

# Introduction

**Wrong-way driver warning** is a new advanced driver-assistance system introduced in 2010 to prevent wrong-way driving. In the case of signs imposing access restrictions, through the wrong-way driver warning function an acoustic warning is emitted together with a visual warning in the instrument cluster – making an effective contribution towards helping to prevent serious accidents caused by wrong-way drivers.

Wrong-Way Driver Warning (WWDW) is based on road sign recognition & warning alert technology. The information about the car location provided by the on-board navigation system is also used. A forward camera behind the windshield recognizes important traffic signs (for example, speed limit, no parking etc.) along the car's route & transfers the same to the dashboard or projection display.

## What is Wrong Way Alert?

Wrong-Way Alert is the only wrong way driving solution designed to retrofit on existing signage. Wrong-Way Alert raises driver awareness in dangerous “wrong-way driver” situations. The system is affordable, scalable, and designed for quick retrofit installation. Wrong-Way Alert consists of *System Expandable* components which can be purchased separately as needed.

## Wrong-Way Alert *System Expansion & Upgrades*

The WWA system is a highly advanced Wrong Way warning system that can be logically designed to accommodate the various specific needs and challenges of each application. 4g Cell technology integrates for automatically alerting law enforcement or traffic safety personnel via SMS or email, that a wrong way incident is occurring, in real-time.

Highway on and off-ramps come in all shapes and sizes. Because of the extreme danger caused by inadvertent wrong-way drivers in these situations, an alert system has been needed which can easily be adapted to the particular roadway design. Our wireless controller and collaborator pairs have been designed to activate simultaneously and communicate with one another within milliseconds. Each controller or collaborator has a D2D range of up to 2000' in ideal conditions. The range can be extended via strategic placement since the units pass along network communications until they reach the intended units. This adaptability can allow the network to “see” around obstacles and terrain in complicated on or off-ramp situations. The mesh-net architecture allows for many configurations.

## System Expansion Options:

### **Back-Lit, Ring Enhanced Wrong Way and Do Not Enter Signs**

- Outstanding display performance pairs industry-standard reflective material with revolutionary internal illumination.
- Highly efficient.

- Proprietary illumination provides even, overall lighting of the sign to provide exceptional warning in any lighting scenario.

## **Components of the WWA System:**

### **High Intensity LED Enhancement Rings**

- Flush Mounted Enhancement Bars quickly mount on existing Wrong Way signage with permanent bonding tape and/or optional mechanical fasteners
- Sealed, potted unit
- Fully MUTCD compliant, sign shape is fully maintained even if several LEDs are physically damaged

### **Controller Unit**

- Optional detection of the wrong-way driver using doppler radar
- The unit can be located anywhere within the detection zone (dictated by configuration utility setup)
- Mesh-Net System – Allows immediate Collaboration with remote devices tied into the system
- Accepts 12V input from AC transformer or Solar supply (AC power highly recommended)
- Fast 4g Cellular Modem delivers full video notification in an average of 10 seconds
- HD Resolution IP Camera

### **Collaborator Unit**

- *Collaborates* detection of the wrong-way driver using synced doppler radars
- The unit can be located anywhere on-ramp within 4 detection zones (dictated by configuration utility setup)
- Mesh-Net System – Allows immediate Collaboration with remote devices tied into the system
- Solar or AC options available
- Flashing output configurable to several flash patterns and frequencies

## **Tools, Software and Keywords used**

MATLAB, Simulink, MBD - Model-Based Development, Test Coverages, Auto-Code Generation, Model Checker and Advisor, ISO 26262 safety, WWDW.

## **Aim**

To design a Simulink model for few of features of Wrong-Way Driver Warning System(WWDW). Details regarding the function of features are described in the given Requirement Document. Simulink Model includes Simulink model and Data Dictionary.

# Methodology

## Given -

Requirement document: Wrong-Way\_Driver\_Warning\_Detection\_Requirement.docx with 14 requirements guidelines from OEM.

**Step 1:** Studying all the parts of the requirement document provided, also called as requirement analysis.

Requirement Analysis is the first step in designing any model for a feature. Requirement sheet containing information regarding WWDW, function requirements and feature description is provided. Requirement sheet itself explains in detail about how the WWDW works.

## Requirement Document Analysis -

Requirement Analysis is the first step in designing any model for a feature. Requirement sheet containing information regarding WWDW features, functional requirements and feature description is provided. Requirement sheet itself explains in detail about how the WWDW works. Various key points can be drawn from the requirement sheet which is necessary for designing the model for the system. Overall component systems or subsystems included in WWDW is explained with the help of Function Architecture. It shows the flow of information from various sensors to the Driver

Information Display. Requirement Document also includes details regarding input/output signals to be used while designing the model and its parameters like data type, size etc. At the end of Requirement Document, function requirement of features to be designed are mentioned. There are 14 requirements which need to satisfy while designing the model.

Function Requirement 1 – Input Signal Processing 1

Function Requirement 2 – Input Signal Processing 2

Function Requirement 3 –Input Signal Processing 3

Function Requirement 3 –Input Signal Processing 3

Function Requirement 4 – Input Signal Processing 4

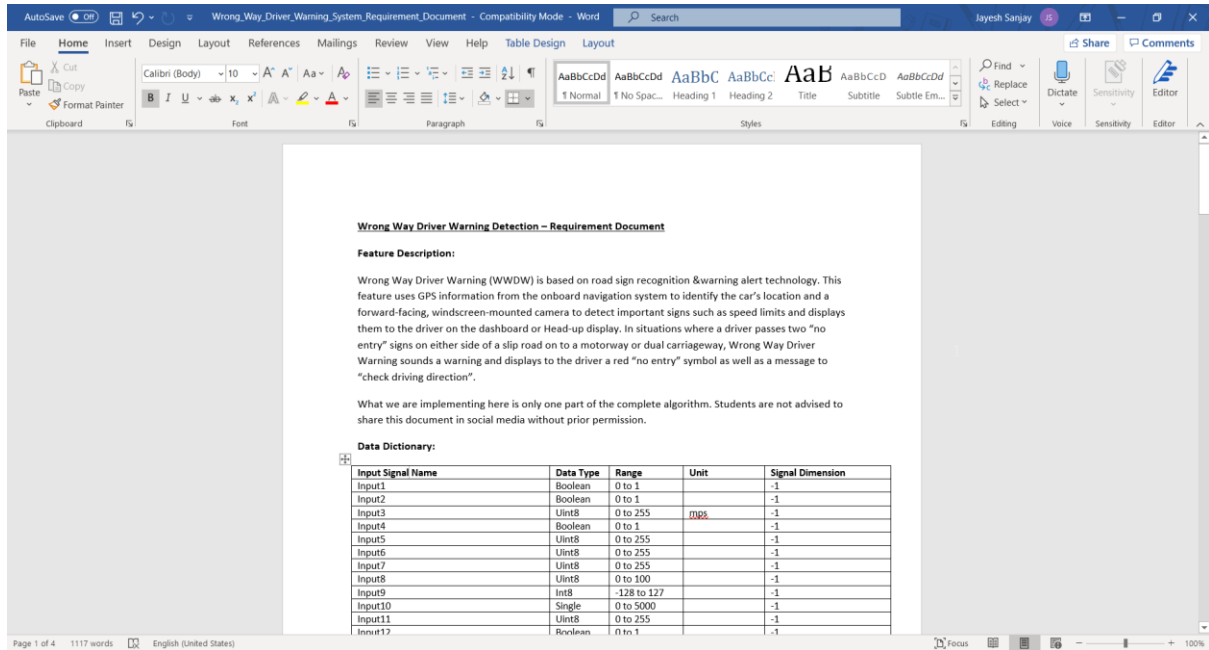
Function Requirement 5 – Input Signal Processing 5

