



McMaster Centre for Software Certification

Guide and Reference Manual to Documenting Simulink Systems Using the Simulink Design Documenter (SDD) Tool

Gordon W. Marks, Alexander Schaap, Gehan M. K. Selim,
Lucian M. Patcas

Guide and Reference Manual to Documenting Simulink Systems Using the SDD Tool

Gordon W. Marks, Alexander Schaap, Gehan M. K. Selim, and
Lucian M. Patcas

McMaster Centre for Software Certification (McSCert), McMaster University,
Hamilton, Ontario, Canada

Table of Contents

1	Introduction	4
1.1	How to Read this Document	4
1.2	Purpose of the SDD Tool	4
1.2.1	Importance of Documentation	4
1.2.2	Who should use an SDD document?	5
1.2.3	When should you start generating an SDD document?	5
1.2.4	Benefits of using the SDD Tool to create an SDD document	5
2	Installation	6
3	Quickstart	7
4	Structure of Generated SDD Document	26
4.1	Title Page	26
4.2	Document Details	26
4.3	Changelog	26
4.4	Table of Contents	26
4.5	Introduction	26
4.5.1	Document Purpose	26
4.5.2	Scope	27
4.6	Design Description	27
4.6.1	Purpose	27
4.6.2	Interface	27
4.6.3	Requirements Specification	28
4.6.4	Internal Design	28
4.7	Glossary	28
4.7.1	Acronyms	28
4.7.2	Definitions	29
4.7.3	Notation	29
4.8	Appendix	29
4.9	Summary of Warnings	29
5	How to Generate SDD Documents with the Tool	30
5.1	Basic Process	30
5.2	Configurations	31
5.2.1	Create the configuration file	32
5.2.2	Setting configuration variables	32
5.2.3	Configuration Variables	32
5.3	Identify the Top-System	33
5.4	Identify the Significant Subsystems	34
5.5	Document the Significant Subsystems	34
5.5.1	Why document each of the Significant Subsystems?	34
5.5.2	What to document in the Significant Subsystems	34
5.5.3	How to document the Significant Subsystems	35
5.6	Document the Top-System	35
5.6.1	What to document in the the Top-System	35

5.6.2	How to document the Top-System	36
5.7	Detailing the Interface of a Simulink system	36
5.7.1	Block Type	36
5.7.2	Name	36
5.7.3	Units	36
5.7.4	Min & Max	36
5.7.5	Data Type	37
5.7.6	Description	37
5.8	Requirements Traceability	37
5.8.1	What is Requirements Traceability?	37
5.8.2	Why is Requirements Traceability useful?	37
5.8.3	How to Establish Requirements Traceability	38
5.9	General Information	38
5.9.1	Saving the General Information files	38
5.9.2	What sections are referred to as “General Information”?	39
5.10	SDD Appendix	39
5.10.1	Create the appendix tables file	40
5.11	Generation	40
5.12	Maintaining Documentation	41
5.12.1	What not to do	41
5.12.2	Likely maintenance cases	41
6	Appendix	43
6.1	Nomenclature	43
6.2	Syntax for text based content	43
6.3	The SDD Blocks Library	44
6.3.1	Accessing The SDD Blocks Library	44
6.3.2	Modifying an SDD Block	44
6.3.3	Benefits of using The SDD Blocks Library	44
6.3.4	Fallback to using The SDD Blocks Library	45
6.3.5	Block Parameters & Properties	45
6.4	Where to Save Files	45
6.5	Errors	46

Chapter 1

Introduction

The Simulink Design Documenter (SDD) Tool provides Simulink developers with a guided, semi-automated process to create Software Design Description (SDD) documents. In short, an SDD document is a document that describes the design and inner workings of a Simulink system. Thus, the SDD document of a Simulink system can be cross-checked with the original system requirements to validate that the system's design satisfies its requirements. Note that a single project can have one or more SDD documents, corresponding to the available (sub)systems.

This guide discusses the aims and capabilities of the SDD Tool, and provides a hands-on tutorial to demonstrate how to use the tool.

1.1 How to Read this Document

While this is a large document, please note that users are not expected to read it all. A reasonable strategy when using this guide might be to first familiarize yourself with what the SDD Tool is for (Section 1.2) as well as installing it (Chapter 2). Then, the user can go through Chapter 3 which provides a quick start on how to use the SDD Tool to generate a basic SDD document. Chapter 5 would be useful to users who want a more detailed description of how to use the tool and all its available features. For a detailed description on the output of the SDD Tool, the user can refer to Chapter 4.

1.2 Purpose of the SDD Tool

The purpose of the SDD Tool is to facilitate the production of useful SDD documents for software systems developed with Simulink. Our hope is that this tool will encourage Simulink developers to document their systems by automating mundane parts of the documentation process to reduce the effort needed to create documentation.

Please note that an SDD document is not the only document that should be produced in a project that uses Simulink. Among other documents, Software Requirements Specification (SRS) documents are also very important to a successful project and they should not be overlooked. However, the SDD Tool focuses only on SDD documents.

1.2.1 Importance of Documentation

Although producing documentation is widely seen as a tedious task, it is not possible to ignore it, as this leads to systems which are difficult to understand and maintain. In practice, projects become increasingly complex with time, making software maintenance a time-consuming task. The problem becomes more apparent when new developers join a project, since a large amount of time is spent in comprehending the inner workings of the project. With SDD documents, these issues will be mitigated and a lot of resources can be saved.

1.2.2 Who should use an SDD document?

Any stakeholder in a project may wish to examine its SDD documents. Developers read SDD documents to understand how to modify (parts of) the system. Certain managers should examine these documents to ensure that the software satisfies its original requirements. For open-source projects, end users may refer to it to learn more about the system and how to use it.

1.2.3 When should you start generating an SDD document?

Right away! The longer a developer waits before starting to document a project, the less appealing it will be to start, the more difficult it will be to remember specifics of the system, and the more likely it is that new developers may join the project as others leave. This last case is especially problematic since the rationale behind design decisions will be lost if the developers leave without documenting such justifications.

1.2.4 Benefits of using the SDD Tool to create an SDD document

The SDD Tool and its generated SDD document are beneficial to Simulink developers for the following reasons:

- The developer can build and document Simulink systems within the MATLAB/Simulink environment, without having to switch to additional documentation tools.
- The SDD Tool automates tedious tasks of the documentation process, e.g., formatting headings, adding system images, and documenting Simulink systems' interfaces.
- The generated SDD document has a well-defined structure. This ensures the inclusion of crucial sections in the document, and the consistency between the structure of documents generated from different Simulink systems.
- The SDD Tool and the structure of documents it generates clarify the expected content for each section of the SDD document. This prevents different developers from having different understandings of what each section should contain.
- The SDD document is easily reproduced if the original Simulink model (containing the [Top-System](#)) is available.

Chapter 2

Installation

To have obtained this user guide, it is assumed that the user already has the folder named “SimulinkDesignDocumenter”, which contains the files pertaining to the tool. Please ensure that all the files in this folder are on the MATLAB search path. To learn more about MATLAB’s search path, see The MathWorks’ documentation¹. For information about adding folders to the search path, see the `addpath` command documentation².

Beyond the SDD Tool itself, the following software needs to be installed on your machine for the tool to work:

- **MATLAB version R2016b.** MATLAB R2016b is the version in which the majority of testing and development was done. Other versions will have some differences in formatting, but are worth trying if R2016b is not available. That said, the version in which the Simulink models were saved should not matter.
- **Simulink and Simulink Report Generator.** These are normally downloaded with MATLAB. To ensure that they are available, run the `ver` command via the MATLAB command window and check that “Simulink” and “Simulink Report Generator” appear in the output.
- **The Signature Tool.** The Signature Tool is used to accurately identify interfaces of Simulink systems in the SDD Tool. This tool can be found on The MathWorks’ File Exchange³, which requires you to have a Mathworks account. Download and install the Signature Tool in one of the following ways.
 - (Recommended) Select “Download Toolbox”, then run the downloaded file with MATLAB and follow on-screen instructions.
 - Select “Download Zip”, then unzip the files and ensure that they are on the MATLAB search path. To learn more about MATLAB’s search path, see The MathWorks’ documentation⁴. For information about adding folders to the search path, see the `addpath` command documentation⁵.

¹ http://www.mathworks.com/help/matlab/matlab_env/what-is-the-matlab-search-path.html

² <http://www.mathworks.com/help/matlab/ref/addpath.html>

³ <https://www.mathworks.com/matlabcentral/fileexchange/49897-signature-tool>

⁴ http://www.mathworks.com/help/matlab/matlab_env/what-is-the-matlab-search-path.html

⁵ <http://www.mathworks.com/help/matlab/ref/addpath.html>

Chapter 3

Quickstart

This chapter will guide the reader through a simple example to create an [SDD](#) document for a Simulink system.

Locate the tool

Once downloaded, the location of the SDD Tool can be found using the following command `which('GenSDD')`. The command will indicate the location of the `GenSDD.m` file which is located within the folder where the SDD Tool is located. Throughout this chapter, we refer to files within this root folder using a `~/` at the beginning of these files' path. For example, the `GenSDD.m` file is found at `~/GenSDD.m`.

About this example

For the example we use in this chapter, the [Top-System](#) will be the `sldemo.househeat` system which comes with MATLAB. Figure 1 demonstrates the [Top-System](#). This system models the outdoor environment of a house, its thermal characteristics, and heating system. The user can find the model with the `which('sldemo.househeat')` command.

To keep the original model intact, save a copy of the model on your MATLAB search path and use that copy to try out the steps in this chapter. The files we used for this example can be found in the directory `~/Examples/sldemo.househeat_doc_ex`. Thus, this directory contains the final model after performing the steps in this example. Note that the model was renamed to `sldemo.househeat_doc_ex.slx` in order to avoid shadowing between the files, but this would need to be changed back to `sldemo.househeat.slx` to test it out again.

Open the model that contains the [Top-System](#) by entering the following command in the MATLAB command window:
`open_system('PathToMyModel')`, replacing `PathToMyModel` with the full path of the model you copied earlier.

Navigate to the desired save location

The “Current Folder” in MATLAB refers to the location where the generated SDD document will be saved, and can be identified and changed in several ways. For example, the desired path can be entered into the MATLAB address bar. Additionally, the folder can be located by navigating in the “Current Folder” pane in MATLAB. The “Current Folder” pane should appear similar to Figure 2 in MATLAB. Finally, one can use the `pwd` command in the MATLAB command line to identify the current folder and the `cd` command to change it.

Chapter 3. Quickstart

Fig. 1: The sldemo_househeat system used for demonstration in this chapter.

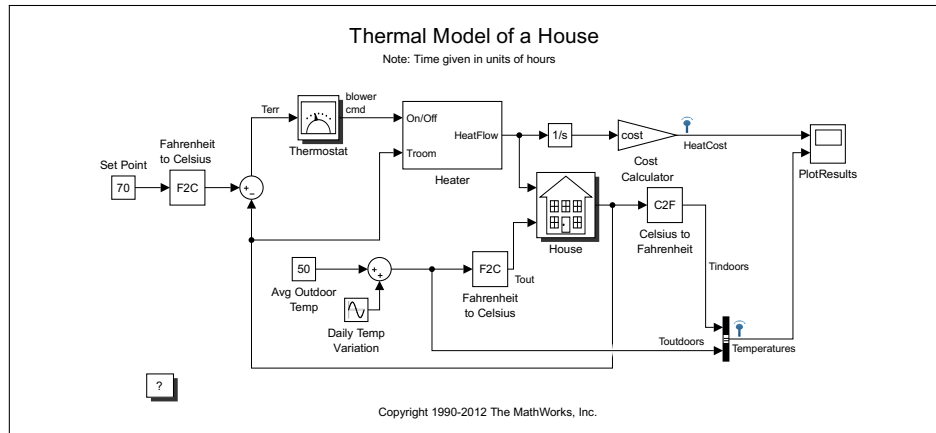
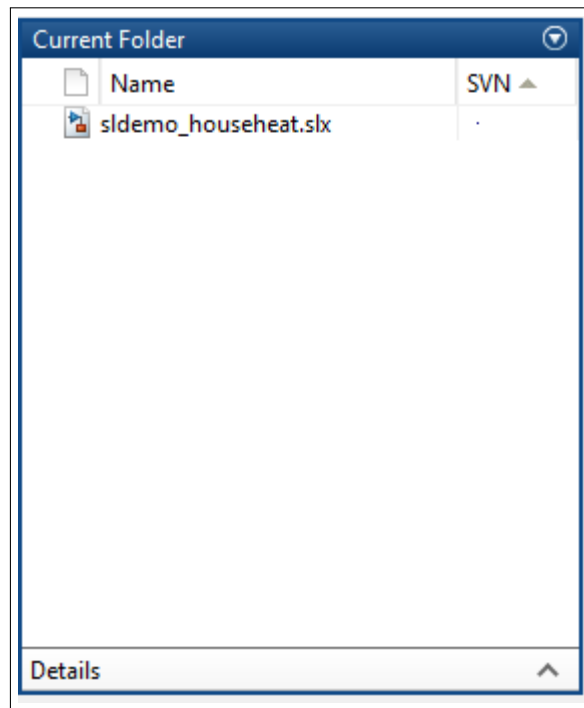


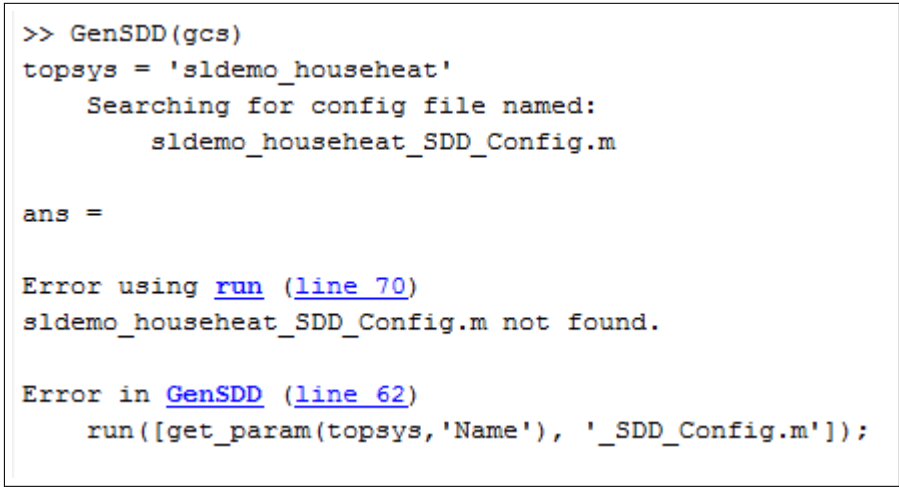
Fig. 2: The “Current Folder” pane as it appears in MATLAB. The pane size, contents, and style may look different depending on the settings, the current folder, and the MATLAB version.



