# Upgrading Custom Simulink Library Components for use in Newer Versions of Matlab

Camiren L. Stewart

GS-07, Level 01

Electrical Engineering, Masters of Engineering

University of Cincinnati, Cincinnati, OH.

Summer, 2014

# Upgrading Custom Simulink Library Components for use in Newer Versions of Matlab

Camiren L. Stewart[1]
*University of Cincinnati, Cincinnati, Ohio, 45220*

## Nomenclature

| | | |
|---|---|---|
| *COTS* | = | Commercial Off-The-Shelf |
| *CMS* | = | Configuration Management Software |
| *CSCI* | = | Computer Software Control Item |
| *DWL* | = | Development Workstation (Linux Environment) |
| *E2E* | = | End to End |
| *GOTS* | = | Government Off-The-Shelf |
| *GSE* | = | Ground Support Equipment |
| *KSC* | = | Kennedy Space Center |
| *LOC* | = | Lines of Code |
| *LCS* | = | Launch Control System |
| *OS* | = | Operating System |
| *OSI* | = | Operating System Image |
| *NASA* | = | National Aeronautics & Space Administration |
| *PLC* | = | Programmable Logic Controller |
| *PSP* | = | Procedure Support Patch |
| *RTW* | = | Real-Time-Workshop |
| *SCCS* | = | Spaceport Command and Control System |
| *SIM* | = | Simulation |
| *SSH* | = | Secure Shell |

## I. Introduction

The Spaceport Command and Control System (SCCS) at Kennedy Space Center (KSC) is a control system for monitoring and launching manned launch vehicles. Simulations of ground support equipment (GSE) and the launch vehicle systems are required throughout the life cycle of SCCS to test software, hardware, and procedures to train the launch team. The simulations of the GSE at the launch site in conjunction with off-line processing locations are developed using Simulink, a piece of Commercial Off-The-Shelf (COTS) software. The simulations that are built are then converted into code and ran in a simulation engine called Trick, a Government off-the-shelf (GOTS) piece of software developed by NASA.

In the world of hardware and software, it is not uncommon to see the products that are utilized be upgraded and patched or eventually fade away into an obsolete status. In the case of SCCS simulation software, Matlab, a MathWorks product, has released a number of stable versions of Simulink since the deployment of the software on the Development Work Stations in the Linux environment (DWLs). The upgraded versions of Simulink has introduced a number of new tools and resources that, if utilized fully and correctly, will save time and resources during the overall development of the GSE simulation and its correlating documentation. Unfortunately, simply importing the already built simulations into the new Matlab environment will not suffice as it will produce results that may not be expected as they were in the version that is currently being utilized. Thus, an upgrade execution plan was developed and executed to fully upgrade the simulation environment to one of the latest versions of Matlab.

[1] KSC Pathway Intern, NE-C1, Kennedy Space Center, University of Cincinnati

## II. Upgrade Direction

The current release that the SCCS Simulation Product group is using is Matlab R2009b that comes packaged with Simulink 7.4. After operating in this environment for five years, it was decided to upgrade to a newer version to not only take advantage of new features, but to also avoid the use of deprecated features that were removed between R2009b and the current version. As of the writing of this paper, the current operating system (OS) that is loaded onto the DWL machine is Red Hat® Version 5. According to Mathworks, Matlab R2013b and later versions require Red Hat Version 6. For this reason, it was decided that the upgrade will be to R2013a and its correlating Simulink version (Simulink 8.1). Refer to Figure 1 below for more information regarding Matlab version restrictions.