Function Requirement 6 – Input Signal Processing 6

Function Requirement 7 –Input Signal Processing 7

Function Requirement 13-Input Signal Processing 8

## Function Requirement 14 -Input Signal Processing 9



**Wrong Way Driver Warning Detection - Requirement Document**

**Feature Description:**

Wrong Way Driver Warning (WWDW) is based on road sign recognition & warning alert technology. This feature uses GPS information from the onboard navigation system to identify the car's location and a forward-facing, windscreen-mounted camera to detect important signs such as speed limits and displays them to the driver on the dashboard or Head-up display. In situations where a driver passes two "no entry" signs on either side of a slip road on to a motorway or dual carriageway, Wrong Way Driver Warning sounds a warning and displays to the driver a red "no entry" symbol as well as a message to "check driving direction".

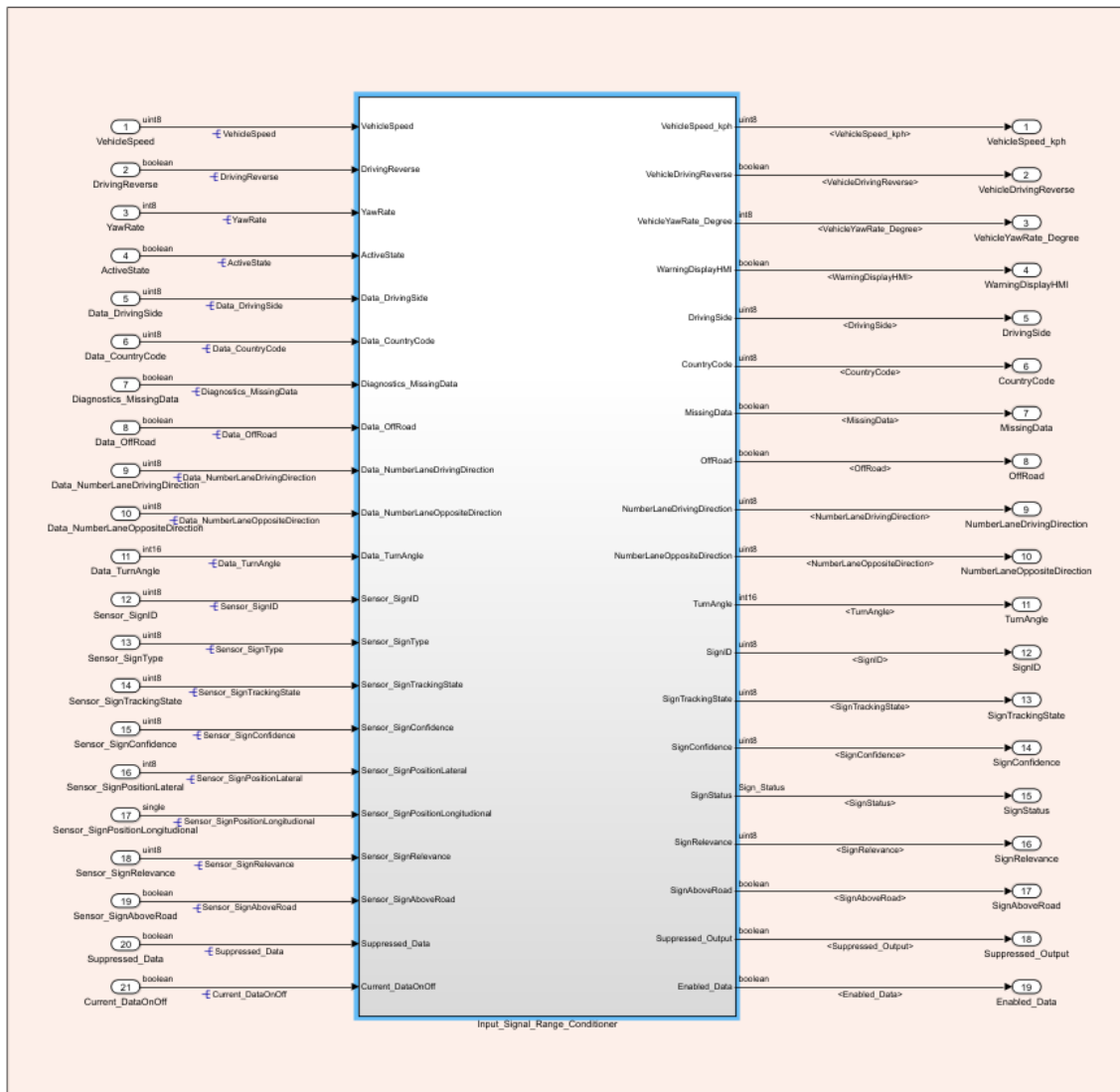
What we are implementing here is only one part of the complete algorithm. Students are not advised to share this document in social media without prior permission.