Generate the SDD document

Next, we generate a preliminary SDD document to view the document's structure before we start documenting our example. This step helps the user understand what else needs to be done to create the SDD document, but is otherwise unnecessary. The SDD document can be generated by clicking in the “sldemo_househeat” Simulink system and then entering the following command in the MATLAB command line: `GenSDD(gcs)`. Note that `gcs` points to the current Simulink system or the system which was most recently clicked in. Thus, we click in our [Top-System](#), (i.e., “sldemo_househeat”) first, and then enter the previous command in the MATLAB command line. The MATLAB command line window should look similar to the snapshot shown in Figure 3 at the start of the SDD document generation.

Fig. 3: A snapshot of the initial command window output after generating a preliminary SDD document for our example.



```
>> GenSDD(gcs)
topsys = 'sldemo_househeat'
    Searching for config file named:
           sldemo_househeat_SDD_Config.m

ans =

Error using run (line 70)
sldemo_househeat_SDD_Config.m not found.

Error in GenSDD (line 62)
    run([get_param(topsys, 'Name'), '_SDD_Config.m']);
```

Additionally, a pop-up window similar to that shown in Figure 4 appears during the SDD document generation to indicate that there are warnings. The user should continue the SDD document generation process, unless the user already knows what the warnings are for and how to fix them.

Once generation has completed, two things occur. First, the generated files will appear in the current folder previously indicated by the `pwd` command. An updated view of the “Current Folder” pane should look similar to the snapshot shown in Figure 5. Second, the generated SDD document should have opened automatically. If the warnings pop-up window shown in Figure 4 appeared to the user, then a “Summary of Warnings” chapter will be added at the bottom of the generated SDD document. A snapshot of the chapter is shown in Figure 6. The “Summary of Warnings” chapter provides a list of the generated warnings, explaining their cause and solution.

Fig. 4: A snapshot of the pop-up which appears during the SDD document generation if there were warnings.

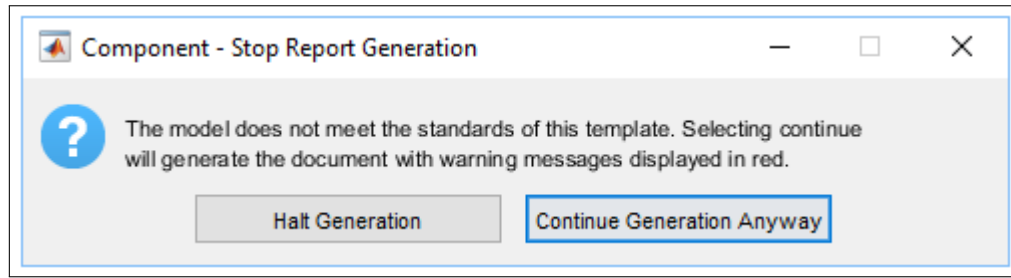


Fig. 5: An updated snapshot of the “Current Folder” pane after SDD document generation.

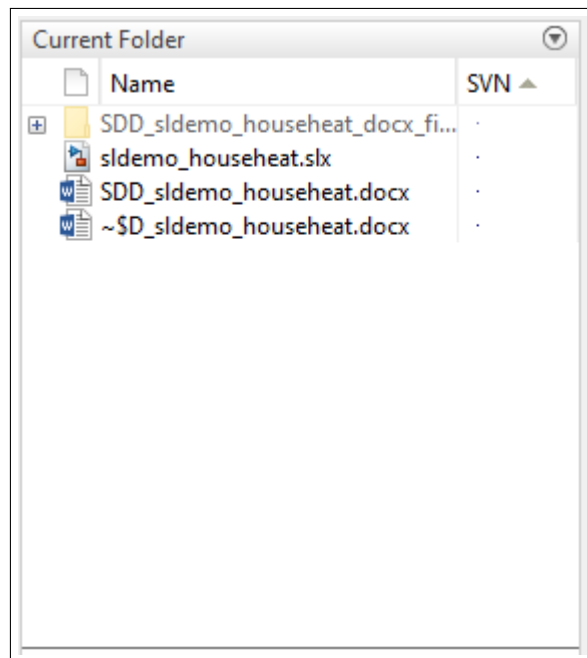
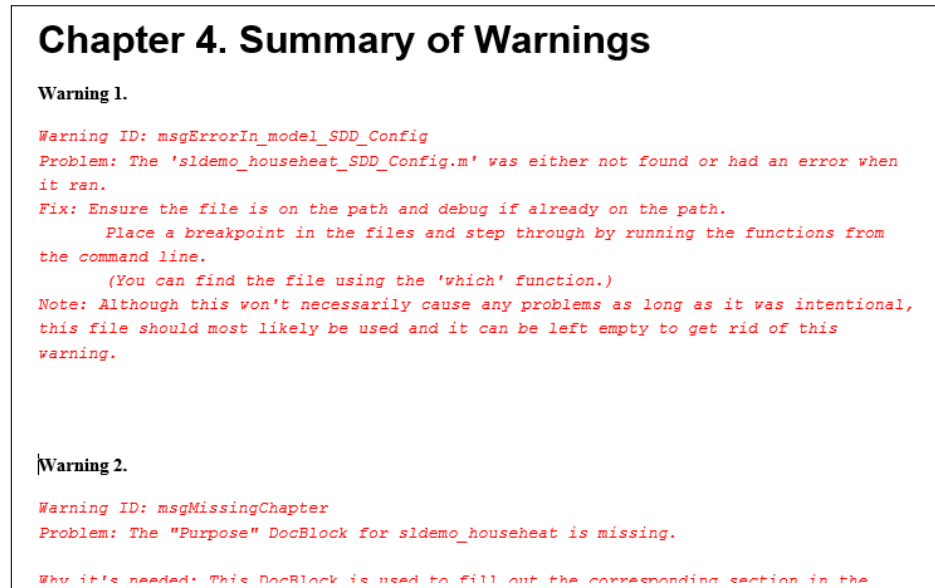


Fig. 6: A snapshot of the “Summary of Warnings” chapter in the generated SDD document.



Preparing the configuration file

The first warning shown in Figure 6 indicates that the first thing to be done is to create the “sldemo_househeat_SDD_Config.m” file. A template for this file can be found at: `~/Report_Specific.Files/TopsysName_SDD_Config.m`.

Copy this file to the location of your choosing and rename it to “sldemo_househeat_SDD_Config.m”.

As shown in Figure 7, we copied the file to the same folder containing the generated SDD document and the sldemo_househeat model that is used for this example.

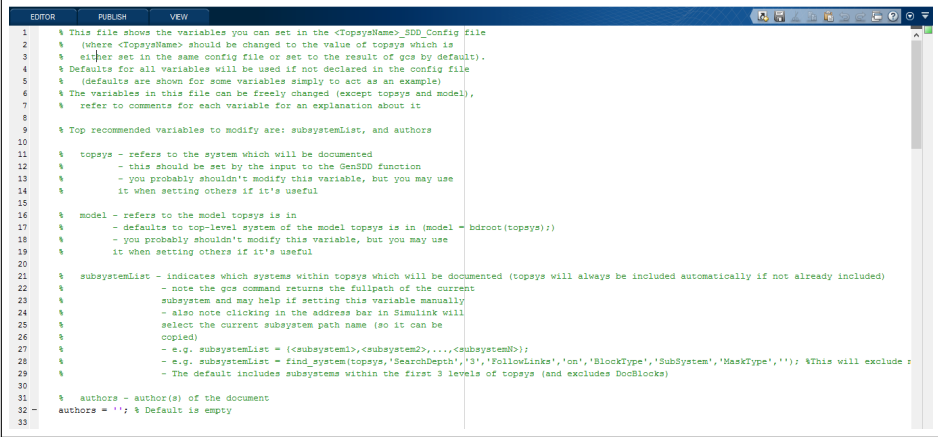
Fig. 7: A snapshot showing the files in the folder where the SDD document is saved. These files need not be saved together.

Name	Date modified	Type	Size
SDD_sldemo_househeat_docx_files	11/4/2016 1:55 AM	File folder	
SDD_sldemo_househeat.docx	11/4/2016 1:55 AM	Microsoft Word D...	125 KB
sldemo_househeat.slx	10/14/2016 2:00 PM	Simulink Model (S...	27 KB
sldemo_househeat_SDD_Config.m	10/24/2016 2:02 AM	MATLAB Code	9 KB

Chapter 3. Quickstart

Now, the config file needs to be modified. So, open the “sldemo.househeat_SDD_Config.m” file. Figure 8 shows a snapshot of the file’s contents.

Fig. 8: A snapshot of the initial configuration file.



```
1 % This file shows the variables you can set in the <TopologyName>_SDD_Config file
2 % (where <TopologyName> should be changed to the value of topsys which is
3 % either set in the same config file or set to the result of gcs by default).
4 % Defaults for all variables will be used if not declared in the config file
5 % (defaults are shown for some variables simply to act as an example)
6 % The variables in this file can be freely changed (except topsys and model),
7 % refer to comments for each variable for an explanation about it
8
9 % Top recommended variables to modify are: subsystemList, and authors
10
11 % topsys - refers to the system which will be documented
12 % - this should be set by the input to the GenSDD function
13 % - you probably shouldn't modify this variable, but you may use
14 % it when setting others if it's useful
15
16 % model - refers to the model topsys is in
17 % - defaults to top-level system of the model topsys is in (model = bdroot(topsys));
18 % - you probably shouldn't modify this variable, but you may use
19 % it when setting others if it's useful
20
21 % subsystemList - indicates which systems within topsys which will be documented (topsys will always be included automatically if not already included)
22 % - note the gcs command returns the fullpath of the current
23 % subsystem and may help if setting this variable manually
24 % - also note clicking in the address bar in Simulink will
25 % select the current subsystem path name (so it can be
26 % copied)
27 % - e.g. subsystemList = (<subsystem1>,<subsystem2>,...,<subsystemN>);
28 % - e.g. subsystemList = find_system(topsys,'SearchDepth','3','FollowLinks','on','BlockType','SubSystem','MaskType',''); %This will exclude
29 % - The Default includes subsystems within the first 3 levels of topsys (and excludes RootBlocks)
30
31 % authors - author(s) of the document
32 % authors = ''; % Default is empty
33
```

This file is heavily commented and explains how to modify the different configuration parameters in detail. For this example, we will modify the *subsystemList* variable and the *authors* variable. The *subsystemList* variable is the most important variable to set in this file. For the *subsystemList* variable, we need to decide which subsystems to document within the **Top-System**, and these systems will be referred to as the **Significant Subsystems**. For example, the user might just want to document the “House” subsystem in the **Top-System**. The name that needs to be used for the *subsystemList* variable, however, is not “House”. The easiest way to determine the full name of a subsystem is to navigate to that subsystem within Simulink and identify the name to use using either the `gcs` command or from the contents of the address bar in Simulink. Figure 9 shows these two methods for the House subsystem.

Knowing the names of the **Significant Subsystems** and the author(s), the *subsystemList* and the *authors* variables can now be set as shown in Figure 10.

Although the *subsystemList* variable only includes one subsystem, the `sldemo.househeat` system will be documented by default. For more complex systems, the user might find it useful to add more subsystems to the *subsystemList* variable.

To see how the document has changed, generate the SDD document again. The command window output should look similar to the snapshot shown in Figure 11. The notable difference is that the errors that appeared in Figure 3, are now absent.

Looking at the first page of the generated SDD document (shown in figure 12), one can see that the author indicated in the configuration file will now appear here.

