Small changes in the device software can significantly affect system timing, which can, in turn, greatly affect the communications timing. An instrumented communications test system can be automated to a large extent to allow automated or semi-automated regression runs of communications tests as the device software evolves.
- Long-run tests. In some systems, communications events happen infrequently, requiring long tests to capture enough events to complete a meaningful test. Automated scripts accomplish this nicely with minimal manual labor. Testing professionals recognize that the one parameter that most closely correlates with finding defects is the number of hours of test time on the device under test. Automating the test process to cycle through a number of use scenarios repeatedly over a long time period can be valuable. Often, it enables the testing engineer to find defects that would otherwise show up only in the field.
- Stress tests. Stressing communications is also difficult – if not impossible – to do by manual methods, but it is the most likely scenario in which the communications will fail. An instrumented test system can be used to stress a communications link because it pushes communications at a maximum rate for extended periods. It also inserts error conditions that stress the error detection and recovery processes and further use up bandwidth to deliver the same content. Test systems implemented with user-interface

controls can further stress the system by driving the user interface at speeds that would be difficult for a user to sustain for any reasonable period. This stress can help flush out timing problems between user-interface stimulated communications and the communications timing

Configuration-Management Issues

A few configuration-management issues are special to communications interfaces that are embedded in devices. Computer-based test fixtures need to be revised for most changes to the communications protocol because their scripts are so closely tied to the version of protocol. Like the device software, the test-fixture software should be under version control and documented in the configuration-management system.

The configuration management system should document which version of communications protocol is implemented on each version of the device software. It is important to realize that the communications protocol is the language spoken between two intra-device processors or between the device itself and an external processor.

To get all components to work together, it is important to know whether they all speak the same language. Each version of software should document the protocols with which it is compatible. It is also important for maintenance and support of the device to understand which field-replaceable units are compatible with each other.

Conclusion

Communications interfaces in medical devices are notorious for their ability to defy traditional manual test efforts for exposing communication software defects. A careful, systematic approach to communications-requirements identification, evaluation, and review is superior to an iterative, evolutionary requirements-discover process that takes place at the programmer's keyboard.

Solid requirements also facilitate a systematic testing approach in which a verification test can be created for each communications software and protocol requirement. Such testing can be quite technical and involved, and so it requires substantial planning, consideration of test fixture connections in the device design, and development of test fixtures and test-fixture software.

Testing medical device microprocessors that need to communicate with each other presents technical, planning, budgetary, and compliance challenges. As designs increasingly take advantage of low-cost microprocessors, their integration can lead to complex issues with communications interfaces. Unless these issues are addressed carefully, manufacturers may find themselves trying to identify defects – both software and hardware – after a device is already on the market.

The investment in proper testing communications protocols is significant, but so too are the benefits. It's far less expensive to spend the money up front than to have to fix defects in malfunctioning devices that are already in use in clinical settings.

References

[1] "General Principles of Software Validation; Final Guidance for Industry and FDA Staff" FDA, 2002, [on-line] Available from Internet: http://www.fda.gov/cdrh/comp/guidance/938.html.

ABOUT THE AUTHORS:

David Vogel is the founder and president of Intertech Engineering Associates, Inc.

Dr. Vogel was a participant in a joint AAMI/FDA workgroup to develop a standard for Critical Device Software Validation which was subsequently included in the IEC 62304 Software Lifecycle Standard. He was also a participant on the joint AAMI/FDA workgroup to develop a Technical Information Report (TIR) for Medical Device Software Risk Management. Currently, Dr. Vogel is a member of the AAMI/FDA workgroup developing a TIR on Quality System Software Validation.

A frequent lecturer for workshops and seminars on topics related to medical device development and validation, Dr. Vogel also is the author of numerous publications and holds several patents.

Dr. Vogel received a bachelor's degree in electrical engineering from Massachusetts Institute of Technology. He earned a master's degree in biomedical engineering, a master's degree in electrical and computer engineering, and a doctorate in biomedical engineering from the University of Michigan.

David Bernazzani is a principal software engineer at Intertech Engineering Associates, Inc.

Mr. Bernazzani has managed several teams in the design, development and implementation of software specifically designed for medical instrumentation. Most recently he led a team that designed, developed and implemented a graphical user interface for use in an organ transplant device centered on the Motorola MX1 series processor. His primary areas of responsibility involved writing and debugging device drivers for the MX1, designing and implementing the communications layer (both hard-link and BlueTooth) as well as extensive work with the Nucleus RTOS and the Metagraphics windowing library.

Mr. Bernazzani received a bachelor's degree in computer science from Northeastern University.

ABOUT INTERTECH:

Intertech Engineering Associates has been helping medical device manufacturers bring their products to market since 1982. Through a distinct top-down service model, Intertech offers high-level consulting and hands-on engineering. By balancing technical expertise and practical business experience, we support clients through all phases of product development. While we do make your job easier, Intertech exists not to replace but to partner with clients to help balance the concerns of quality, time and cost.

With considerable experience in FDA regulatory compliance, our time-tested development process can anticipate and solve problems inexpensively on the planning board rather than through costly solutions later in the development, test, or post-deployment phases. By using deliberate processes, Intertech ensures an improvement in quality and can build client expertise.

Call us today for more information or a free consultation at 781.801.1100

Intertech Service Offerings:

Risk Analysis and Management
Software Design and Development
Electronic Design and Development
Requirements Development and Management
Documentation and Traceability
Verification and Validation
Evaluations, Reviews, Inspections
Planning
Project Management
Compliance Consulting and Training
Manufacturing and Quality System Software Validation