**Data Dictionary:**

Input Signal Name	Data Type	Range	Unit	Signal Dimension
Input1	Boolean	0 to 1		-1
Input2	Boolean	0 to 1		-1
Input3	UInt8	0 to 255	mps	-1
Input4	Boolean	0 to 1		-1
Input5	UInt8	0 to 255		-1
Input6	UInt8	0 to 255		-1
Input7	UInt8	0 to 255		-1
Input8	UInt8	0 to 100		-1
Input9	Int8	-128 to 127		-1
Input10	Single	0 to 5000		-1
Input11	UInt8	0 to 255		-1
Input12	Boolean	0 to 1		-1

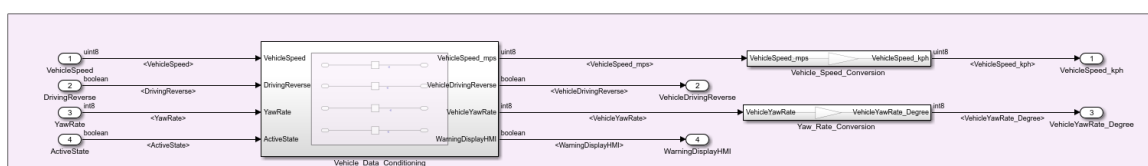
### Step 2: Modelling according to the requirement document

After studying all requirements of feature from the requirement sheet, the next step is to design the Simulink model based on these points. Firstly, a skeleton model is designed. It consists of a subsystem containing all inputs and outputs of the system.



Above is the skeleton model of the system. It consists of 21 inputs and 19 output signals. This Subsystem is named a Input\_Signal\_Range\_Conditioner. It consists of many other subsystems as per the requirement of features.

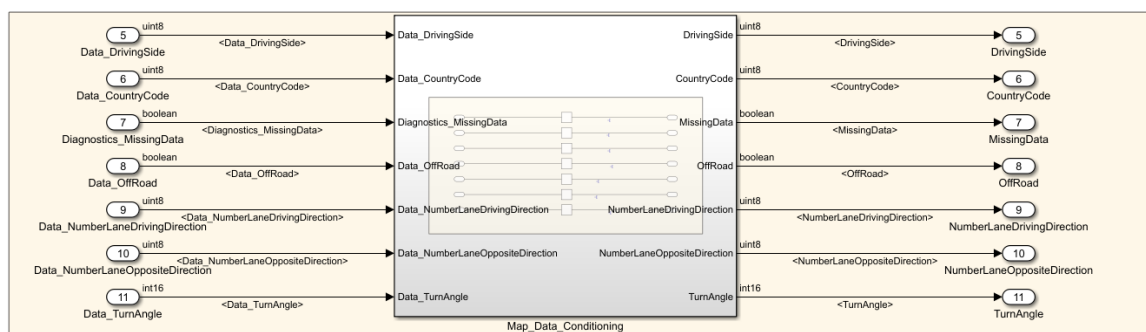
Inside this main subsystem, a view is below



There are 9 subsystems under the main subsystem. These subsystems are created as per function requirements.

For satisfying function requirements FUN – REQ – 00001/2/3 following 3 substems are created.

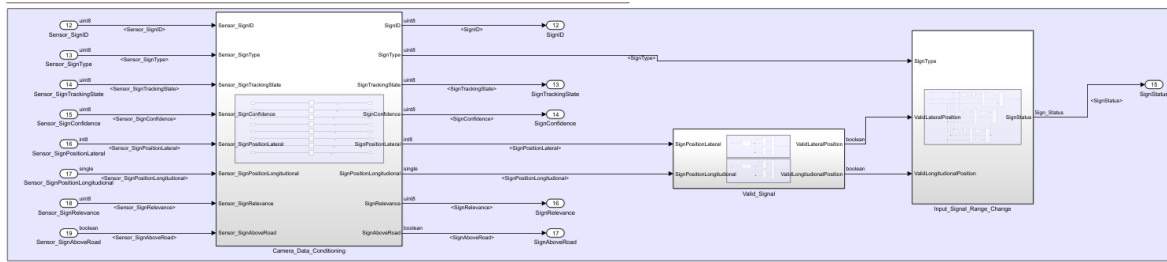
As per the first function requirement, signals: VehicleSpeed, DrivingReverse, YawRate, ActiveState needs to be renamed without any processing. This requirement is satisfied with subsystem Vehicle\_Data\_Conditioning. In between import and output, a slider gain block with value 1 is placed. Value 1 didn't interfere with the input signal but it facilitates renaming the signal as per requirement.



For function requirement 00002/3, vehicle speed needs to be converted from mps to kph and yaw rate from rad/s to deg/s. For this purpose, subsystems Vehicle\_Speed\_Conversion and Yaw\_Rate\_Conversion are created. Conversion is done with the help of Gain Block with values specified in the requirement document.

As per function requirement FUN – REQ – 00004 signals: Data\_DrivingSide, Data\_CountryCode, Diagnostics\_MissingData, Data\_OffRoad, Data\_NumberLaneDrivingDirection, Data\_NumberLaneOppositeDirection, Data\_TurnAngle need to be renamed without being undergone any change.

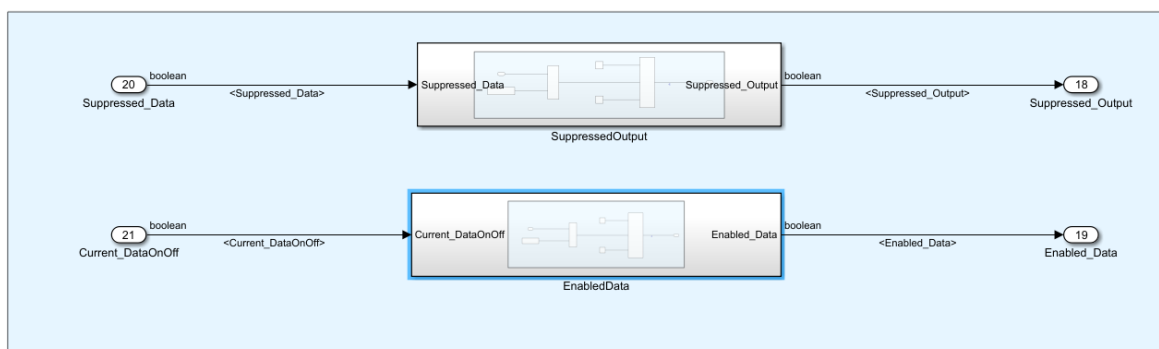
For this purpose, a subsystem named Map\_Data\_Conditioning is created. As per previous systems, in this system, a slider gain block with value 1 is used for renaming the signals from import to output.



As per requirement FUN – REQ – 00005, signals:Sensor\_SignID, Sensor\_SignType, Sensor\_SignConfidence, Sensor\_SignTrackingState, Sensor\_SignPositionLateral, Sensor\_SignPositionLongitudinal, Sensor\_SignRelevance, Sensor\_SignAboveRoad needs to be renamed without undergone any process. For this purpose, a subsystem named Camera\_Data\_Conditioning is created. Inside the subsystem, slider gain block with value 1 is used.

For function requirements FUN – REQ – 00006/7/8/9, two subsystems are developed named Valid\_Signal and Input\_Signal\_Range\_Change.