Another notable change is in the structure of the overall SDD document. Previously, it had been set up by default to describe more systems. Currently, given that the *subsystemList* variable has been set, the structure was changed to only provide sections for the `sldemo.househeat` and the `sldemo.househeat/House`

Fig. 9: Two ways to get the current subsystem name. On the left, we retrieve the subsystem's name using the `gcs` command. On the right, we retrieve the subsystem's name from the Simulink address bar.

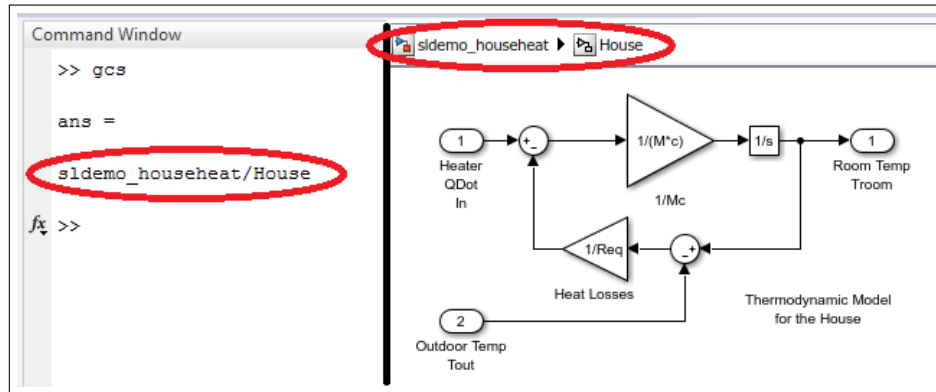


Fig. 10: Setting the `subsystemList` and the `authors` variables for our example.

```

21 % subsystemList - indicates which systems within tc
22 %               - note the gcs command returns the
23 %               subsystem and may help if setting
24 %               - also note clicking in the address
25 %               select the current subsystem path
26 %               copied)
27 %               - e.g. subsystemList = {<subsystem
28 %               - e.g. subsystemList = find_system
29 %               - The default includes subsystems
30 - subsystemList = {'sldemo_househeat/House'};
31
32 % authors - author(s) of the document
33 - authors = 'FirstName LastName'; % Default is empty
34

```

Chapter 3. Quickstart

Fig. 11: A snapshot of the initial command window output for the second SDD document generation in this example, and the warning pop-up.

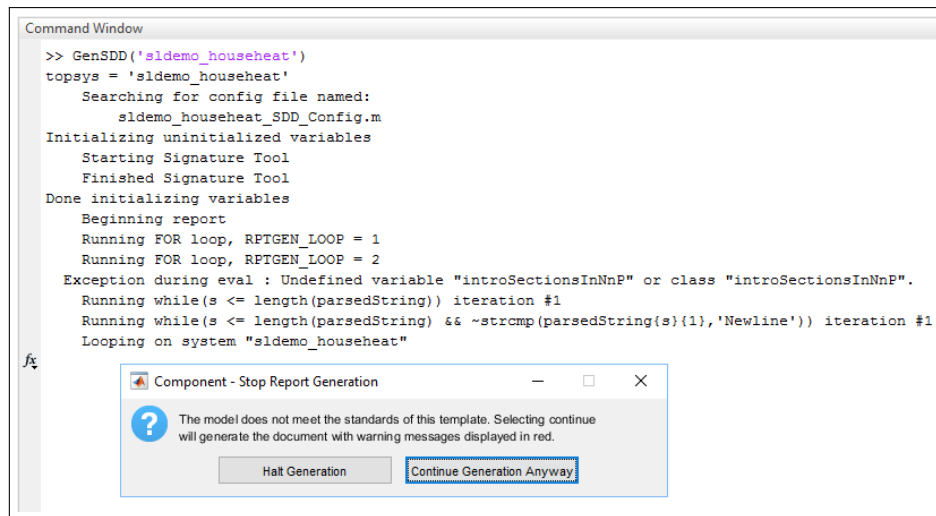
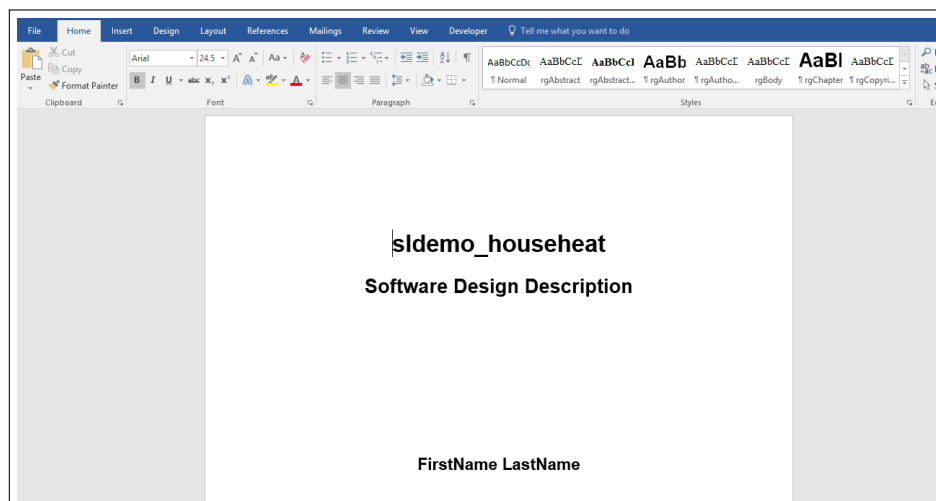


Fig. 12: A snapshot of the title page for the example with the author(s) set to 'FirstName LastName'.



systems. Figure 13 shows the Table of Contents of the newly generated SDD document.

Fig. 13: A snapshot of the table of contents of the updated SDD document.

Table of Contents	
Chapter 1. Introduction	0
1.1. Document Purpose	0
1.2. Scope	0
Chapter 2. Design Description of sldemo_househeat	1
2.1. Purpose	1
2.2. Interface	1
2.2.1. Inputs	1
2.2.2. Outputs	2
2.2.3. Declarations	2
2.2.4. Updates	2
2.3. sldemo_househeat Requirements Specification	3
2.4. Internal Design	3
2.4.1. Internal Design Supplement	3
2.4.1.1. Rationale	4
2.4.1.2. Anticipated Changes	4
2.4.1.3. Design Description of House	5
2.4.1.3.1. Purpose	5
2.4.1.3.2. Interface	5
2.4.1.3.2.1. Inputs	5
2.4.1.3.2.2. Outputs	6
2.4.1.3.2.3. Declarations	6
2.4.1.3.2.4. Updates	6
2.4.1.3.3. House Requirements Specification	7
2.4.1.3.4. Internal Design	7

Adding the required documentation

The “Summary of Warnings” chapter in the generated SDD document shows which warnings need to be addressed, and hence what additional steps need to be done to generate an informative SDD document. Figure 14 shows some of these warnings.

At this point, eight warnings remain, all of which share the same Warning ID, i.e., they all describe a similar problem with a similar fix. The following steps will be performed to fix each of these warnings.

First, open the Library Browser either using the `slLibraryBrowser` command at the command line or by clicking the appropriate button from Simulink, as shown in Figure 15. Note that the aforementioned button shown in Figure 15 may differ between MATLAB versions.

After the Library Browser opens, navigate in the left pane to SDD Blocks. Three sets of blocks appear, two of which provide optional blocks and one that provides blocks which are required to appropriately document the system. For this example, we are interested in the latter option which is labeled “Required SDD_Blocks For Documented (Sub)Systems”. Figure 16 shows where the desired option is within the Library Browser. By analyzing the warnings in the

Fig. 14: A snapshot of the “Summary of Warnings” chapter after the second SDD document generation.

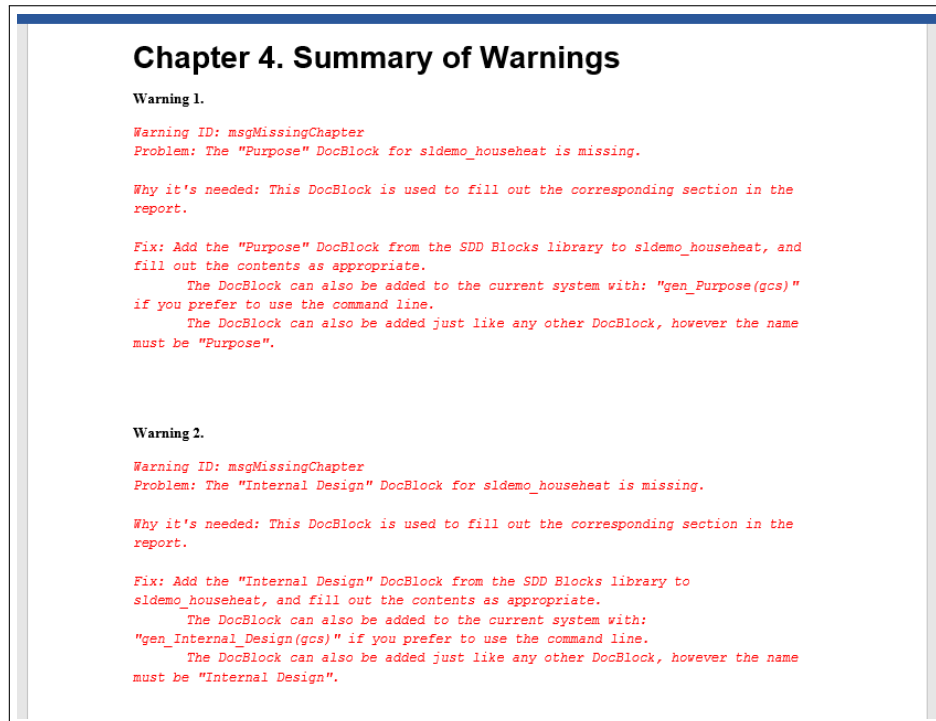
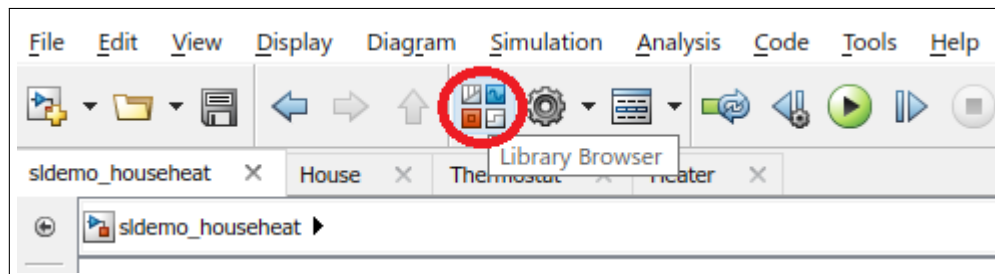
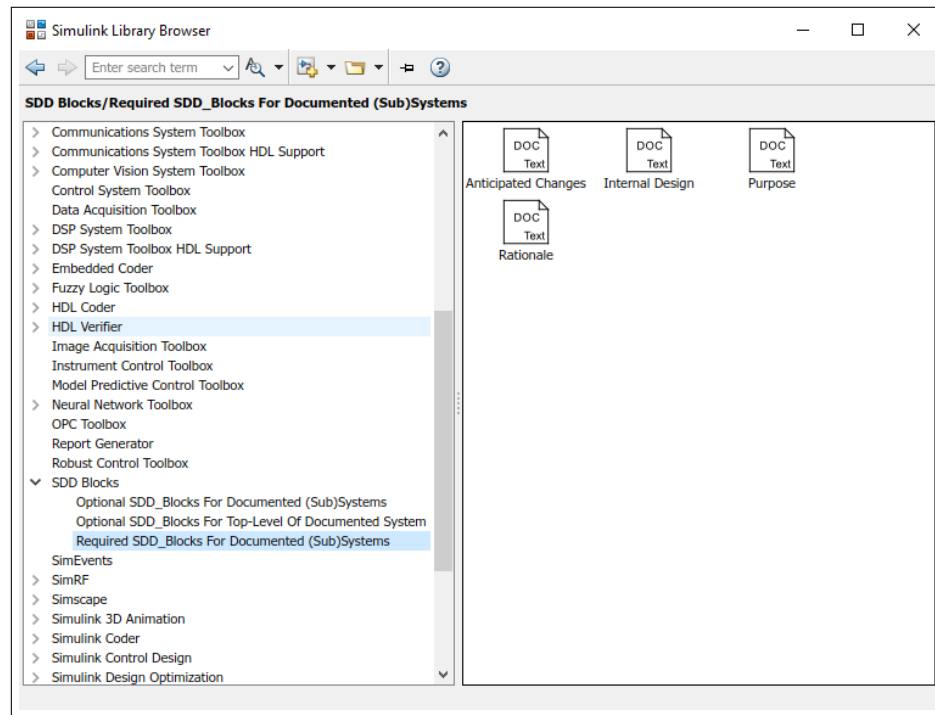


Fig. 15: A snapshot demonstrating the button to click to open the Library Browser in MATLAB R2016b.



generated SDD document, one can notice that the warnings stated that one of these blocks was missing from either the sldemo.househeat system or the sldemo.househeat/House system. Thus, our goal is to drag and drop these blocks within those systems. Figures 17 and 18 show the sldemo.househeat and the sldemo.househeat/House systems after adding the blocks grouped in the “Required SDD_Blocks For Documented (Sub)Systems” to them.

