Automated RDM responder testing

BY SIMON NEWTON

Using open source software to save time and money

WHEN A DRAFT STANDARD is finally sent to ANSI for ratification members of the PLASA Technical Standards Program breathe a collective sigh of relief. Years of work is finally over, but for lighting manufacturers the work is just beginning. Around the world, engineers purchase the standard, interpret it as best they can, and start building the new functionality into their next product.

Unfortunately natural languages like English are not well suited to technical specifications. There are often multiple ways in which directives can be interpreted and it is impractical to cover all possible corner cases in the standard. Attempts to do so would cause the document to grow to hundreds of pages long and significantly increase amount of time required. The result of this means that published standards may be vague and lacking in some areas, leading to devices that, while arguably technically complying with what's written in the standard, fail to communicate with each other.

The PLASA Plugfests assist in addressing this issue. Engineers from different manufacturers meet informally and work together to solve interoperability problems. Every year more manufacturers realize the benefits to be gained from the Plugfest, and the event has seen solid growth in both the diversity of manufacturers and the amount of hardware available for testing.

But this approach does not scale. With the number of responders increasing, it is no longer feasible to perform in depth testing of all responders during the weekend of the Plugfest. On top of this, many manufacturers can't attend the Plugfests and the six month frequency slows down product development and bug fixes.

It became clear to a number of us in late 2010 that automated testing could be used to solve this problem. The development of a full compliance test would have been prohibitively expensive and time consuming but many of the benefits of automated testing could be gained from an application level RDM responder test suite.

I wrote the RDM responder tests over the following two months and they were first used at the January Plugfest in Texas. I've since improved them to the point where they are ready for general use; and they were released on 6 March 2011.

The tests form part of the Open Lighting Project (http://opendmx.net/index.php/Open_Lighting_Project), a collection of efforts to accelerate the adoption of new protocols within the industry and to drive innovation. The test platform uses the Open Lighting Architecture software to communicate with USB RDM devices acting as RDM controllers as shown in **Figure 1**. My software currently supports two widely available USB RDM devices and support for others can be added using the plugin framework provided.

Test layout

The RDM responder suite consists of a series of tests written in Python, each designed to exercise a different section of a responder's handling code. I chose Python because the language is simple to learn and flexible enough to write succinct test cases. Some tests are very simple, such as sending a GET message and checking for a response, while others are more complicated, such as performing a sequence of operations on the responder and then checking that the responder's state matches what was expected.

Every year more manufacturers realize the benefits to be gained from the Plugfest . . .

Individual tests are grouped into categories, which correspond to the PID groupings in the E1.20 standard. At the end of the test suite execution each test will be in one of four states: *passed*, *failed*, *broken* (indicates a logic problem with the test

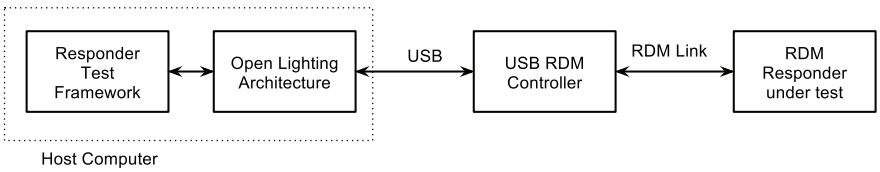

Figure 1 – Hardware and software components used for RDM responder testing

itself), or *not run*. **Figure 2** shows the output from a typical test run:

BY CATEGORY		
Control:	8 / 8	100%
Sub Devices:	1 / 1	100%
Configuration:	9 / 9	100%
Product Information:	17 / 17	100%
Network Management:	4 / 4	100%
Core Functionality:	2 / 2	100%
Error Conditions:	86 / 86	100%
Display Settings:	4 / 4	100%
DMX512 Setup:	9 / 9	100%
Power / Lamp Settings:	12 / 12	100%
Sensors:	6 / 6	100%
Status Collection:	2 / 2	100%
160 / 162 tests run, 160 passed, 0 failed, 0 broken		

Figure 2 – RDM Responder test output for a bug-free device

Verbose mode can be enabled by passing the -v flag when executing the tests. This prints out every RDM message sent and received from the responder, as well as what the expected response(s) are.