Subsystem Valid\_Signals receives its input from Camera\_Data Conditioning. This subsystem checks the validity of SignPositionLateral, SignPositionLongitudinal. As per requirement document Traffic sign, the lateral position is valid within -60 to +60 while the longitudinal position is valid within range 0 to 255. Inside the Valid\_Signal Subsystem, two subsystems are created named Lateral\_Sign\_Value\_in\_ISO\_Coordinates and Sign\_Positional\_Longitudinal to manipulate the lateral and longitudinal signals. Signal named SignPositionLateral is converted from iso to cartesian coordinates using gain block. Value of gain block is given in the requirement document. Signal named SignPositionLongitudinal to need to be processed through slider gain block with value 1 for renaming.



For deciding type of sign i.e reserved, new, updated and invalid, another subsystem is modelled named Input\_Signal\_Range\_Change. It receives its input SignType from Camera\_Data\_Conditioning subsystem, ValidLateralPosition/ValidLongitudinalPosition

from Valid\_Signal subsystem. Signal named SignType is compared with calibration values provided in Requirement Document and based on its logical output. If three conditions i.e. comparison with calibration, valid lateral and longitudinal position is satisfied, the signal is tagged as "reserved" else further conditions for status "new", "updated", "invalid" is checked.

For requirements FUN – REQ – 00013/14, two separate subsystems are designed named SuppressedOutput, EnabledData.

Inside subsystem SuppressedOutput, input signal Suppressed\_Data is compared with calibration value mentioned in requirement document using relational operator block. If condition satisfies, a Fail signal i.e. 0 passes through else True signal i.e. 1 passes further.

The similar strategy used for designing subsystem EnabledData. If condition satisfies, a True signal i.e. 1 pass further else False signal i.e. 0.

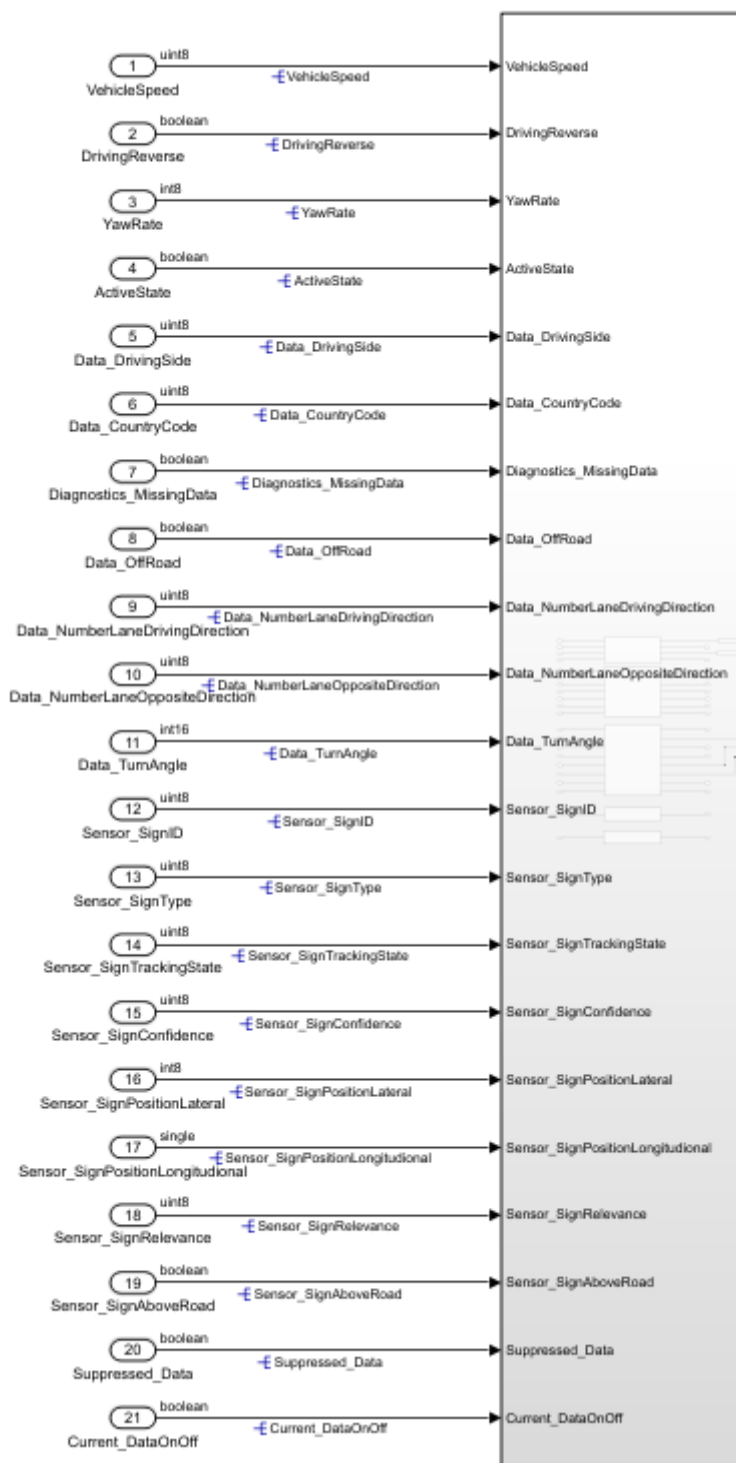
### **Step 3: SLDD Creation and Signal Resolution**