Fig.16: A snapshot of the Library Browser, showing the required blocks for generating SDD documents using our tool.



The SDD document can now be regenerated since the warnings were addressed. However, new warnings will be produced, as shown in Figure 19.

These warnings indicate that there is unmodified data in the blocks that were added. Such warnings are due to the fact that the SDD Tool checks to ensure that the user does not simply add these blocks without modifying their contents. Thus, to address these warnings, the user should modify the contents of the blocks by double-clicking the blocks and adding the necessary documentation to each block. For example, double-clicking the Purpose block from the sldemo.househeat system will open its contents to reveal text as shown in Figure 20.

Note that the default text in the SDD blocks consists of a number of lines preceded by a percent symbol. These lines are comments and thus, will not be included in the generated SDD document. These comments describe some features that are accessible through the text in the SDD blocks.

As for the documentation to add in the SDD blocks, the user should add any necessary information that they believe will sufficiently describe how the system

Chapter 3. Quickstart

Fig.17: A snapshot of the sldemo.househeat system after adding the required DocBlocks to it.

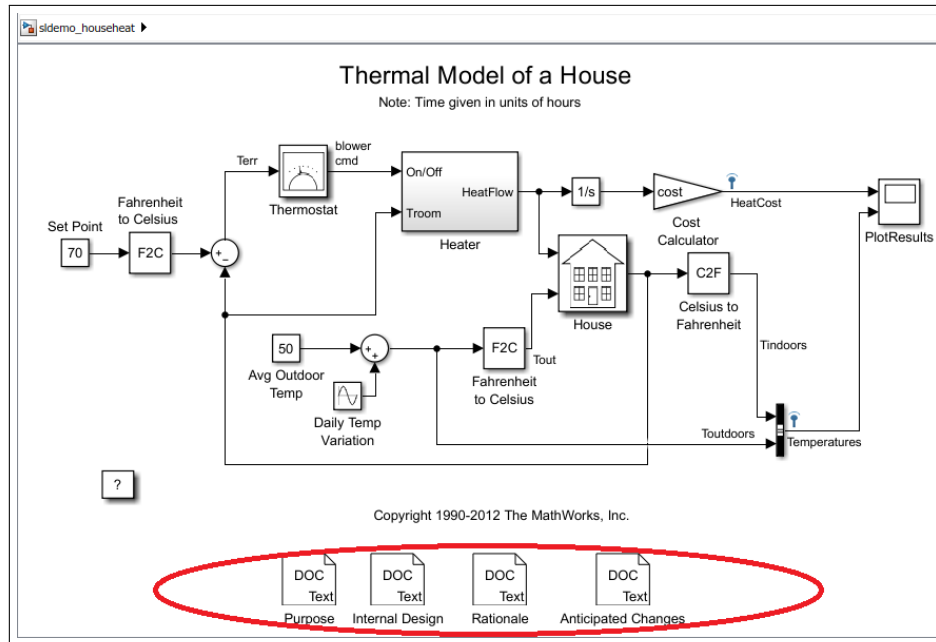


Fig.18: A snapshot of the House system after adding the required DocBlocks to it.

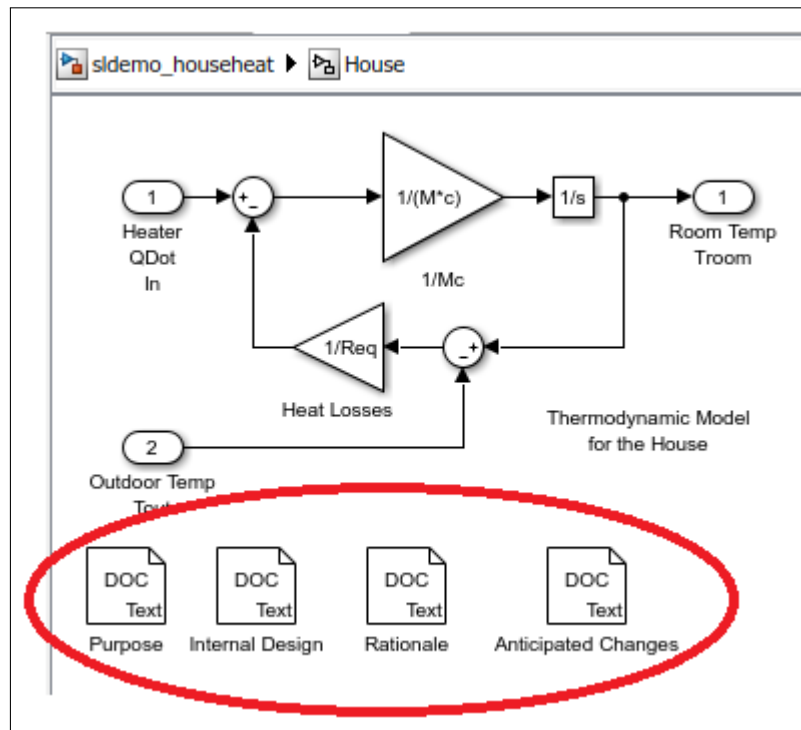


Fig. 19: A snapshot of the Summary of Warnings chapter after the third attempt at generating the SDD document for our example.

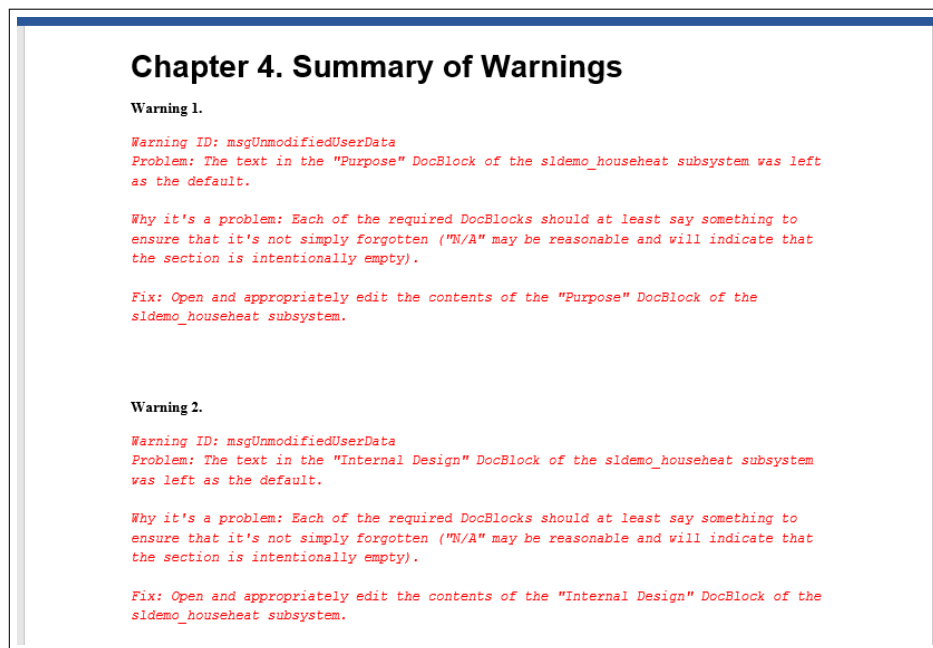
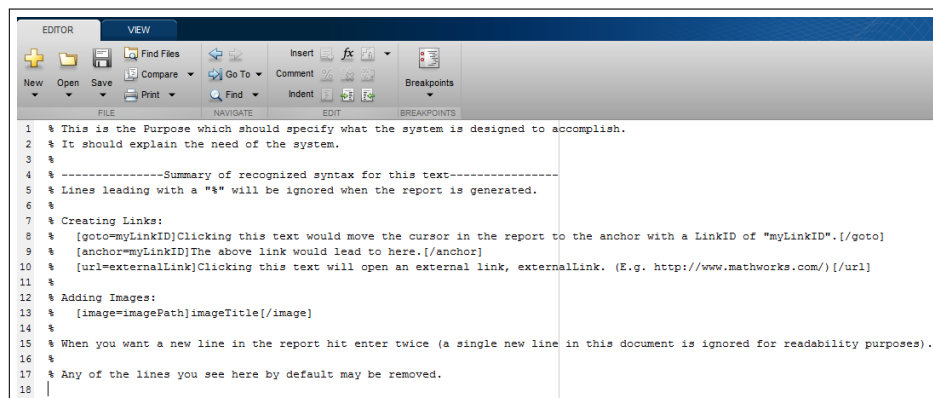


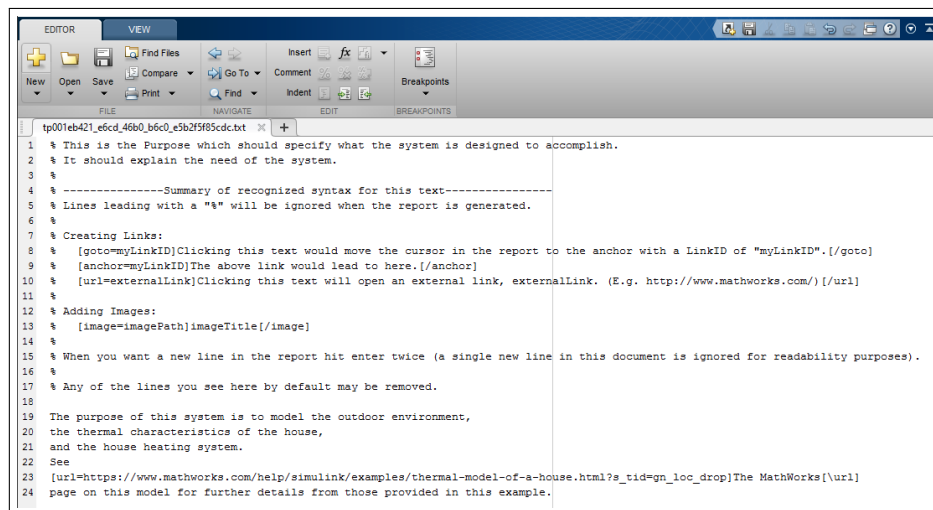
Fig. 20: A snapshot of the default contents of the Purpose blocks from the SDD Blocks library.



Chapter 3. Quickstart

works. The amount of information to add varies between systems, depending on their complexity. For the purpose of our example, we add text to the blocks based on information found on The MathWorks page about the `sldemo.househeat` model⁶. The text added to the Purpose block is shown in Figure 21. The user will notice that we have also created a link to The MathWorks page about the `sldemo.househeat` model within this block's text using tags defined in the comments. This file needs to be saved before closing it, for the block to be changed within the model. Saving this file will not save the model, too. Do not be concerned with the save location or what it is saved as, simply press `ctrl+S` before closing the file.

Fig. 21: A snapshot of the Purpose block after adding a short description of the system from The Mathworks.



Some people may be unhappy with the text interface previously shown to document the system, and prefer a format more similar to Microsoft Word. The blocks in the SDD Block library are made from DocBlocks, which can be formatted in different ways. To change the format for the Internal Design block for example, right-click the Internal Design block and select “Mask > Mask Parameters”, as shown in Figure 22.

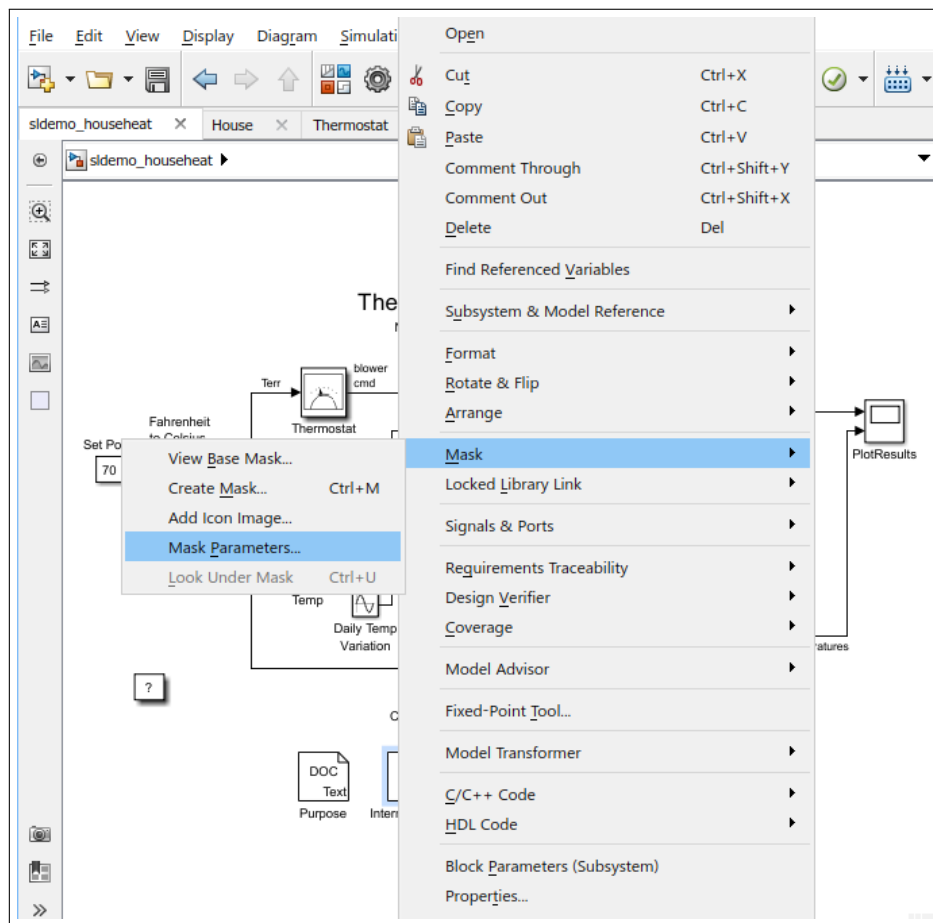
From the window that opens, select “RTF” (figure 23) from the drop down menu, then click “Apply”. The image of the block in the system now says “RTF” in place of “Text”, which was there previously. In general, DocBlocks in our tool always use either the Text or RTF format.