Warning and advisory messages

Individual tests can also output warning and advisory messages. While not important enough to declare the test a failure, these messages alert the user to possible problems with the responder. Warning messages indicate behavior that, while not matching the E1.20 standard, is unlikely to cause usability issues, for example: declaring support for a required PID like DEVICE_INFO in the supported parameters section. Advisory messages indicate issues that are not covered by the standard but may cause problems with controllers such as a sensor temperature where the reported value is outside the stated scale range.

Plugfest—get your geek on!

The PLASA Control Protocols Plugfest was developed in January 2009, then as part of ESTA, by members of various Control Protocols Task Groups. They represent a variety of manufacturers and twice a year, they are all in the same place at the same time. They bring in some lights and consoles and lots of other bits and pieces and, maybe not put on a show, but get a lot of work done. The event was originally created to support the Remote Device Management protocol (ANSI E1.20 RDM) and has expanded to cover all CPWG protocols, including the popular ANSI E1.31 DMX over Ethernet protocol. As the content grew, so did the attendance and, consequently, the space. Beginning in a small suite in 2009, Plugfest moved this year into a large suite with room to grow. Why not meet in a standard meeting room? Well, the schedule may say it ends at 10:00 p.m., but the geek squad may often be found working long past that time, often into the wee hours of the morning. They need their freedom, and a place for bagels, scones, and coffee.

So that's how it started. What it has become is an extremely efficient way for manufacturers' engineers to make sure your products will work with the products of other manufacturers. If it doesn't, you can find out *on the spot* what is wrong, fix it, and try again. People from all over the world are bringing their gear to Plugfest just to improve their performance. As a result, there is a huge pool of knowledge and experience in Texas for the weekend. And most of the members of the RDM and E1.31 task groups (they wrote the standards) are there. They are happy to answer your questions and talk about protocols. Plugfest is one more way that the Technical Standards Program brings colleagues and competitors together to work on furthering our industry. Get *your* geek on!

The July Plugfest will run from Friday afternoon July 22 through Sunday evening July 24 at the DFW Marriott Solana in Westlake, TX. If you are interested in participating, please contact Scott Blair at sblair@rdmprotocol.org to register. Plugfest is open to all, but, with its growing popularity, we have to coordinate participation to make sure everyone has a space at the table.

Test dependencies and ordering

Some tests require the information from the output of other tests in order to run correctly. As an example, a test that enumerates the sensors needs to cross-check the result with the number of sensors declared in the DEVICE_INFO message. This dependency structure is achieved through the use of test properties. Continuing with the sensor example, the DeviceInfo test can declare that it provides the *sensor_count* property and then the EnumerateSensors test can declare that it requires the *sensor_count* property. This ensures that EnumerateSensors will be run after the DeviceInfo test and that it will have access to the required information.

On each execution of the test suite, the available tests are topologically sorted based on the dependency graph and then executed. If only certain test cases are of interest, the tests to be run can be restricted through the use of a command line flag.

Guidelines for writing tests

Writing good test cases can take some practice. Along with the usual best practices for testing there are a number of guidelines that can make the RDM responder tests more effective:

Tests should be strict in what they accept. Through the use of dependencies tests can limit the number of acceptable responses. For example in a test for an optional PID such as DISPLAY_INVERT, rather than simply accepting an ack or a nack the tests can use the output of SUPPORTED_PARAMETERS to decide whether an ack or a nack-unsupported-command-class is expected.

All tests should include a documentation string. This provides comments in the code to explain what the test is doing and is printed to the output when verbose mode is enabled.

The dependencies of each test should be limited to those that are strictly necessary. A test requires all its dependencies to pass before executing, which means the more dependencies that are listed the lower the chance the test will be run on a responder with bugs. This limits the amount of useful debugging information available during the initial test runs.