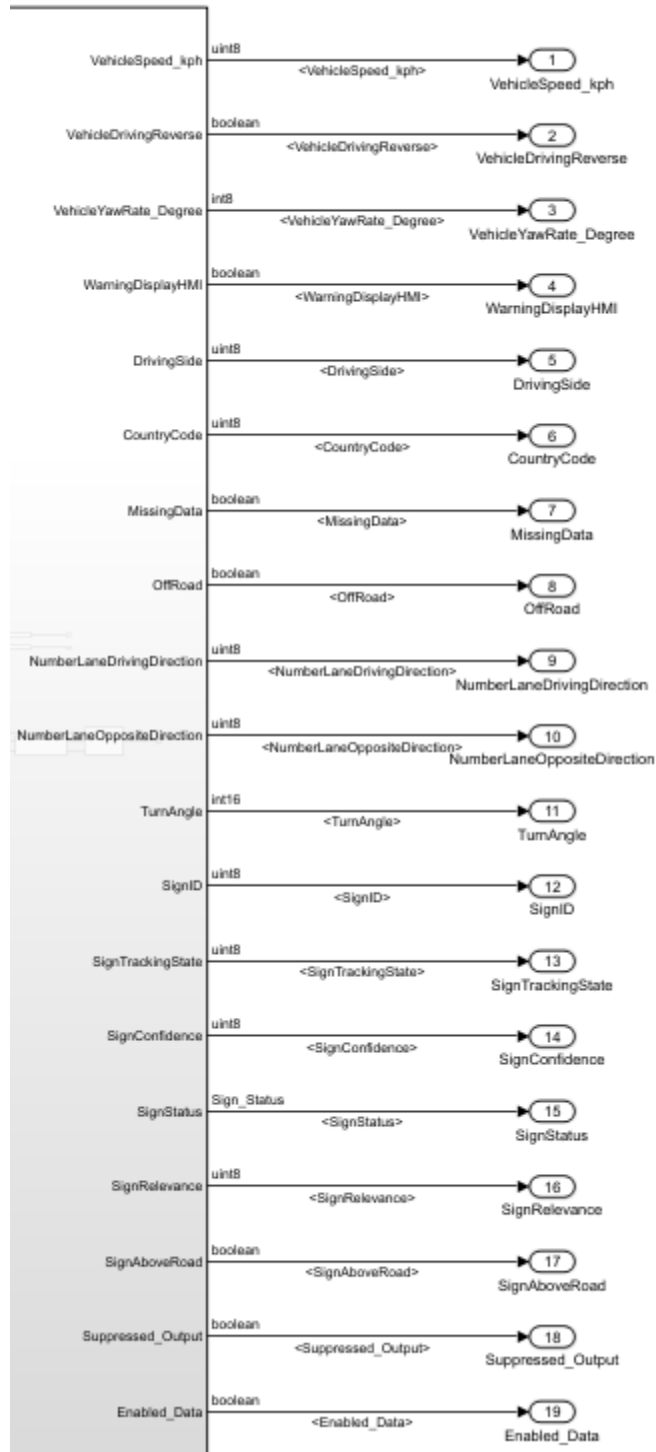
Whatever the signals used while designing a model are described in Simulink Data Dictionary. A new SLDD named WWDW\_dd.sldd is created and linked with the model. Signal Properties of all signal used are described in SLDD.

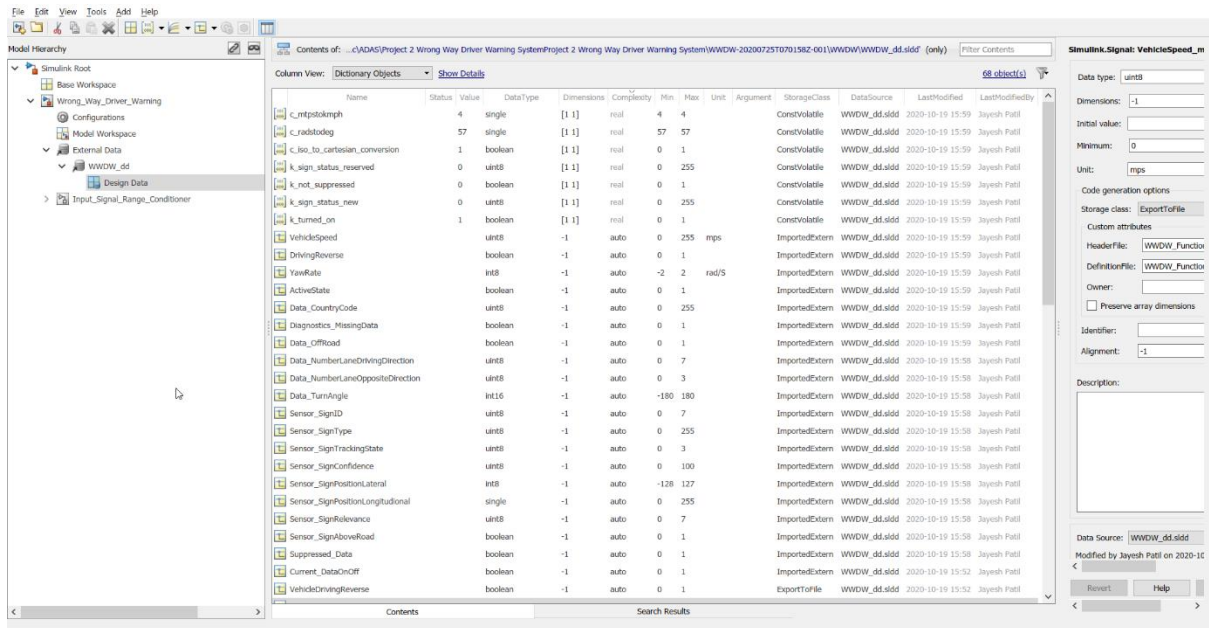
Signal properties include signal data type, minimum and maximum values, unit, storage class, header file & definition file names and dimensions. Firstly, alias types are described for all type of data types and then signals and calibration values are specified.

Once the signal being specified in SLDD, next job is to name the important signals in the model. All input and output signals are given their names specified in SLDD. If signals are originating from a block, then these signals should be marked as Simulink signal object. As these signal goes from each subsystem and then blocks, it is marked as a propagated signal in its further process.









#### Step 4: Model Advisor Check

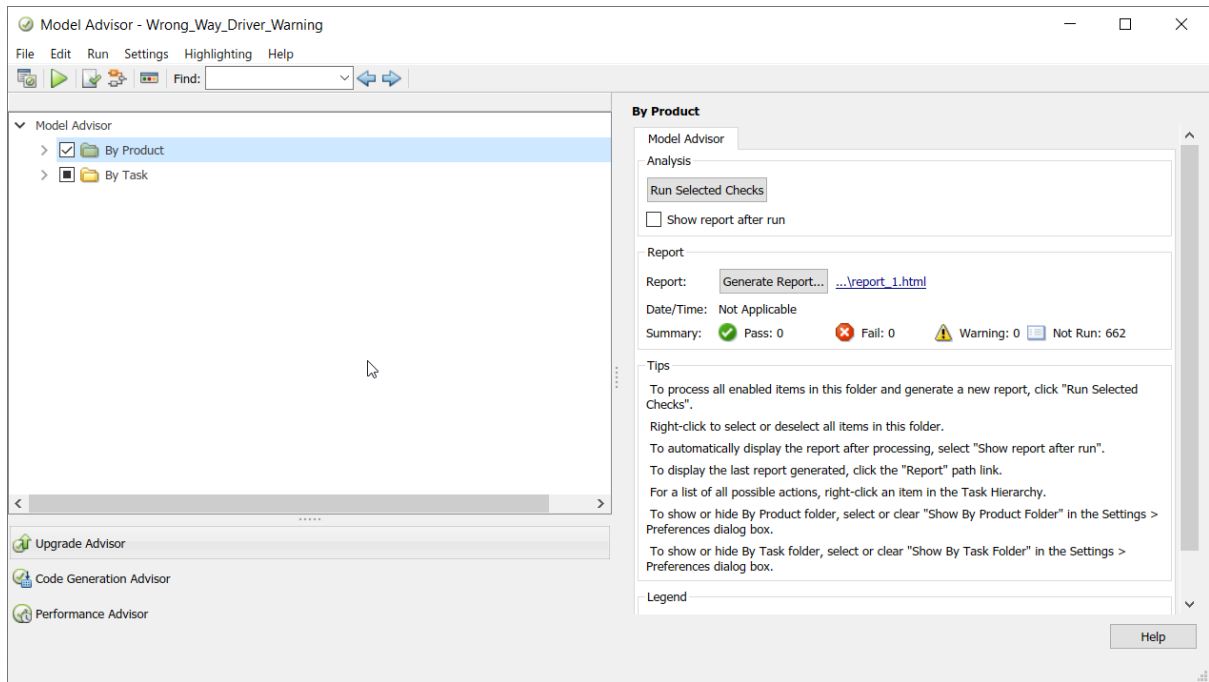
The Model Advisor checks your model or subsystem for modelling conditions and configuration settings that cause inaccurate or inefficient simulation of the system that the model represents. The Model Advisor checks can help you verify compliance with industry standards and guidelines. By using the Model Advisor, you can implement consistent modelling guidelines across projects and development teams.