Now double-click the Internal Design block to see its contents (shown in Figure 24).

The contents which the Internal Design block would have had in its Text format are now in this RTF file. However, unlike the text file, the percent symbol no longer indicates content to be ignored, i.e., any content in the RTF file will

⁶ https://www.mathworks.com/help/simulink/examples/thermal-model-of-a-house.html?s_tid=gn_loc_drop

Fig. 22: A snapshot showing how to open the mask parameters of a block in MATLAB R2016b.



Chapter 3. Quickstart

Fig. 23: A snapshot of the “Block Parameters” window showing the “Document type” dropdown.

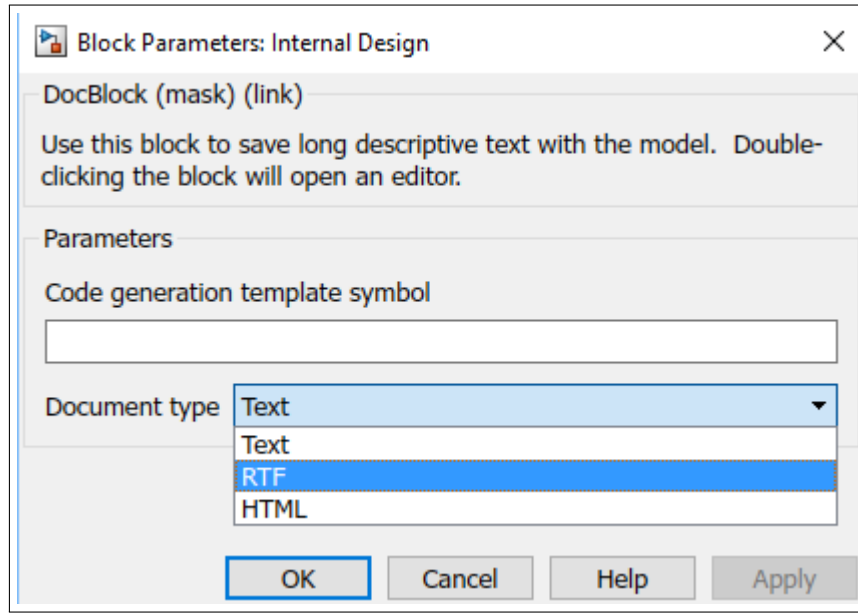
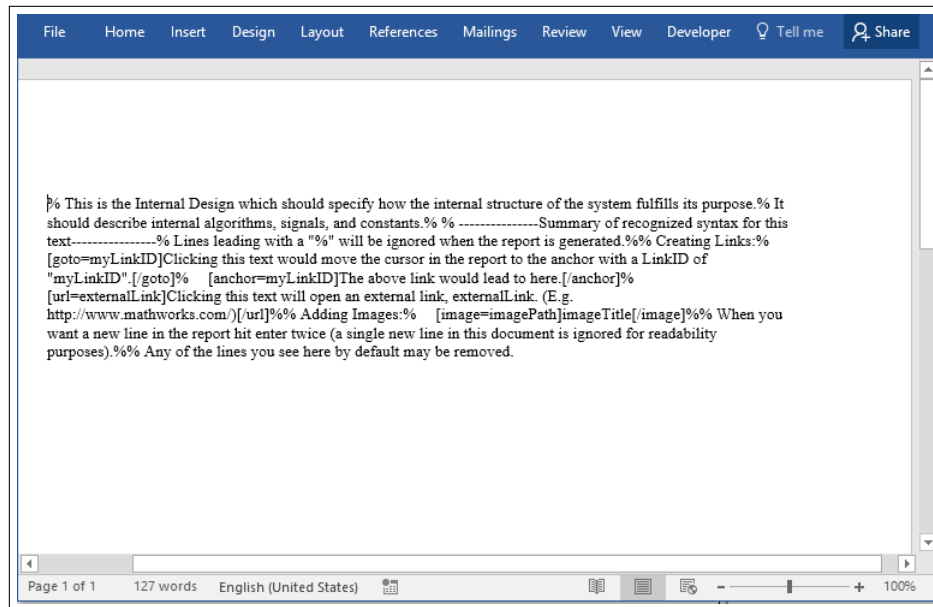
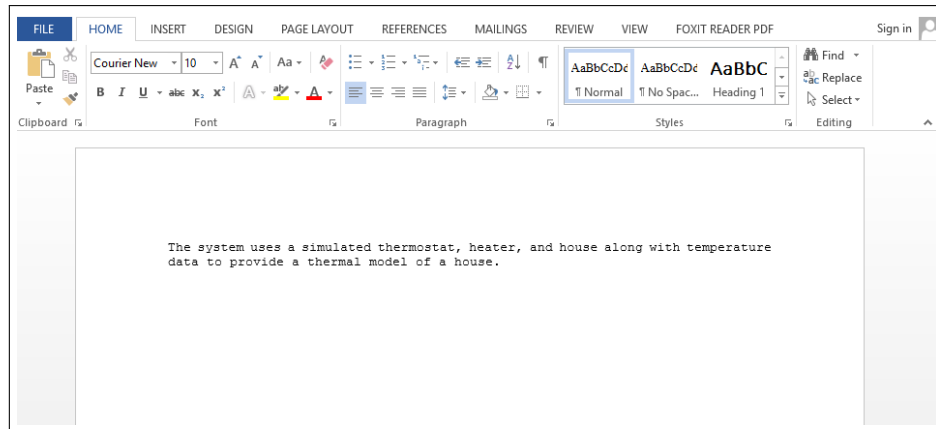


Fig. 24: A snapshot of the default contents of an Internal Design block from the SDD Blocks library if its format is changed to RTF.



appear in the generated SDD document. Additionally, any instructions following the “Summary of recognized syntax for this text” shown in Figure 24, no longer apply since that syntax is intended for text files. A description of the internal design can be added to the file, as shown in Figure 25.

Fig. 25: A snapshot of the Internal Design block after removing the default text and adding an appropriate description.



Save the contents of the file before closing. Now, continue to fill out the content of the remaining six blocks from the SDD Blocks library. Figures 26 and 27 show the remaining six SDD blocks we created for this example. Note that the Rationale blocks have no descriptive text since we were not the developers of this model and we could not find any documentation for it.

After adding documentation to all of the SDD blocks, the user can generate the SDD document again. There should no longer be a pop-up window during the document generation, and the “Summary of Warnings” chapter should no longer appear in the SDD document. This indicates that your SDD document is complete and has all the required information! The document we generated at this stage can be viewed at:

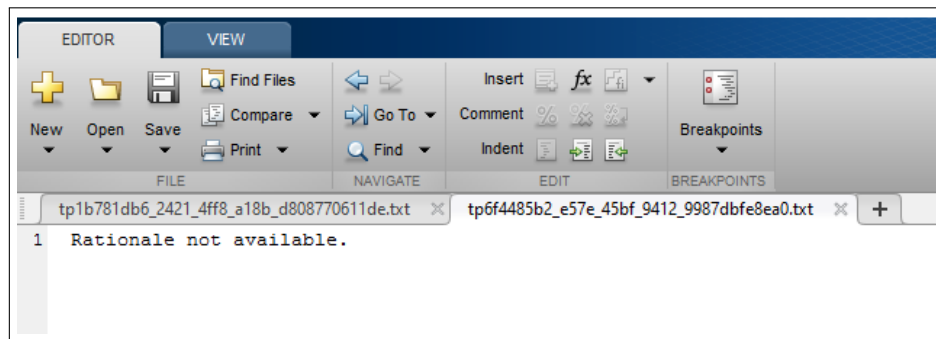
“~/Examples/sldemo.househeat_doc_ex/SDD_sldemo.househeat.docx”.

Chapter 5 of this user guide will describe in depth additional steps to prepare an SDD document using our tool, and will explain other features that were not discussed in this chapter.

Chapter 3. Quickstart

Fig. 26: Contents of the Rationale and Anticipated Changes SDD blocks for the sldemo.househeat system.

(a) Contents of the Rationale block for the sldemo.househeat system.



(b) Contents of the Anticipated Changes block for the sldemo.househeat system.

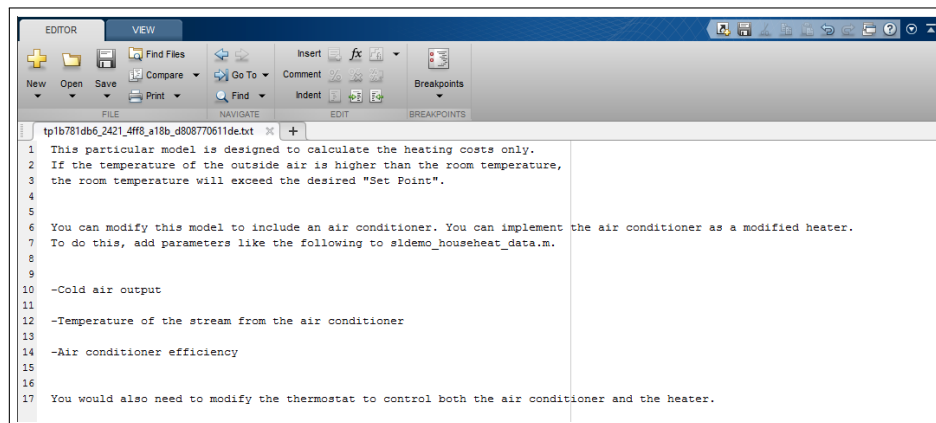
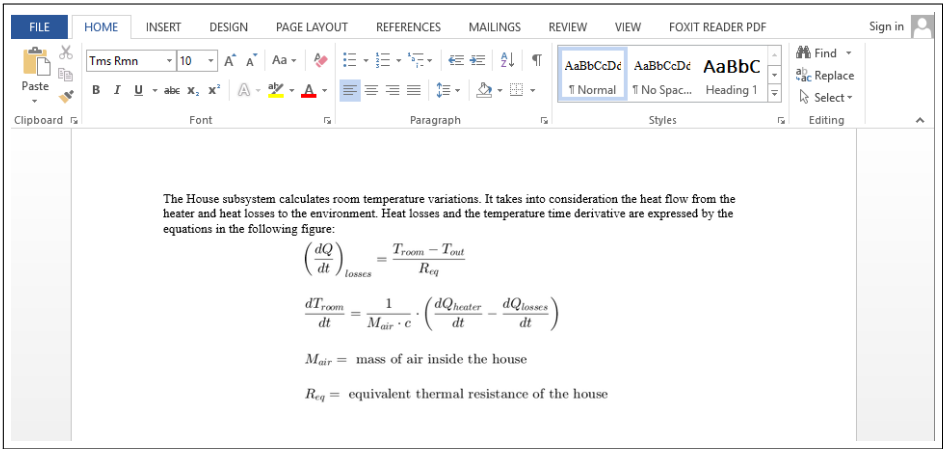
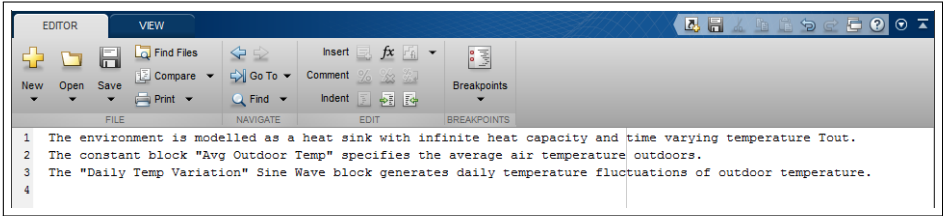


Fig. 27: Contents of all SDD blocks for the sldemo_househeat/House system.

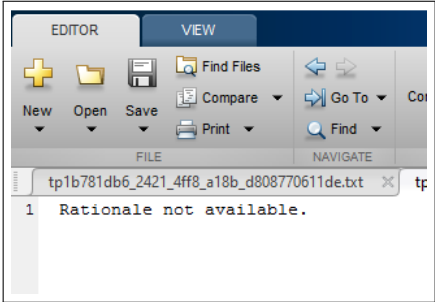
(a) Contents of the Purpose block for the sldemo_househeat/House system.



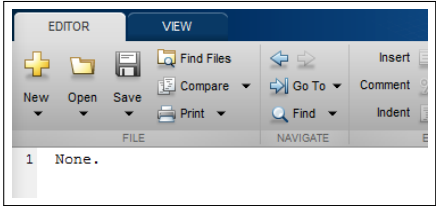
(b) Contents of the Internal Design block for the sldemo_househeat/House system.



(c) Contents of the Rationale block for the sldemo_househeat/House system.