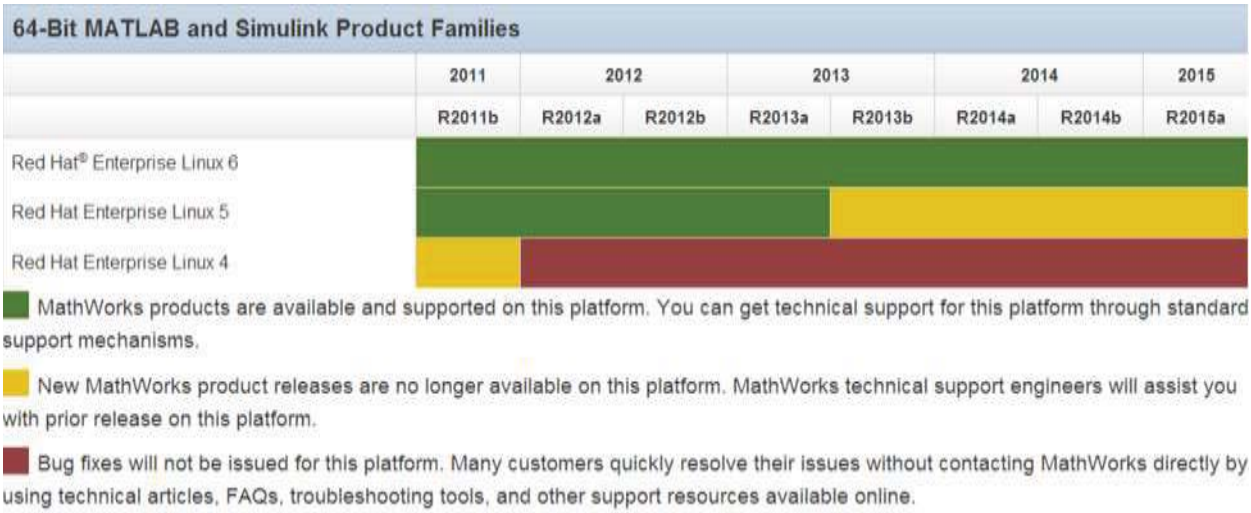


*Figure 1 - 64-Bit MATLAB and Simulink Product Families*

Apart from Matlab needing upgraded, additional toolboxes and utilities will need to also be patched or upgraded. One of these pertinent tools is the Trick (Procedure Support Patch) PSP. This is an add-on provided by Mathworks™ that enables a channel for communication to be open between Matlab 2013a and Trick. When Real-Time-Workshop (RTW) is used to convert the Simulink models to C-code, an execution environment needs to be selected. Installing the Trick PSP enables Matlab to select Trick as a target execution environment. The code needed to re-interface Trick with the code generated from RTW was patched and delivered by Mathworks. Fortunately, this is a simple task to complete as it only needs to be installed.

On the current Operating System Image (OSI), Matlab 2013a and the new Trick PSP does not exist. Using coordinated efforts with the Simulation Modelers, Simulation Framework Engineers, and the OSI group, the trio worked together in order to obtain an image that would replace the current image that is on the DWLs. DEEDWL052 (hostname) was used as the basis for our image. After the SIM Framework and Modelers were content on the configuration which was inspected via Secure Shell (SSH), the image was deployed on a selected few (three) machines for further evaluation. After the machines were successfully evaluated, the Simulation team reported back to the OSI group the additional changes that were needed. After the OSI group mediated the changes, the new OSI image that contained Matlab R2013a and the upgraded Trick PSP were deployed on all the SIM machines.

One more overall environment change was needed, and that had to deal with the Configuration Management Software (CMS) tool, AccuRev. Within the AccuRev environment, there exists many versions of the custom Simulink library components that are used to build the GSE simulations. Specifically, there exists a SIM_MODEL_libs repository that holds all development work for the custom library components. This includes the electrical library, fluid library, gas library, and cross component library along with each unit test for the library. Additionally, there is a copy of the component library that exists in every subsystem stream; there currently exists 55 subsystem streams in AccuRev. Initially, it was thought the developers could make changes to library components in the subsystem stream and then cross-promote that component to SIM_MODEL_libs. This created an inherited risk that not all library components

will get upgraded in the subsystem stream. This will occur if there is an overlap of a component between the subsystem stream and SIM_MODEL_libs. To mitigate the risk, the AccuRev library streams were collapsed to one central library stream that currently exists at SIM_MODEL_libs. Not only will this enhance the upgrade progress, but it will also help out every day business in SIM by enabling all subsystems to catch changes to components immediately without having to cross-promote to SIM_MODEL_libs. This action will expedite issues that might exist when changes are made to components since the subsystem model will catch the changes immediately.

The plan of attack was to have all modelers promote any external or modified files that live in subsystem workspaces (i.e. not SIM_MODEL_libs) if they wish to keep them. A change palette would then be built (but not committed) for each stream to see what files differ. That information would then be "kept" in AccuRev and then cross referenced to see what files have updates from the potential multiple sources. The ones with single sources would get promoted and the ones that have edits from more than one stream go through a separate resolution process (merge being the last option). After all discrepancies are ironed out, the libs directory in each subsystem stream would then be cross-linked (Xlinked) in AccuRev and pointed to a filter named SIM_MODEL_libs_FILTER_noTests. This filter will obviously host all of the library files and will exclude all of the test files.