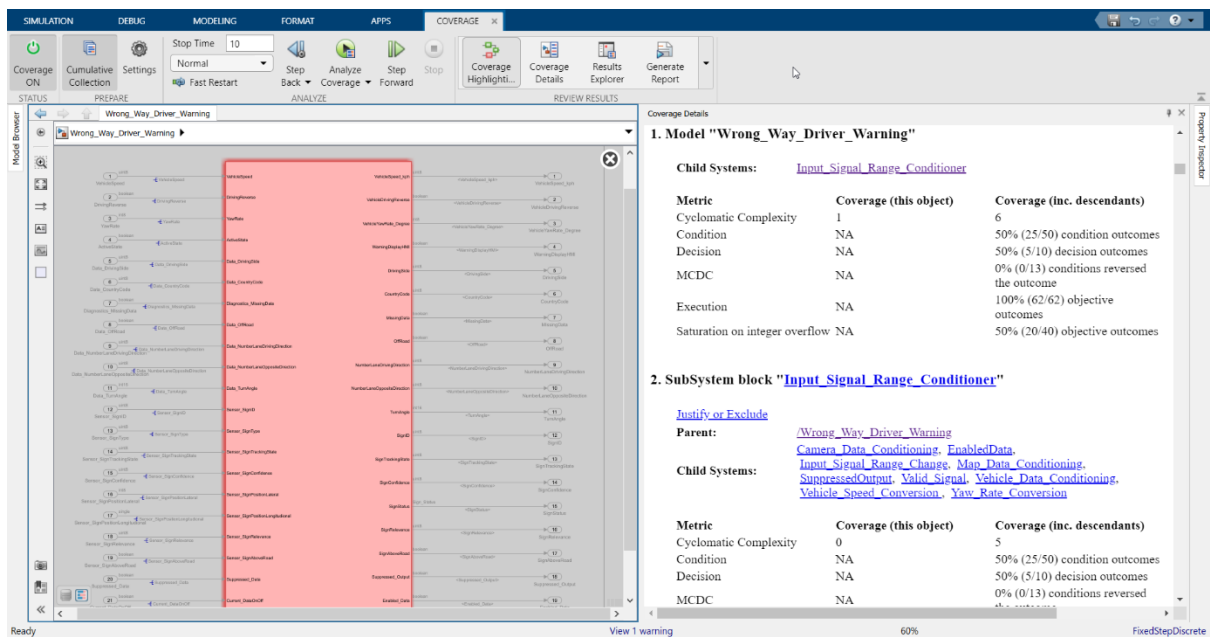
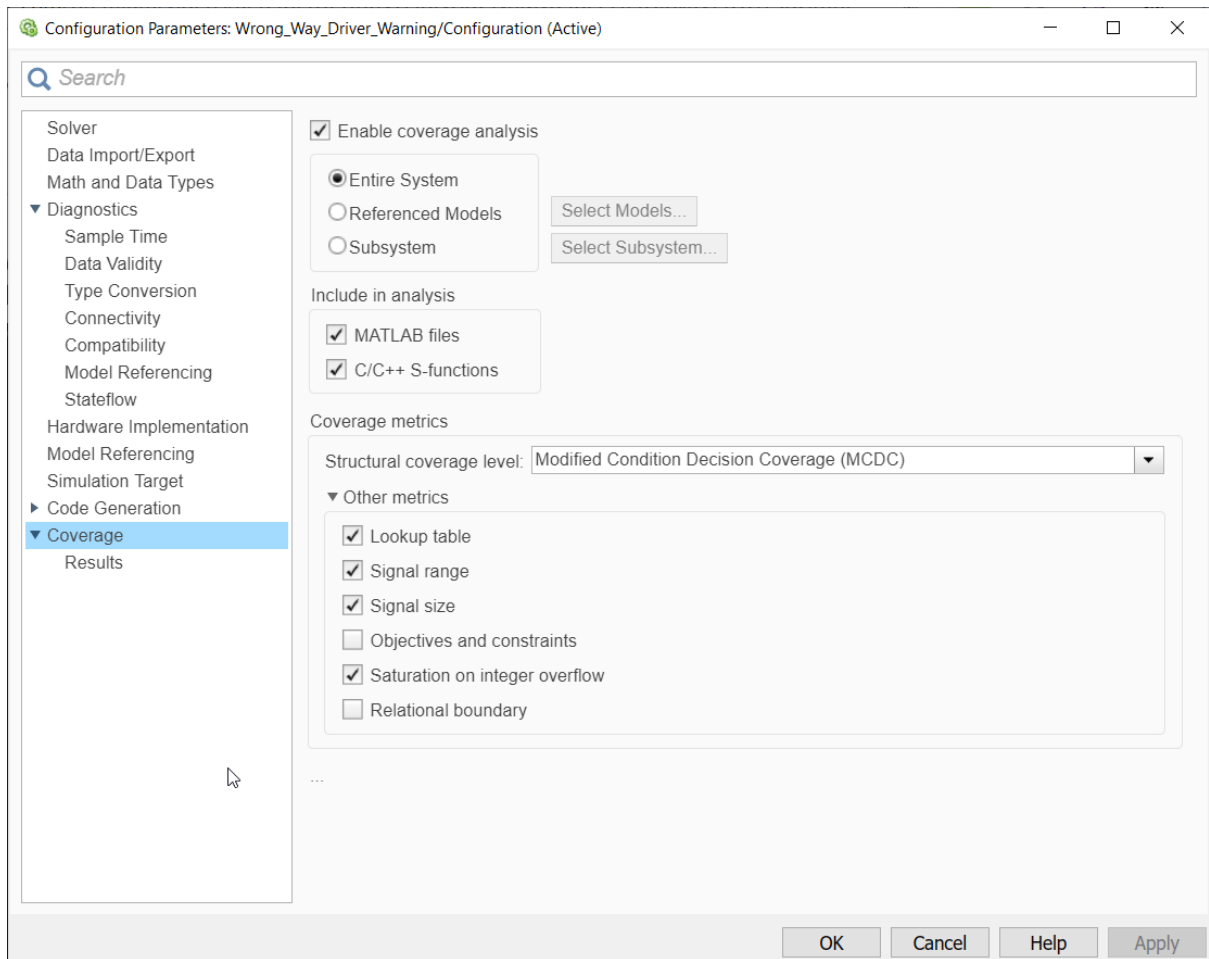
Upon completing the analysis of your model, the Model Advisor produces a report that lists the suboptimal conditions, settings, and modelling techniques and proposes solutions, when applicable.