(d) Contents of the Anticipated Changes block for the sldemo_househeat/House system.



Chapter 4

Structure of Generated SDD Document

This chapter describes the structure (and potential content) of the [SDD](#) document generated by the SDD Tool. Detailed steps describing how to generate this SDD document are discussed in [Chapter 5](#).

To facilitate understanding the structure of the generated SDD document, section names in this chapter closely correspond to section names in the generated SDD document, and have the same order as in the generated SDD document.

4.1 Title Page

The generated SDD document starts with a title page, that includes the document's title, subtitle, list of authors, and possibly an image (e.g. the logo of the company that generated the SDD document).

4.2 Document Details

This section provides information on how the document was generated, and may include an abstract and/or a legal notice.

4.3 Changelog

The Changelog section contains a table of the changes made to the system over time. For every change, the table includes the date of the change, the version of the file, a description of the changes made, the sections affected by the changes made, and the author of the changes.

4.4 Table of Contents

The Table of Contents contains a numbered list of the sections in the SDD document and their page numbers. Each section in the Table of Contents acts as a link that can be used to navigate to the corresponding section.

4.5 Introduction

This section introduces readers to SDD documents in general, and is not specific to the documented [Top-System](#). This section has two subsections: Document Purpose and Scope.

4.5.1 Document Purpose

This section explains the purpose of the SDD document.

4.5.2 Scope

This section elaborates on what should and should not be included in the generated SDD document.

4.6 Design Description

The Design Description section comprises the bulk of the SDD document and explains how the [Top-System](#) works. This section sufficiently describes the inner workings of the [Top-System](#) such that any user who is unfamiliar with it can easily figure out how it works, and understand how to modify the [Top-System](#) to implement new features or to address issues.

4.6.1 Purpose

This section describes *what* the given system is designed to accomplish, i.e., the function of the system.

4.6.2 Interface

In the context of Simulink, a system's interface is the set of blocks that act as either input or output to that system. While it is not difficult in Simulink to see that inports and outports act as inputs and outputs to systems, these are not the only blocks that can impact the system's functionality. Other blocks that can be in an interface include data store blocks and goto/from blocks. These blocks are capable of passing data into or out of a system without the need of a signal line connection.

The Interface section in the generated SDD document has four subsections, namely *Inputs*, *Outputs*, *Declarations*, and *Updates*. The Inputs subsection displays information on inports, data store reads, global froms, and scoped froms. The Outputs subsection displays information on outports, data store writes, global gotos, and scoped gotos. The Declarations subsection displays information on data store declarations and goto tag declarations, which are used to define the scope of some of the system's inputs and outputs. The Updates subsection will give different results depending on how the Signature Tool is called. If the Signature Tool is called in the default manner, then the Updates section will list data stores and goto tags that are both read from and written to.

Each of the four aforementioned subsections displays the blocks in the given system's interface in tables. The specifics about the blocks that go in each table are determined by the Signature Tool. Information given for each block includes:

Name An identifier for the block that is unique within a system.

Units E.g., m , m/s^2 , $N * m$.

Min The lower bound on the block's output value.

Max The upper bound on the block's output value.

Data type E.g., double, int8, boolean.

Description A description of the block.

4.6.3 Requirements Specification

This section specifies requirements that should be satisfied in the [Top-System](#). Ideally, this section will include tables with links, tracing from a component in the system to a corresponding section in an [SRS](#) document. This section will not be included for a given system if there are no written requirements associated with that system through the RMI provided by The MathWorks.

4.6.4 Internal Design

This section specifies *how* the internal structure of the system fulfills its purpose, by describing internal algorithms, signals, and constants. This section also contains a snapshot of the given system.

4.6.4.1 Internal Design Supplement

This section provides details on the given system's internal design, as outlined in the following subsections. At the top of this section is a table containing subsystems directly within the given system (i.e. subsystems for which it is the parent system).

4.6.4.1.1 *Rationale*

The rationale of a system justifies *why* certain design decisions were made. It should also justify why alternative design decisions (that were considered) were not selected.

4.6.4.1.2 *Anticipated Changes*

This section describes alterations that may need to be made to the system in the future. This information should help other developers implement designs that can be easily adapted to accommodate these changes.

4.6.4.1.3 *Nested Significant Subsystems*

The remaining subsections of the Internal Design Supplement (described in Section 4.6.4.1) are design descriptions for the [Significant Subsystems](#) nested within the [Top-System](#). In other words, the document recurses back to Section 4.6 to provide the same information for the [Significant Subsystems](#). These subsections are generated in a hierarchical manner that resembles the structure of the [Significant Subsystems](#) in the [Top-System](#).

4.7 Glossary

The Glossary of the generated SDD document contains acronyms, definitions, and notations that may help the reader understand the SDD document.

4.7.1 Acronyms

General Acronyms

This list explains any acronyms that will be useful for the reader of the SDD document. For example, it may include acronyms commonly used in industry.

System Specific Acronyms

This list explains acronyms specifically relevant to the [Top-System](#), that users of the [Top-System](#) may not be aware of.

4.7.2 Definitions

General Definitions

This list explains definitions that will be useful for the reader of the SDD document. For example, it may include jargon commonly used in industry.

System Specific Definitions

This list explains definitions specifically relevant to the [Top-System](#), that users of the [Top-System](#) may not be aware of.

4.7.3 Notation

General Notation

This list explains notations that will be useful for the reader of the SDD document. For example, it may include an explanation of the naming conventions used in the organization for which the SDD document is generated, or it may describe a graphical notation commonly used within the organization.

System Specific Notation

This list explains notations that will be used in the SDD document of the [Top-System](#). This may include an explanation of naming conventions used in the [Top-System](#) or it may refer to a graphical notation used in a diagram in the SDD document.

4.8 Appendix

The appendix may include tables of any desired information. For example, users may want tables listing variables that are used to calibrate the [Top-System](#), together with their data types and values.

4.9 Summary of Warnings

This section provides a summary of the warnings produced while generating the SDD document of the [Top-System](#). These warnings explain their cause and how to fix them. If no warnings were produced during the generation of the SDD document, then this section will be omitted.

Chapter 5

How to Generate SDD Documents with the Tool

This chapter describes how to generate the [SDD](#) document and populate each of its sections. Section [5.1](#) summarizes the steps required to generate the [SDD](#) document, and the remainder of this chapter explains these steps in detail.

5.1 Basic Process

After downloading and installing the software necessary to run the SDD Tool (Ch. [2](#)), you can start using the tool to generate an SDD document for a Simulink system. In this section, we summarize the steps needed to generate an SDD document. Each step is followed by a reference to another section that explains the step in detail. Each step is also prepended with a prefix to state whether the step is mandatory or optional, as per the legend shown in Table [1](#). The steps are given in a suggested order, but the order is usually not critical unless the steps have interdependencies:

Prefix	Meaning of Prefix
<i>opt</i>	This step is optional in producing an SDD document
<i>sug</i>	This step is optional, but suggested in producing an SDD document
<i>n-sug</i>	This step is not suggested but may be desired
<i>app</i>	This step is suggested if applicable to the document
<i>1-time</i>	This step needs to be done one time for numerous SDD documents
<i>req</i>	This step is required in order to produce an SDD document with the SDD Tool

Table 1: Meanings of the prefixes used with the steps to generate an SDD document document (Section [5.1](#)).

1. *req* Decide which system will be the [Top-System](#) ([5.3](#)). In short, the top system is the (sub)system to be documented, together with its sub-systems (if any).
2. *sug* Create a configuration file ([5.2.1](#)).
3. *sug* Add DocBlocks to the [Top-System](#) ([5.6](#)) from the [SDD Blocks library](#).
 - (a) *sug* Add Changelog ([5.6](#)).
 - (b) *opt* Add System Specific Acronyms ([5.6](#)).
 - (c) *opt* Add System Specific Definitions ([5.6](#)).

- (d) *opt* Add System Specific Notation (5.6).
4. *sug* Set the subsystemList variable to list the [Significant Subsystems](#) in the configuration file (5.4). By default, the [Significant Subsystems](#) includes all subsystems within the first 3 levels of the [Top-System](#).
5. *req* Add DocBlocks to the [Significant Subsystems](#) (*sug* from the [SDD Blocks library](#)).
 - (a) *req* Add Purpose (5.5.2).
 - (b) *req* Add Internal Design (5.5.2).
 - (c) *req* Add Rationale (5.5.2).
 - (d) *req* Add Anticipated Changes (5.5.2).
 - (e) *opt* Add Requirements Specification (5.5.2).
6. *1-time* Create external files for [General Information](#).
 - (a) *opt* SDD_Document Purpose.txt (5.9.2).
 - (b) *opt* SDD_Scope.txt (5.9.2).
 - (c) *app* SDD_Acronyms.txt (5.9.2).
 - (d) *app* SDD_Definitions.txt (5.9.2).
 - (e) *app* SDD_Notation.txt (5.9.2).
 - (f) Save each of these files in the same folder ([where to save](#)). Optionally, set the folder with these files with the pathToIntroSections configuration variable ([setting configuration variables](#)). By default, the individual files will be searched for on the MATLAB search path.
7. *sug* Add details to the interface blocks (5.7).
 - *sug* Set min/max values (5.7.4) & description (5.7.6) for blocks in system interfaces (this is useful regardless of documentation).
 - *opt* Create a getUnit function (5.7.3).
8. *sug* Add [RMI](#) links for requirements traceability (5.8).
9. *sug* Set configuration variables (setting configurations: 5.2.2) (list of configurations are listed in Section 5.2.3). More details about setting the variables can be found as comments in the file “Report_Specific_Files/ Topsys-Name_SDD_Config.m”.
10. *opt* Create an appendix tables file (5.10.1)
 - (a) Create a copy of the TopsysName_AppTables.m file. If the [Top-System](#) is named “MySystem”, the file should be renamed to “MySystem_AppTables.m”.
 - (b) Read some of the comments in the file and modify the file as desired.
11. *req* Generate the [SDD](#) document (5.11)
 - (a) *req* Ensure the model containing the [Top-System](#) is loaded
 - (b) *req* Run GenSDD('MySystem') replacing “MySystem” with the full path name of the [Top-System](#). Tip: If the [Top-System](#) is the Simulink system that was most recently clicked in, then the gcs command results will correspond with the appropriate input to the GenSDD function. The file saves at the location indicated by the pwd command.

5.2 Configurations

The configuration file is a MATLAB script where the user can initialize configuration variables (described in Sections 5.2.1, 5.2.2, and 5.2.3). If the user does not initialize these variables, the SDD Tool sets them to default values.

5.2.1 Create the configuration file

To create a configuration file, follow these steps:

1. Create a copy of `TopsysName_SDD_Config.m` saved in the `Report_Specific_Files` folder of the SDD Tool's directory.
2. Save the copied file in the appropriate directory, as described in Section 6.4.
3. Rename the copied file such that “`TopsysName`” is replaced with the name of the actual [Top-System](#). For example, if the [Top-System](#) is named “`MySystem`”, the configuration file should be renamed to “`MySystem_SDD_Config.m`”.

5.2.2 Setting configuration variables

Most configuration variables are set to a *char* (based on the desired option), and will look like “`<config.variable> = 'value'`”. Information on how to set each variable is described in the `TopsysName_SDD_Config.m` file in the `Report_Specific_Files` folder of the SDD Tool's directory.

5.2.3 Configuration Variables

In what follows, we list and describe configuration variables that we suggest the user sets:

- *authors*: The names of the author(s) of the SDD document.
- *subsystemList*: The [Significant Subsystems](#) that the SDD document should document. This variable can be set as follows:

```
subsystemList = {'subsys1', 'subsys2', 'subsys3'};
```
- *srsPath*: A path to an SRS document corresponding to the [Top-System](#).
- *getUnit*: Function to find a unit associated with a given block. For example, to set this variable the user can create a function called “`myGetUnitFunction`”, and save it in the appropriate folder (as described in Section 6.4). This function should take a block as an input and return units of the block as a *char*. Then, the user can add the following line to the configuration file:

```
getUnit = @myGetUnitFunction;
```

We further list and describe additional configuration variables that the user may want to edit:

- *title*: The title of the document; the default is the [Top-System](#)'s name.
- *subtitle*: The subtitle of the document; the default is “Software Design Description”.
- *titleImage*: A path to an image to display on the title page.
- *pathToIntroSections*: A path to a directory containing the “[General Information](#)” sections. The default will search for the individual files on the MATLAB search path with the `which` command (note the `which` command may find the wrong file if the filename is not unique on the search path).
- *abstract*: Adds content for an abstract to the SDD document (this is handled by the document purpose section, but may still be desired).
- *legalNotice*: Adds a legal notice to the SDD document.