## III. Upgrade Process

The library component upgrade process is a list that must be executed in the numerical order that is described in the following sections in order to achieve favorable results. These Matlab scripts are separated by library (electrical, fluid, & gas) and would have supporting scripts that would be called when the same operation is needed for all three libraries. The default operation is described below with technical outliers for each library called out when needed.

First, the custom library components for each library need to be loaded, saved, and closed in the version to which the library components are being upgraded. There are upwards of 50 new variables that are presented in R2013a library blocks when compared to R2009b library blocks. Organization of certain items has also changed. Additionally, not only do the library files need to be saved in the new version, but so do the test harnesses – for the same reasons as the library files.

Next, updates that are classified as manual (are not upgraded with any tool or script available from the standard Matlab load) are executed. This manual process is scripted out so that it can be automated in a sequential manner with minimal hands-on time from an operator. A number of these manual updates revolve around errors that are produced with the primitive Simulink blocks. For example, there needs to exist a new component that safely divides two or more numbers. In the new version of Matlab, there exists an error (and the model will fault) if a divide by zero occurs. Thus, the creation of the component X_Safe_Divide was created and replaces all of the primate divide blocks that exist in any custom library block. The same goes for primitive reciprocal blocks. In addition to the manual updates to library components, this is also where updates to the ksc_template and the places it is used are executed. An example of this would be everywhere that there is a test harness that is built for library component testing. There are only a number of changes that need to occur to the template, but they are all directly related to satisfying Model Advisor standards that are set. Apart from these checks, the electrical library has additional logic in its script in this section due to the Programmable Logic Controller (PLC) components not following the same standard to which the remainder of the custom component library adheres.

After the manual updates are processed, the script moves on to automatic updates that can be accomplished using given tools and functions that come pre-loaded with the Matlab and Simulink software. There are a number of changes that have taken effect in Simulink 8.1 (upgrade version) versus Simulink 7.4 (current version) regarding deprecated variables and additional changes to the way the primitive Simulink blocks are built. An example of this is the omission of the square-root function in the Math Function block; now square-root is its own primitive Simulink library block. The "slupdate" command will take care of these all these changes. The only dependencies would be having the custom library component in a model that can compile. Fortunately, it is a standard that all the custom library blocks that are built should be unit tested and documented. This makes the overall process relatively straightforward to carry out: The library link to will be disabled in the correlating test harness, the "slupdate" command will be executed, and the changes will be propagated to the library component from the test harness and saved.

Finally, after all of the updates and patches are appropriately applied to the custom library components, they will need to be regression tested. Unfortunately, there is a change to the file hierarchy that comes with upgrading to Matlab R2013a. Matlab 2013a outputs a different file structure while running unit tests for library components. In summary, the component's *_Test_report.html is embedded in a new folder named ./outputs/*_Test_report.html and the coinciding *_Test_results.mat file is located directly in the output folder. This is better illustrated in the figure below.