Once the signal being specified in SLDD, next job is to name the important signals in the model. All input and output signals are given their names specified in SLDD. If signals are originating from a block, then these signals should be marked as Simulink signal object. As these signal goes from each subsystem and then blocks, it is marked as a propagated signal in its further process.

**Model Advisor Check:** Model Advisor Check option is available in Modelling Tab. As the scope of the project is small, the Advisor check is carried out for a few parameters only. Warnings got after running the check are resolved and the final report is generated and saved. The same report is also attached to the project. It is not necessary to resolve all the errors, so important errors are resolved which may hamper the code generation process.

**Configuration Parameter Update:** Once it is sure that the model doesn't have an error, the model set can be set appropriately. Model Setting option is available under Modelling Tab. All Configuration parameter is set and its images are saved in ppt. The same ppt is attached to this project.





## Coverage Report for Wrong\_Way\_Driver\_Warning

### Table of Contents

1. [Analysis Information](#)
2. [Tests](#)
3. [Summary](#)
4. [Details](#)

### Analysis Information

#### Model Information

Model version	1.138
Author	JAYESH PATIL
Last saved	Mon Oct 19 18:15:32 2020

#### Simulation Optimization Options

Default parameter behavior	inlined
Block reduction	off
Conditional branch optimization	on

#### Coverage Options

Analyzed model	Wrong_Way_Driver_Warning
Logic block short circuiting	off
MCDC mode	masking

In the model advisor, the model is checked for the following tasks and guidelines,

Modelling Standards for ISO 26262

Modelling Standards for EN 50128

Model Metrics

Code Generation Efficiency

Model Referencing

Modelling Standards for MAB

Modelling Standards for JMAAB

Modelling Standards for MISRA C:2012

**Model Advisor Report - Wrong\_Way\_Driver\_Warning.slx**

Simulink version: 10.1  
System: Wrong\_Way\_Driver\_Warning  
Treat as Referenced Model: off

Model version: 1.139  
Current run: 20-Oct-2020 13:55:34

**Run Summary**

Pass	Fail	Warning	Not Run	Total
352	69	58	598	1077

No supported compiler was found. You can install the freely available MinGW-w64 C/C++ compiler; see [Install MinGW-w64 Compiler](https://www.mathworks.com/support/compilers). For more options, visit <https://www.mathworks.com/support/compilers>.

**By Task**

- 1 Modeling Physical Systems** (0 Pass, 0 Fail, 0 Warning, 2 Not Run)
- Check consistency of block parameter units** (Not Run)
- Check for dry hydraulic nodes** (Not Run)
- 2 Simulink Code Inspector compatibility checks** (0 Pass, 69 Fail, 0 Warning, 0 Not Run)
- Check code generation settings** (No supported compiler was found. You can install the freely available MinGW-w64 C/C++ compiler; see [Install MinGW-w64 Compiler](https://www.mathworks.com/support/compilers). For more options, visit <https://www.mathworks.com/support/compilers>.)
- Check data import and export settings**

After the model advisor check report, the result was as follows -

**Pass - 352**

**Failed - 69**

**Warnings - 58**

Furthermore, correct some of the error suggested by the model advisor, the model was rechecked, this time the

the number of warnings got reduced and the number of modules that were passed got increased.

**Pass - 349**

**Failed - 0**

**Warnings - 43**

Filter checks

☒ Passed
 ☒ Failed
 ☒ Warning
 ☒ Not Run

Keywords

Navigation

By Task

1 Modeling Physical Systems
 2 Simulink Code Inspector compatibility checks
 3 Modeling Standards for DO-178C/DO-331
 3.1 High-Integrity Systems
 3.1.1 Simulink
 3.1.2 Stateflow
 3.1.3 MATLAB
 3.1.4 Configuration
 3.1.5 Naming
 3.1.6 Requirements
 3.1.7 Code
 3.2 Simulink
 3.3 Stateflow

View

[Scroll to top](#)  
[Hide check details](#)