The following are variables that we do not suggest changing, but may be worth looking into:

- *allowMissingDocBlocks* : This variable controls whether or not warnings appear in the SDD document for required documentation that is missing from the [Significant Subsystems](#) in the system (i.e. the blocks in the required section of the SDD Blocks library: Purpose, Internal Design, Rationale, and/or Anticipated Changes). By default, this variable is set to `false` indicating that warnings will appear in the generated SDD document if the aforementioned content is missing or not found. We do not suggest changing this variable because this would reduce the likelihood of the developer remembering to add this content which is integral to the document. If the information for these blocks is never expected to be filled in, then the user may want to set this variable to `true` so the document will look nicer.
- *displayWarningSummary*: This variable controls whether or not the Summary of Warnings chapter will appear in the SDD document. By default, this variable is set to `true` indicating that the chapter will appear in the document if there are any warnings to document. We do not suggest changing this variable because this would make it more difficult for the user to figure out the warnings identified by the SDD Tool during the generation of the SDD document. If there is no intention to address the warnings, then the user may want to set this variable to `false` so the document will look nicer.
- *removeInterfaceCols*: This variable controls the columns which appear in the tables in the Interface sections of the generated SDD document. By default this variable is set to include all columns (i.e. Blocks/Name, Units, Min, Max, Type, and Descriptions). The user may want to modify this variable to remove empty columns. However, before removing empty columns, the user should refer to Section 5.7 and attempt to add missing information to the [Significant Subsystems](#) rather than simply removing the columns.
- *removeSubCols*: Similar to the *removeInterfaceCols* variable, this variable controls the columns which appear in the tables in the Internal Design sections of the generated SDD document. Unlike the *removeInterfaceCols* variable, by default this variable is set to include only the Block/Name and description columns as these tables list subsystems and the other columns are not very meaningful for subsystems. The user may want to remove the description column if it is empty and they do not intend to provide descriptions. Information can be added to these columns in the same way it is done for the columns in the interface sections (see Section 5.7).
- *signatures*: This variable determines how the SDD Tool will call the Signature Tool. By default, the Strong Signature is called, with updates included. The reason we do not suggest changing this variable is because it is set reasonably and it would require the developer to have an understanding of the Signature Tool to make changes. If you would still like to consider it please refer to the Signature Tool documentation as needed.

5.3 Identify the Top-System

The SDD Tool can be used to generate an SDD document for a [Top-System](#), which represents any Simulink system within a Simulink model. The user can specify the [Top-System](#) for which the SDD document will be generated by passing its name as an input to the [GenSDD](#) function.

5.4 Identify the Significant Subsystems

The developer should identify the [Significant Subsystems](#) of the [Top-System](#), and set the *subsystemList* variable in the configuration file (Section 5.2.2) to these [Significant Subsystems](#). Basically, the [Significant Subsystems](#) are subsystems that are crucial to understanding of the overall design of the [Top-System](#). The default value of the *subsystemList* variable is set to every system within three levels from the top of the [Top-System](#)'s hierarchy.

5.5 Document the Significant Subsystems

This section will describe the needed information in order to appropriately document the [Significant Subsystems](#).

5.5.1 Why document each of the Significant Subsystems?

To generate a truly useful SDD document, it is important to document more than just the [Top-System](#) because a large number of design decisions were likely made at a lower level and may be difficult to understand without explanation. We recommend that the user documents every subsystem within the [Top-System](#) with [DocBlocks](#) from the [SDD Blocks Library](#), even if it will not be included in the SDD document, this will make it easier for developers to figure out how the [Top-System](#) works if they are inspecting it.

5.5.2 What to document in the Significant Subsystems

The Purpose, Internal Design, Rationale, Anticipated Changes, and the Requirements Specification should be documented for the [Significant Subsystems](#). The SDD Tool requires each of the aforementioned sections to be documented with [DocBlocks](#), except for the Requirements Specification section (since an [SRS](#) document may not be available for the [Top-System](#)). Thus, the tool will generate warnings in the SDD document if any of these sections are missing. These warnings can be removed by setting the [allowMissingDocBlocks](#) configuration variable to `true` (Section 5.2.2).

Purpose The [DocBlock](#) for this section should describe the purpose of the corresponding system. For information on what a purpose should explain, see Section 4.6.1.

Internal Design The [DocBlock](#) for this section should describe the internal design of the corresponding system, and should not cover the subsections described in Section 4.6.4. For information on what an internal design should explain, see Section 4.6.4.

Rationale The [DocBlock](#) for this section should describe the rationale of the corresponding system. For information on what a rationale should explain, see Section 4.6.4.1.1.

Anticipated Changes The [DocBlock](#) for this section should describe the anticipated changes of the corresponding system. For information on what the anticipated changes should explain, see Section 4.6.4.1.2.

Requirements Specification (If Applicable) The [DocBlock](#) for this section should describe the requirements specification of the corresponding system. For information on what a requirements specification should explain, see Section 4.6.3. To include tables with the traceability links mentioned in Section 4.6.3, see Section 5.8.

5.5.3 How to document the [Significant Subsystems](#)

For every subsystem in the [Significant Subsystems](#), use the DocBlocks (described in Section 6.3) with names corresponding to the titles in Section 5.5.2 (i.e., “Purpose”, “Internal Design”, “Rationale”, “Anticipated Changes”, “Requirements Specification”).

If there is nothing to document for a required section, the user can write a filler note in the corresponding DocBlock (e.g., “N/A”). Alternatively, the user can (though they should not) set the *allowMissingDocBlocks* configuration variable to *true* (Section 5.2.2) to prevent the generation of warnings for these missing DocBlocks.

5.6 Document the [Top-System](#)

This section will describe the needed information in order to appropriately document the [Top-System](#).

5.6.1 What to document in the the [Top-System](#)

This section will describe what sections should specifically be documented in the [Top-System](#). These sections are not required; one *should* be done in almost all cases and is thus labeled as “Suggested”, while the others should also be done, but may simply not be applicable for a given SDD document if there is no content to include in it. Each section is described below, first stating the name of the section, followed by the degree to which it is recommended to be included, then followed by the description of the section itself:

Changelog (Suggested) The DocBlock for this section should provide a changelog for the [Top-System](#). For information on what a changelog should include, see Section 4.3. Although the Changelog in the SDD Blocks library comes with a default table with predefined columns, it is not necessary to record the Changelog with that table.

System Specific Acronyms (If Applicable) The DocBlock for this section should list acronyms for the [Top-System](#). For information on what acronyms to list, see Section 4.7.1. Acronyms which are relevant outside of the [Top-System](#), as well as within it, may be documented in an external file to be included in the SDD document. Details on this external file are explained in Section 5.9.

System Specific Definitions (If Applicable) The DocBlock for this section should list definitions for the [Top-System](#). This DocBlock should contain the same content as the System Specific Acronyms DocBlock, except that it is for definitions. For information on what definitions to list, see Section 4.7.2. Definitions which are relevant outside of the [Top-System](#), as well as within it, may be documented in an external file to be included in the SDD document. Details on this external file are explained in Section 5.9.

System Specific Notation (If Applicable) The DocBlock for this section should explain notation for the [Top-System](#). For information on what notation to explain, see Section 4.7.3. Notation which is relevant outside of the [Top-System](#), as well as within it, may be documented in an external file to be included in the SDD document. Details on this external file are explained in Section 5.9.

5.6.2 How to document the [Top-System](#)

Use the DocBlocks described in Section 6.3 with names corresponding to the titles in Section 5.6.1 (i.e., “Changelog”, “System Specific Acronyms”, “System Specific Definitions”, “System Specific Notation”).

5.7 Detailing the Interface of a Simulink system

The information to be added to the interface will vary depending on the available information in the [Top-System](#). In some cases, the relevant data will be available in the [Top-System](#) and thus, will be processed automatically. In other cases, the developer might consider that some data are not necessary to understand the system’s interface. For the latter case, it is worth noting that the developer can remove columns from the tables in the SDD document by customizing the `removeInterfaceCols` configuration variable (Section 5.2.2).

The paragraphs below describe what the SDD Tool will need to fill out the different information for the Interface section.

5.7.1 Block Type

The type of block will be determined automatically by the Signature Tool (e.g., import, output, data store read, data store write, etc.).

5.7.2 Name

The Simulink block name will always be handled automatically by the Signature Tool based on the name in the [Top-System](#).

5.7.3 Units

Because until recently Simulink lacked support for units, the units associated with the output signal of a block may be handled in different ways by different developers. Some developers may use an organization’s naming convention and include unit names in the names of signals, others may use the “unit” block parameter (Section 6.3.5) introduced in R2016a, while others have other ways to make explicit the units associated with signals. Thus, the SDD Tool expects the user to create a MATLAB function to fill the units. The function may be created with any name, and it should take a block name as input (e.g. the value of `gcb`) and return a char representing the unit for the output signal of the block. After creating the function, the user should set the `getUnit` variable in the configuration file (Section 5.2.2) as follows: `getUnit = @<insert function name>;`.

5.7.4 Min & Max

Ideally, the min and max values for the output of the block should have already been set within the [Top-System](#) as it is best practice to do so. The reason for that is that these values are used in automatic generation of test cases. This means that if the values are set, then automatic testing will not need to waste time with cases that will never occur.

In the case where they are already set in the system, no effort is required as the min and max values will already appear within the document for all blocks which have it set.

However, if the values are not set, the developer should set them for the blocks in the interface in order to include this information within the document. Setting the min and max values is a straight-forward process, they correspond with the “Minimum” and “Maximum” block parameters and can easily be set by following the steps in section 6.3.5.1. Note that not all of the interface blocks (e.g. goto/from blocks) will have min and max values.

5.7.5 Data Type

Data types are automatically determined with the help of the Signature Tool. To determine a data type, the Signature Tool has a function which checks a given block’s data type block parameter. For some blocks, the block parameter is set to “Inherit: auto” which means their types depend on the signal that was passed into them and cannot be inferred trivially. In such cases, the user may want to set the block’s data type block parameter manually. Information about setting this block parameter can be found in Section 6.3.5.1.

5.7.6 Description

The description property may be a useful one to set as it may be used to help others to understand the significance of a given block within the [Top-System](#). The description is a block property which can easily be set as described in Section 6.3.5.1.

5.8 Requirements Traceability

This section will describe what requirements traceability is, why it is useful, and how to achieve it in the SDD documents generated by this tool.

5.8.1 What is Requirements Traceability?

Requirements traceability refers to the ability to track design decisions back to requirements. In the context of this tool, requirements traceability is established through links between the generated SDD document and pre-existing requirements documentation.

5.8.2 Why is Requirements Traceability useful?

In general, establishing clear traceability between design and requirements helps to ensure that a design satisfies all requirements and is useful toward understanding the need for different system components.

5.8.3 How to Establish Requirements Traceability

The MathWorks provides the Requirements Management Interface (RMI) to facilitate creation of requirements traceability links between external requirements documentation and Simulink objects. Further, the SDD Tool will automatically include links, which correspond with the RMI links associated with objects within the [Significant Subsystems](#). These links will be included in tables within the “Requirements Specification” sections in the SDD document. Thus if the developer creates an RMI link for each of the major components in the system, then readers of the SDD document will be able to trace between design decisions and the project’s requirements. This, in turn, will also make it easier to check if each requirement has been met as a requirement with no associated RMI link is likely to not have been achieved by the system.

To learn to create RMI links we recommend viewing The MathWorks’s documentation on it at <http://www.mathworks.com/help/slvnv/ug/link-to-requirements-document-using-selection-based-linking.html>.

5.9 General Information

As described in the [Nomenclature](#) section, General Information refers to “SDD document content which is not specific to the [Top-System](#) (or its subsystems) and may be re-used for other SDD documents”.

This section will describe relevant information about the General Information files, including how to save them, what sections are considered “General Information”, and what information belongs in those sections.

5.9.1 Saving the [General Information](#) files

This section outlines information relevant to saving the [General Information](#) files. Note that the information for saving these sections is different from saving other SDD document generation files described in [Section 6.4](#).

The following list outlines information relevant to saving the files:

- Since these files are intended to be used across numerous SDD documents, it may be appropriate to save files associated with this section separately from the other files used to generate the SDD document.
- The files need to be on the MATLAB search path.
- File names should correspond with those listed in the following section ([5.9.2](#)); filenames begin with “SDD_”, are followed by the name of the SDD document section (“Document Purpose”, “Scope”, “Acronyms”, “Definitions”, or “Notation”), and end with the file extension (“.txt” or “.doc”).
- To avoid the possibility of files shadowing each other, you may set the `PathToIntroSections` variable in the configuration file ([5.2.2](#)) which indicates a directory in which these files are saved.
 - Alternatively, you can check which files will be used by default to ensure it is the files you want. For example, for the document purpose, enter the following commands:

```
* which('SDD.Document Purpose.doc')
* which('SDD.Document Purpose.txt')
```

 - * If the first command returns a path, then the file that it indicates will be used for the document purpose.
 - * If the first returns nothing while the second returns something, then the path indicated by the second will be used.

5.9.2 What sections are referred to as “General Information”?

Example files for each of the sections are in the “General_Information” directory where the SDD Tool is saved. The example text files also give an explanation about what they should contain. Each file may be either a text file saved with extension .txt, or an RTF file saved with extension .doc. If a .doc is found it will be given precedence to be used in the SDD document. Files are found using the MATLAB `which` command, but the containing folder can be set if the `pathToIntroSections` configuration variable is set ([setting configuration variables](#)).

The files are as follows:

Document Purpose

- Filename (RTF): “SDD_Document Purpose.doc”
- Filename (Text): “SDD_Document Purpose.txt”
- The default in the General_Information directory is reasonable and need not be changed
- This should explain purpose of the software design description document.

