Quality Testing

Testing Procedure

CONTENTS

[Software-in-the-loop testing 3](#_Toc107324438)

[Software-in-the-Loop Simulation 3](#_Toc107324439)

[Model Setup 3](#_Toc107324440)

[Coder Toolbox 4](#_Toc107324441)

[Generate S- Function 5](#_Toc107324442)

[S-Function Window 6](#_Toc107324443)

[Resources and additional links 7](#_Toc107324444)

# Software-in-the-loop testing

Software in Loop Testing (SIL) is supported by MATLAB coder which enables the user to verify the production-ready source code and compiler object code To simulate the software conveniently, it is usual to use a locally-hosted compiler to run the code on the laptop and then when the code has been satisfactorily tested, deploy the code on the target controller using a cross-compiler.

## Software-in-the-Loop Simulation

A software-in-the-loop (SIL) simulation compiles generated source code and executes the code as a separate process on your host computer. By comparing normal and SIL simulation results, you can test the numerical equivalence of your model and the generated code. During a SIL simulation, you can collect code coverage and execution-time metrics for the generated code.

With SIL simulation, you can verify the behavior of production source code on your host computer.

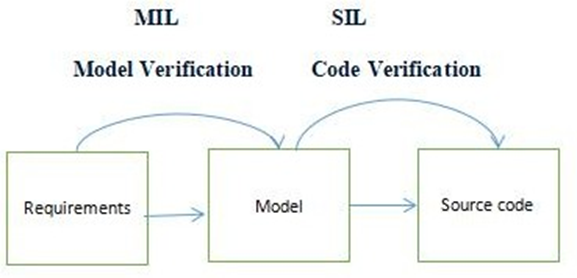


Figure 1. SIL Validation

### Software-in-the-loop Testing Procedure

SIL is to perform using S- Function builder so the testing is as follows Function block which contains header files (.c,.h) code which will be embedded and then it will act like a rapper to interface between the code, input, and output of the blocks and these are points to be considered below for the SIL process.

* Setup the embedded coder for code generation.
* Test Environment by S- Function Blocks.
* Create a reusable Test harness.
* Select the coverage setting if required.

# Model Setup

Setup the model to undergo SIL testing just go to **configure settings under the code generation tab and** check whether it is in **embedded coder (ert. tlc)** or change it.

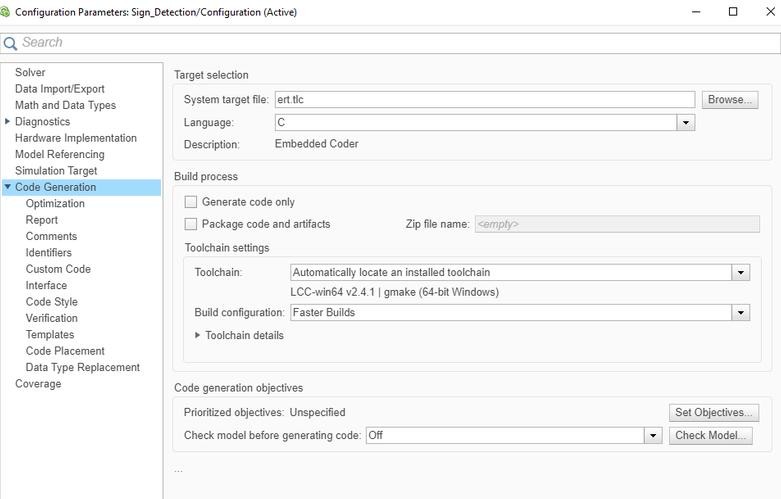


Figure 2. Configuration Parameters

## Coder Toolbox

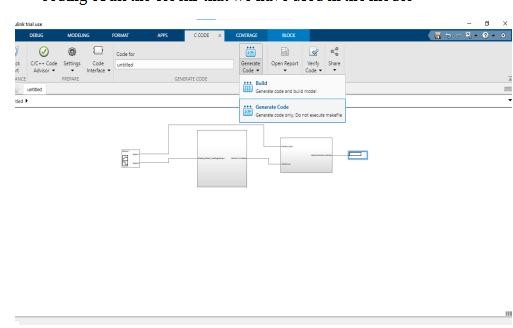
An embedded coder toolbox is part of the coursework and its available under (**APPS tab)** just move the group panes you will find Embedded Coder under the code generation option, select **Generate Code** and the purpose of this coder is readable, compact, and fast C and C++ code for embedded processors used in mass production.

Figure 3. Coder Toolbox

### C/C++ code

Once these instructions are implemented C/C++ code will be executed from the respective model.

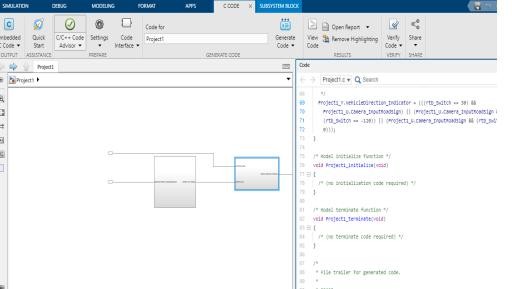


Figure 4. Code Generation

# Generate S- Function

For creating S-Function blocks from a subsystem, Right-click the subsystem and select **C/C++ Code** > **Generate S-Function**. The purpose of the S- function is to interface existing external code with the Simulink model and the generated code.

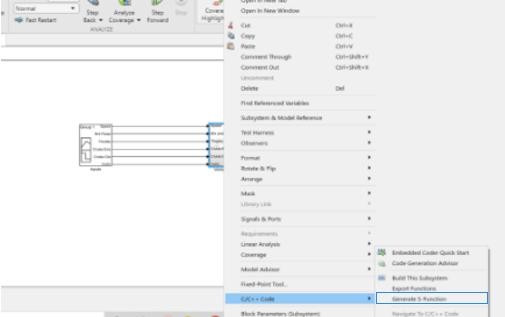


Figure 5. Generate S- Function Builder

## S-Function Window

In the **Generate S-Function** window, you see variables or data objects that are referenced as block parameters in the subsystem. You can declare them as tunable.

The upper pane of the window displays these columns:

1. **Variable Name**: Name of the parameter
2. **Class**: If the parameter is a workspace variable, its data type is displayed. If the parameter is a data object, its name and class are displayed.
3. **Tunable**: Let you select tunable parameters. To declare a parameter tunable, select the check box. In this example, the parameter K is declared tunable.
4. After selecting tunable parameters, click the **Build** button. The build process initiates code generation and compilation of the S-function by using the S-function target. The **Create New Model** option is automatically

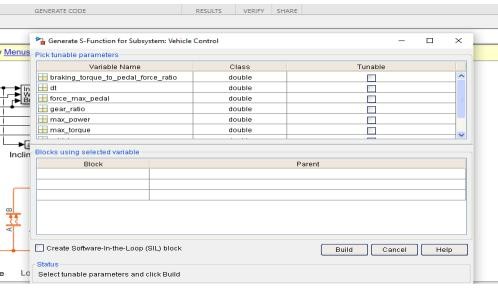


Figure 6 Variable Name

# Resources and additional links

1.Test Script- <https://www.mathworks.com/help/slrequirements/ug/linking-to-a-test-file.html>

2.Test Cases- https://www.mathworks.com/help/slrequirements/ug/link-test-case-to-requirements-documents.html