Caveats and limitations

My test suite does not check for RDM timing issues. This is because the level of timing information that the USB RDM controllers provide back to the host software is limited and I have not had the time to work with the controller developers to export this information. The recommended method for testing for timing issues is to attach an RDM sniffer to the line, run the test suite, and then evaluate the timing information.

It should be noted that the tests are simply my interpretation of the RDM standard. Where the standard is unclear, I discuss the issues with other members of the E1.20 task group, in person or, on the RDM Protocol forums at http://www.rdmprotocol.org/ and together we make a decision on what the correct behavior should be. In a few rare cases the tests will accept multiple behaviors.

Like any software, the tests themselves may contain bugs. The tests have been run on a variety of responders and in the event that bugs are discovered, emailing the Open Lighting mailing list will result in the bug being fixed. Code patches to fix existing tests or add new ones would be greatly appreciated.

Common mistakes

During my testing a number of responders exhibited similar bugs. I've discussed them here in the hope that other RDM developers will avoid making the same mistakes.

The condition that results in the highest number of test failures is inadequate checking of parameter data, both in terms of the request structure as well as parameter values. Many responders neglect to perform strict checks on this data which results in malformed requests being treated as valid requests rather than returning a nack format error. While this makes the responder in question appear to work in almost every case, it means that bugs in controller software can go undetected. When a responder with strict data handling is added to an RDM network it will then fail to work correctly, possibly resulting in technicians wasting time trying to debug the wrong device. The responder test suite checks for these conditions by sending malformed requests to the responder and checking that nack format errors are received.

The test software is open source which means that code is available and can be modified and used without any fees.

On a similar note, another recurring fault is buffer overruns. PIDs such as SENSOR_DEFINITION have the ability to specify a sensor number (offset) in the request. Some responders omit the bounds check for these offsets, resulting in get requests returning data from memory which is used by other data structures. An even more serious problem is the lack of bounds checking for set requests such as CAPTURE_PRESET as this allows otherwise-inaccessible memory locations to be set to arbitrary values. Other bugs encountered include the handling of non-ascii data for PIDs such as DEVICE_LABEL, as well as dealing with empty and full length (32 character) strings.

The broadcast (0xffffffffffff) and vendorcast (0xmmmmffffffff) UIDs also cause problems. Devices must never respond to any messages sent to these UIDs but still need to take action when sent set requests. A similar situation applies to the SUB_DEVICE_ALL_CALL value but in this case the device must respond to get requests with a nack reason of SUB_DEVICE_OUT_OF_RANGE.

Another common bug is the handling of delayed or slow writes. Some responders

ack set requests even though they defer writing the new value to persistent storage. This means that a controller that sends a get request immediately following the set will see the old value and assume that the set failed. If delayed writes are used and the responder cannot confirm that the data has been written to storage in the time required to send the RDM response, an ack_timer must be issued. This notifies the controller that the write has not taken place and the controller can then query for queued messages to confirm the update.

Running the test software

The test software can be downloaded from the Open Lighting site http://opendmx.net/index.php/RDM_Responder_Testing and runs on Mac OS and Linux. It can also be run in Windows using VMWare (instructions are on the website). The software and hardware required to run the tests will also be at the July Plugfest in Texas for others to use (see dates/times in accompanying sidebar). The test software is open source which means that code is available and can be modified and used without any fees. Manufacturers can extend the tests to cover manufacturer specific PIDs, avoiding the need to develop their own test framework.

Acknowledgements

I would like to thank Hamish Dumbreck and Eric Johnson who developed additional firmware for the USB RDM controllers that made this automated testing possible. ■

Simon Newton has been interested in lighting control systems since middle school and founded the Open Lighting Project in 2004 with the aim of accelerating the adoption of new control protocols by the industry. He is a member of the Control Protocols Working Group and contributes to the RDM and RDM over IP standards. His day job finds him designing and building the serving infrastructure for a large Internet company in Silicon Valley. Simon can be reached at nomis52@gmail.com.