Tilt & Telescopic Steering Switch

Prepared by:
Nishantkuamr V Patel

INDEX

1. Objectives

- To create a Model (Model Based Development/ Design) of Tilt & Telescopic Steering Column Switch and Perform Code generation report using the State flow chart logics and Embedded Coder in MATLAB & Simulink.
- Read and studied out the all types of requirements regarding to tilt telescopic switch.
- Then according to them, Model, Simulink Data Dictionary (SLDD) will be developed using Simulink.
- Whole model checking and inspecting under the standards like JMAAB, MAAB, ISO 26262 with the help of Model Advisor App in Simulink.
- Then using Embedded Coder App the code has to be generated in C- language.
- To show and attach the main ert.tlc file and model files (c-files, header files) into report.

2. General Overview

- This feature is widely used in modern automobiles which helps to adjust the steering wheel before the driver starts driving. The driver can adjust according to his/her comfort level. This feature also gives enough space for the driver which helps in entry/exit from the vehicle. Modern vehicles use an electronic switch in order to adjust the steering column to move UP, DOWN, FORWARD and REARWARD. The driver should not use this feature while driving. It may lead to losing control over the steering and could lead to accidents/injuries.

- The UP/ DOWN directions is kind of vertical adjustment whereas FORWARD/ REARWARD directions is kind of inside/ outside movement of steering.

- Below parameters are play important role for this feature.

• Switch contact Debounce
• Determining the state of four-dimensional switch: Tilt-Up, Tilt-Down, Telescopic forward and Telescopic Rearward.
• Transmission of CAN signals
• Detection of wake up input Activation (neglect for now)
• Diagnostic monitoring o switch input (neglect for now)