Scope

- Filename (RTF): “SDD_Scope.doc”
- Filename (Text): “SDD_Scope.txt”
- The default in the General_Information directory is reasonable and need not be changed.
- This should elaborate on the what is and is not to be included in the document.

Acronyms

- Filename (RTF): “SDD_Acronyms.doc”
- Filename (Text): “SDD_Acronyms.txt”
- This is not needed if there are no acronyms used in your company’s documents which readers may be unfamiliar with.
- This should list and explain any acronyms, that are likely to be useful for the reader of the SDD documents to know.

Definitions

- Filename (RTF): “SDD_Definitions.doc”
- Filename (Text): “SDD_Definitions.txt”
- This is not needed if there are no definitions used in your company’s documents which readers may be unfamiliar with.
- This should list and explain any definitions, that are likely to be useful for the reader of the SDD documents to know.

Notation

- Filename (RTF): “SDD_Notation.doc”
- Filename (Text): “SDD_Notation.txt”
- This is not needed if there is no notation used in your company’s documents which readers may be unfamiliar with.
- This should list and explain any notation, that is likely to be useful for the reader of the SDD documents to know.

5.10 SDD Appendix

The SDD document may include an appendix which is capable of including any number of tables with any entries you want. Details about how to use the file are in the `TopsysName_AppTables.m` file within the `Report_Specific_Files` directory.

Chapter 5. How to Generate SDD Documents with the Tool

5.10.1 Create the appendix tables file

To create the appendix tables file, please follow the following steps:

1. Create a copy of `TopsysName_AppTables.m` ([where to save it?](#))
 - Note: The `TopsysName_AppTables.m` file is in the `Report_Specific_Files` directory which is saved with the SDD Tool.
2. Rename the file so that “`TopsysName`” is replaced with the name of the system being documented (i.e. [Top-System](#)).
3. If the [Top-System](#) is named “`MySystem`”, the appendix tables file should be renamed to “`MySystem_AppTables.m`”.

5.11 Generation

Following is a list of steps as well as things to note with regards to generating the SDD document:

1. Load the model containing the [Top-System](#) (for more information about choosing the [Top-System](#), see section 5.3)
 - You can use the `bdisloaded('MyModel')`⁷ command to check if the right model is loaded (just replace `MyModel` as appropriate).
 - Among other methods to load the model, we recommend using `open_system('MyModel')`⁸ and then navigating to the [Top-System](#) (if you do this, then `GenSDD(gcs)` can be used in step 3)
2. The SDD document will be saved in the current MATLAB folder so change the current folder if desired (among other methods, this can be done with the `cd` command). The `pwd` command will indicate the current folder.
3. Generate the SDD document
 - Execute the `GenSDD('topsys')` command from the MATLAB command window where “`topsys`” is the full Simulink path to the [Top-System](#) (not your system’s full file path to the model).
 - E.g. `GenSDD(gcs)` if the current system (i.e. the most recently focused system that is loaded) is the [Top-System](#). The MathWorks provides more information about the `gcs` function which may be useful if it is confusing⁹.
4. If there is a pop-up during SDD document generation, it will likely inform you that there will be warnings in the SDD document. Select “Continue Generation Anyway” unless you would like to stop it to immediately fix something without looking at the warnings.
5. If the SDD document failed to generate, look at the MATLAB command window.
 - The window may indicate problems that occurred during SDD document generation.
6. If the SDD document is successfully generated, then it should open up automatically.
 - The file will be named by the MATLAB code:

```
['SDD-', get_param(gcs, 'Name') , '.docx']
```


(note: `gcs` should be replaced with whatever was used as input to the `GenSDD` function) which should result in something like “`SDD_<TopsysName>.docx`”.

⁷ <https://www.mathworks.com/help/simulink/slref/bdisloaded.html>

⁸ https://www.mathworks.com/help/simulink/slref/open_system.html

⁹ <https://www.mathworks.com/help/simulink/slref/gcs.html>

- As mentioned in step 2, the save folder will be determined by the MATLAB code `pwd` when SDD document generation begins.
 - If you are using MATLAB R2016a (or earlier) to generate the SDD document (not recommended), then save the generated file as a PDF. There is a bug from before MATLAB R2016b which causes the Table Of Contents and RTF DocBlocks to disappear from the original document after closing it, but the PDF will not have this issue.
7. Warnings within the SDD document should indicate anything that still needs to be done for an effective SDD document.
 8. An artefact will be left after generation in the form of a folder which can be safely deleted if desired. The full system path to the folder will be `[pwd, 'SDD_', gcs, '_docx_files']` (note: `gcs` should be replaced with whatever was used as input to the GenSDD function).

5.12 Maintaining Documentation

Maintaining the documentation should not be overly complicated. As changes are made to the system, update the work that was originally done for the documentation. The [Basic Process](#) section may be a useful reference to see if anything should be updated since it outlines every step involved in preparing an SDD document from scratch. That said there are still some things that may be useful to note. This section will discuss what not to do when updating the SDD document as well as some likely cases in which some maintenance would be required as well as explaining what would need to be done.

5.12.1 What not to do

Do not edit the generated document! Every time the file is generated, it will overwrite the previous thus those changes would be lost. Even if the file had been renamed and was thus not lost, regeneration will create a new file that does not have the changes of the other file and it would be tedious to remember what those changes were and to have to do them again thus it is best to not make any modifications to the generated document.

5.12.2 Likely maintenance cases

Below is a list of a few cases in which the documentation may require a degree of maintenance along with a description of what work is likely needed. For each case a the changelog should summarize the changes and a final step will be to regenerate the SDD document.

Top-System name changed: It will be necessary to rename the configuration file as well as the appendix file (if used). The `subsystemList` configuration variable will likely need to be updated as the full paths will have changed.

Design choices changed for one of the Significant Subsystems: The Internal Design as well as Rationale for that system will likely need to be updated.

New technical terms used: Either the general definitions or the system specific definitions should be updated to list the new term.

Chapter 5. How to Generate SDD Documents with the Tool

Requirements change: The RMI links will need to be fixed as appropriate. Various design changes may result as well; new subsystems may be introduced and may need to be added to the [Significant Subsystems](#) and thus need their own documentation.

Chapter 6

Appendix

6.1 Nomenclature

Below is a list of terms and acronyms used in this document along with a description of how they were used. Note that some of these terms were created specifically for use in this guide.

SDD: Software Design Description. The type of document the SDD Tool aims to produce. The aim of the document is to explain the design of a system.

SRS: Software Requirements Specification. The aim of the document is to explain what a system should accomplish.

SDD Tool: This is the name of the tool this user guide is about. The tool aims to produce [SDD](#) documents.

Simulink model: In this document, think of a Simulink model as a .mdl or .slx file.

Simulink system: In this document, think of a Simulink system as a collection of Simulink objects located at a certain path within a Simulink model. The `gcds` command will return the most recently focused Simulink system (as long as it is still loaded).

Top-System: This will refer to the Simulink system being documented by a given SDD document. It may be any Simulink system including the top-level system in a model or a subsystem within it. Notably, this is the highest system in the hierarchy of systems being documented; the highest level system in the [Significant Subsystems](#).

Significant Subsystems: This will refer to the list of Simulink subsystems which will be documented by a given SDD document. This also includes the [Top-System](#).

General Information: This will refer to SDD document content which is not specific to the [Top-System](#) (or its subsystems) and may be re-used for other SDD documents. This content goes near the top of the SDD document and serves to introduce the reader to the document.

DocBlock: A DocBlock is a standard block that comes with MATLAB. The block can be found in the Library Browser¹⁰ and allows the user to store documentation information within a model. Each DocBlock stores information in one of three formats, Text, RTF, or HTML.

RMI: Requirements Management Interface¹¹. Used to create links associating external documents with Simulink objects (among other things).

6.2 Syntax for text based content

When using this tool there are a number of instances in which plain text may be included in the SDD document. To expand on the user's capabilities with this

¹⁰ <http://www.mathworks.com/help/simulink/gui/use-the-library-browser.html?requestedDomain=www.mathworks.com>

¹¹ <http://www.mathworks.com/help/slvnv/requirements-traceability.html>

plain text we have devised a simple syntax which allows the user to include links and images to be included in the SDD document. This syntax is usable within any Text format [DocBlocks](#) which are included in the SDD document, as well as within Text files which store [General Information](#) such as Document Purpose or Acronyms (the list of these files is given in [5.9.2](#)). For a full explanation of the capabilities of this syntax please refer to the TextDocBlockSyntax.txt file which comes with the SDD Tool.

6.3 The SDD Blocks Library

When adding [DocBlocks](#) to any of the [Significant Subsystems](#), we recommend using the DocBlocks in the SDD Blocks library which comes with this tool.

Each DocBlock in this library has a different use which is described in [Ch. 5](#). Some of the blocks must be used in each of the [Significant Subsystems](#) and are also recommended to be used in other subsystems, some may optionally be used in these subsystems, and some are to be used only in the [Top-System](#).

6.3.1 Accessing The SDD Blocks Library

This library and the blocks in it can easily be found by opening the Simulink Library Browser footnote [10](#) and navigating to the “SDD Blocks” section.

The blocks are then contained within sub-libraries which are named to indicate when the blocks within should be used. To add a block into the SDD document, drag it from the library browser directly into the desired system.

If there are any issues with the block library, the same blocks can be found in the SDD.Blocks.slx file (tip: use `which('SDD.Blocks.slx')` to find this file). This will not work if the model was saved before version R2012a (when .slx was introduced).

6.3.2 Modifying an SDD Block

After adding a block, its contents can be modified by double-clicking it to open the document which should explain what sort of content should be included within it.

If you would like to change the DocBlock format, Text and RTF DocBlocks are supported and this format can be set by selecting a given DocBlock, right-clicking, selecting “Mask” then “Mask Parameters...” (depending on MATLAB version), and then choosing the desired DocBlock format from the “Document type” drop-down.

6.3.3 Benefits of using The SDD Blocks Library

The benefits of using this library are that:

- it can serve as a quick reminder for which DocBlocks are needed,
- the developer will not need to worry about giving the blocks the right name (which is useful because the DocBlocks require specific names for the SDD Tool),
- it will contain a brief description of what content the DocBlock should contain,
- and it will contain a brief description of the syntax which can be used in Text DocBlocks (for further information about this syntax refer to [6.2](#)).

6.3.4 Fallback to using The SDD Blocks Library

If use of this library is not desirable or possible, the blocks within it are ultimately regular DocBlocks with specific names so the library is not strictly needed. Simply create DocBlocks, change their names as appropriate, and refer to this document for details about what information to give in them (although one could guess what the contents should be, it is important to ensure all developers on a team have the same understanding of what the content should be to ensure consistency, as such it is important to refer to something before filling out the contents).

6.3.5 Block Parameters & Properties

Block parameters and properties refer to values and information associated with a given block which can be set by the user.

6.3.5.1 Opening & Editing Block Parameters or Properties

Follow the following steps in order to open and edit the Block Parameters/Properties:

1. Right-click a block and select the “Block Parameters” or “Properties...” option as appropriate.
 - For block parameters, simply double-clicking the given block will generally work as well.
2. Enter the desired value in the desired field.
 - The field may be found under different tabs although not all blocks have these tabs.
 - Different blocks have different fields; do not expect all blocks to have a certain field.
3. Click “OK” or “Apply”.

This information may also be input programmatically. For information about entering information programmatically, search the The MathWorks’s documentation.

6.4 Where to Save Files

Files used to generate this SDD document can generally be saved anywhere, as long as it is still found on the MATLAB search path. However, there are still some suggestions we make for it:

- Save them in a folder called SDD Tool found in the same directory as the file containing the [Top-System](#).
- In MATLAB, before generation, navigate to the folder containing created above (for example using `cd`, the GUI, or the address bar).
 - This will help to avoid shadowing.
 - Otherwise you can ensure that shadowing does not occur by entering the command: `which('<my file>')`, where “<my file>” is a file that may be shadowed. The result should indicate the path to the file you would like to use.

Note: To find out about where the generated files are saved see [Generation](#) step 2.

6.5 Errors

This is not a comprehensive list of errors nor of known errors; many are missing.

Error 1

```
>> GenSDD(gcs)
topsys = ''
Error using GenSDD (line 57)
Invalid Simulink object name:

Caused by:
    Error using GenSDD (line 57)
    Empty object name found.
```

If you are using `GenSDD(gcs)` to generate the SDD document, ensure that `gcs` evaluates to the [Top-System](#). In the case shown above, no model was open thus the `gcs` command did not evaluate as desired.

Error 2

```
Report complete
>> GenSDD('C:/sources/LEAP_SDD_Tool/Examples/DemoModel_2011b/DemoModel_2011b')
topsys = 'C:/sources/LEAP_SDD_Tool/Examples/DemoModel_2011b/DemoModel_2011b'
Error using GenSDD (line 57)
Invalid Simulink object name:
C:/sources/LEAP_SDD_Tool/Examples/DemoModel_2011b/DemoModel_2011b

Caused by:
    Error using GenSDD (line 57)
    No hierarchy is allowed when specifying a block by its SID.
```

In the example here, `GenSDD` was given the full path to a model on the user's computer system. `GenSDD` does not take the full path to a model on your computer system, it uses the full path within Simulink to a system within a given model (these paths are indicated in the address bar within Simulink).