Simulink version: 10.1

Model Advisor Report - Wrong\_Way\_Driver\_Warning.slx

Model version: 1.140

System: Wrong\_Way\_Driver\_Warning

Treat as Referenced Model: off

Current run: 20-Oct-2020 14:11:38

Run Summary

Pass	Fail	Warning	Not Run	Total
349	0	43	685	1077

By Task

1 Modeling Physical Systems
 0 0 0 2

Check consistency of block parameter units

Not Run

Check for dry hydraulic nodes

Not Run

2 Simulink Code Inspector compatibility checks

0 0 0 69

Check code generation settings

Not Run

Check data import and export settings

Not Run

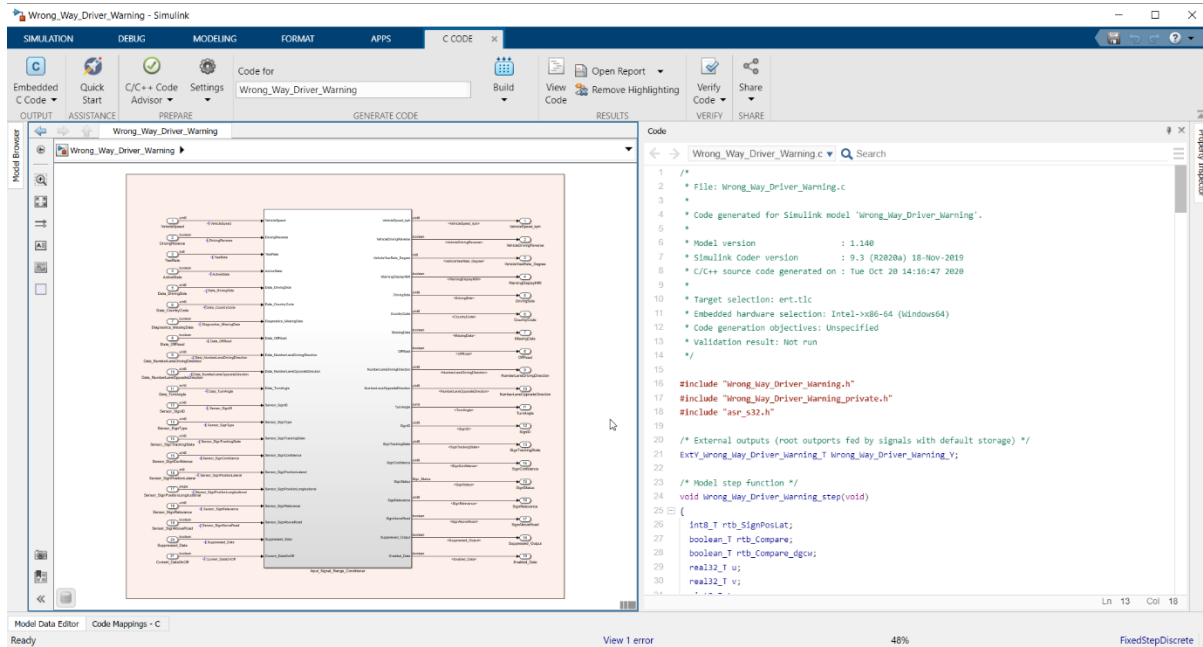
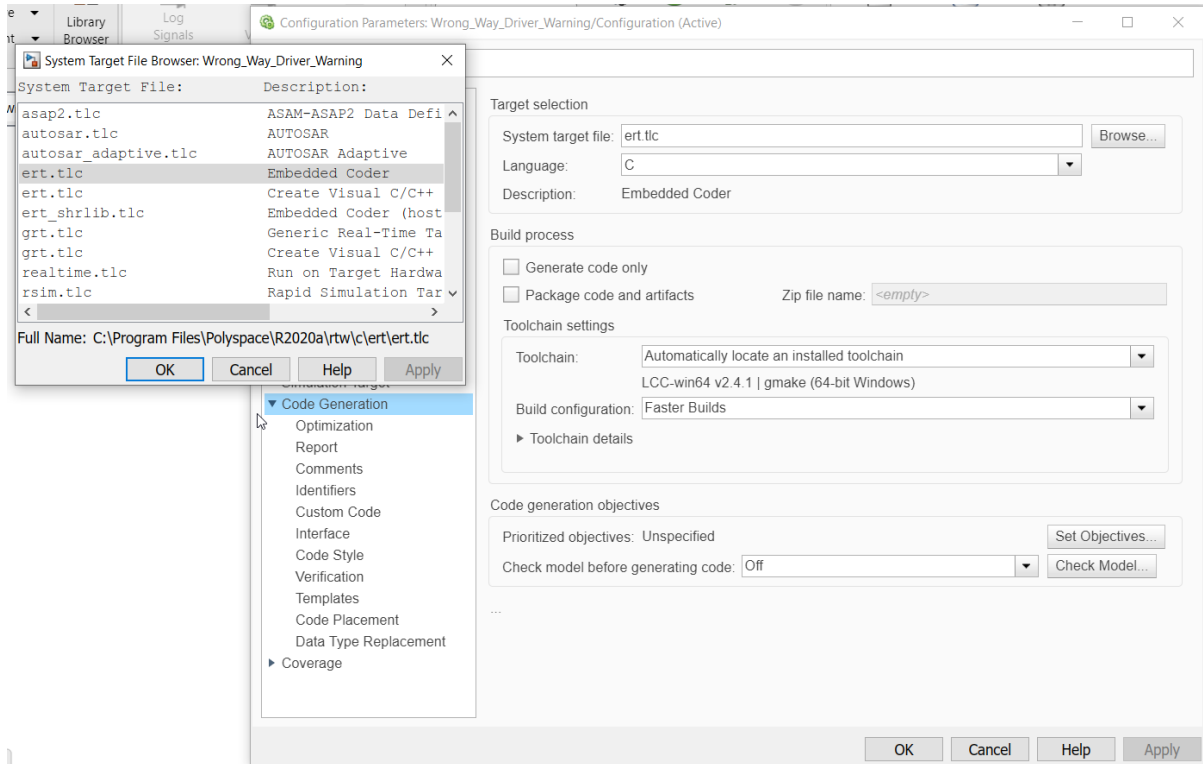
The report is then generated for the given Simulink model.

## Analysis and Results

### Step 5: Code Generation

Last Part of the model design is code generation. Once all necessary changes been done as per model advisor, code generation can be started. ctrl+B is a shortcut key to start code generation. After successful completion of the code generation process, c-code is generated for the model created in Simulink.





## Code Generation -

```
/*
 * File: Wrong_Way_Driver_Warning.c
 *
 * Code generated for Simulink model 'Wrong_Way_Driver_Warning'.
 *
 * Model version          : 1.135
 * Simulink Coder version  : 9.3 (R2020a) 18-Nov-2019
 * C/C++ source code generated on : Wed Jul 15 18:04:00 2020
 *
 * Target selection: ert.tlc
 * Embedded hardware selection: Intel->x86-64 (Windows64)
 * Code generation objectives: Unspecified
 * Validation result: Not run
 */

#include "Wrong_Way_Driver_Warning.h"
#include "Wrong_Way_Driver_Warning_private.h"
#include "asr_s32.h"

/* External outputs (root outputs fed by signals with default storage) */
ExtY_Wrong_Way_Driver_Warning_T Wrong_Way_Driver_Warning_Y;

/* Model step function */
void Wrong_Way_Driver_Warning_step(void)
{
    int8_T rtb_SignPosLat;
    boolean_T rtb_Compare;
    boolean_T rtb_Compare_dgcw;
```

## Conclusion

Requirement Document :

Wrong\_Way\_Driver\_Warning\_System\_Requirement\_Document.docx

Model File : Wrong\_Way\_Driver\_Warning\_2019b.slx

SLDD File : WWDW\_dd.sldd

Model Advisor Report : Model\_Advisor\_Report.pdf

C File : Wrong\_Way\_Driver\_Warning.c