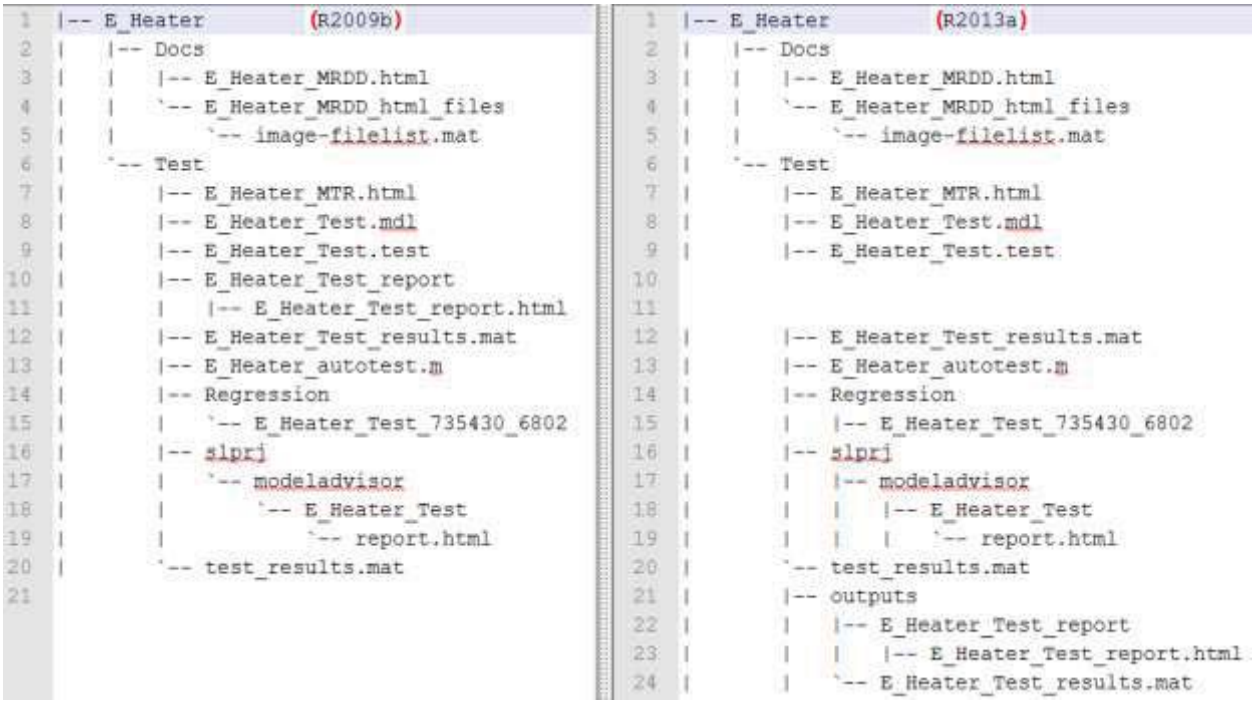


*Figure 2 - Comparison of R2009b and R2013a unit test output file structure*

In order to solve the issue of having multiple *_Test_results.html & *_Test_results.mat files, one of which is left over from the R2009b version of Matlab, the files will be moved in AccuRev to the new file structure that R2013a outputs. This shuffle will need to occur for every custom library component that has a full unit test written for it. This is not part of the individual library upgrade script; however, there is a script that is written, composed of Matlab commands, AccuRev commands, and Linux commands, that will accomplish this task. After the test successfully finishes, the results will need to be reviewed and then promoted to AccuRev.

Lastly, apart from upgrading the custom library components, there is one more task that will need to be executed in order to complete the upgrade: the individual GSE simulation files for each subsystem will need to be upgraded. Like the library components, a simple, load/save/close will need to be executed as well as updates to the configuration parameters, similar to the test harness configuration parameters, as each subsystem simulation file uses ksc_template (a template with preconfigured configuration parameters that is used for unit testing and simulation models). This script is written and is meant to be executed by the owners of each subsystem model. The above process is fully automated and direction is given through the script on how to upgrade the simulation file successfully.

Naturally, the above does not fully explain the full process of the 1,400 lines of code (LOC); however, it does provide an overview of what the upgrade process entails. There are many safety nets built into place that will execute to avoid breaking the current operating state in the Simulation Product Group after the upgrade is completed.

## IV. Conclusion

During the lifecycle of software development, there inevitably comes a time that COTS software is patched and upgraded which will include new features to further support development. Upgrades come at a cost, but so does not

upgrading. In the case of the SCCS simulation team, it took a couple months of writing code, testing that code, and coordinating with other groups, but that upgrade is worth it in the long run – especially with a project that is spanning the number of years like SCCS is currently planned.

## V. Acknowledgments

First, I would like to thank Michael Waldorf in his support to this project. A NASA Contractor, Mike served in an un-official mentor role and was someone I could go to in order to exchange ideas for better ways of solving problems. This project would have taken much more time to complete if it wasn't for his guidance. Secondly, I would like to thank the Pathways Program for this opportunity to be part of the NASA team and contribute to human spaceflight. Lastly, I would like to thank Lien Moore and Cheryle Mako. These pair are an extraordinary pair of NASA employees that provide an endless amount opportunities for all of their mentees, myself included.

## VI. Works Cited

The MathWorks, Inc. (2014, July 01). *Platform Road Map for the MATLAB and Simulink Product Families*. Retrieved from MathWorks: http://www.mathworks.com/support/sysreq/roadmap.html