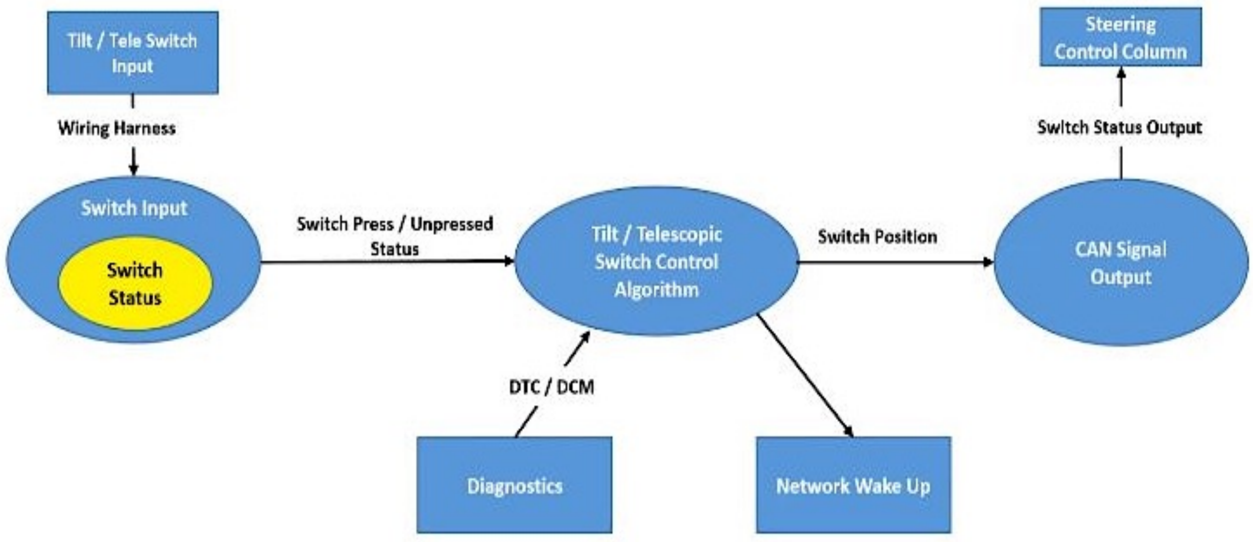


Image Courtesy: Internet

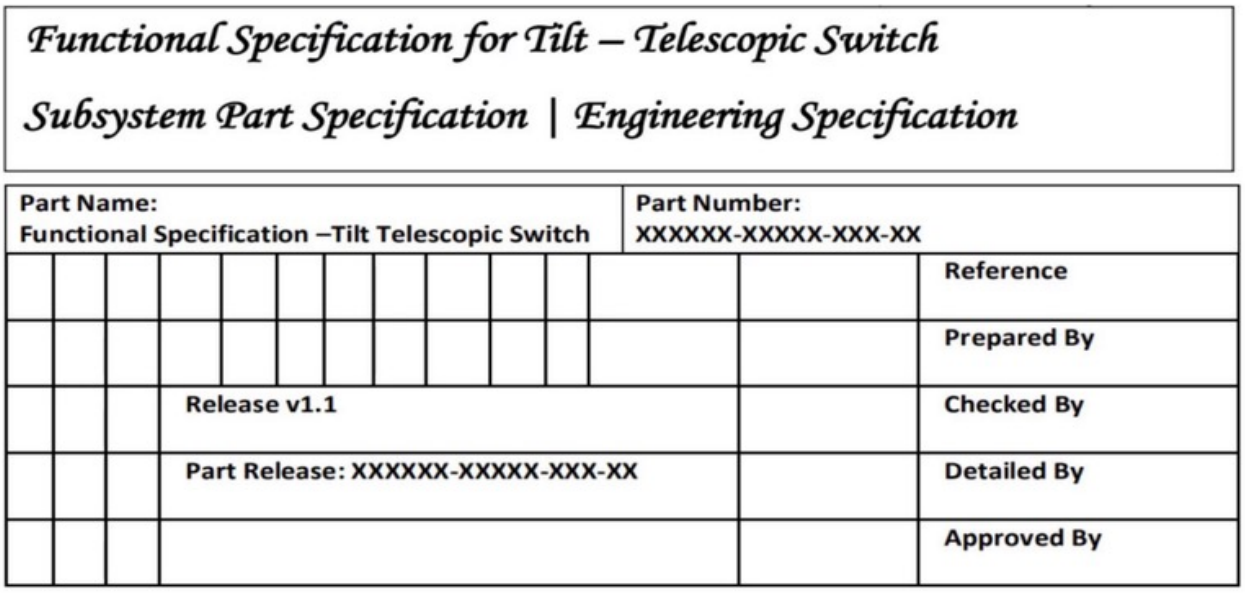
The above picture is bascially the Working Diagram of switch.

- Tilt and Telescopic Switch is a physical switch which the driver gives his/ her inputs for adjustments in different directions. The physical switch should be flexible to move in all four directions. Wiring harness connects the switch with ECU in order to transfer input signals.

- Switch input is the interface at model level to identify the status (present exact position) whether it is pressed, unpressed or stuck condition. Based on the status of the switch the signals are sent to 'Control Algorithm' to perform its action. This algorithm detects the user input and switch position and then sends signals through CAN bus protocol to perform actions as per the user inputs.

- This can be proved as a better technique for driver Ergonomics while driving as it offers more space, better position for steering wheel to use it at ease level.

3. Requirement Analysis



Functional Specification for Tilt – Telescopic Switch

Subsystem Part Specification | Engineering Specification

Part Name: Functional Specification –Tilt Telescopic Switch	**Part Number:** XXXXXX-XXXXX-XXX-XX	
		Reference
		Prepared By
Release v1.1		**Checked By**
Part Release: XXXXXX-XXXXX-XXX-XX		**Detailed By**
		Approved By

Image Courtesy: Skill Lync

- The above photo is the example of Requiment analysis report made by the development team and reuirement manager at stage-1 of V-model cycle in automotive industry. This is just an demo as this report format can be varies from company to company.
- The Reuirement analysis includes all kind of reuirements like functional and general.
- Feature owner gives the requirement that has to be developed. Every individual part of software/hardware will have part numbers. Usually, ECU is a part member.

- <u>3.1 General Requirements:</u>

- The tilt & telescopic switch interface has to wake up if the ignition status is RUN available at CAN Bus.
- At wake up, all the buttons states shall be in <Switch Not Pressed> state.

- Battery voltage shoud be between 10 to 20 Volt range.

▪ 3.2 Functional Description:

- Switch contact debouncing.
- Determine Switch states- tilt up/ tilt down/ telescopic forward/ telescopic rearward.
- Transmission of CAN signals.
- Detection of wakeup input activation.
- Diagnostic monitoring of the switch input.

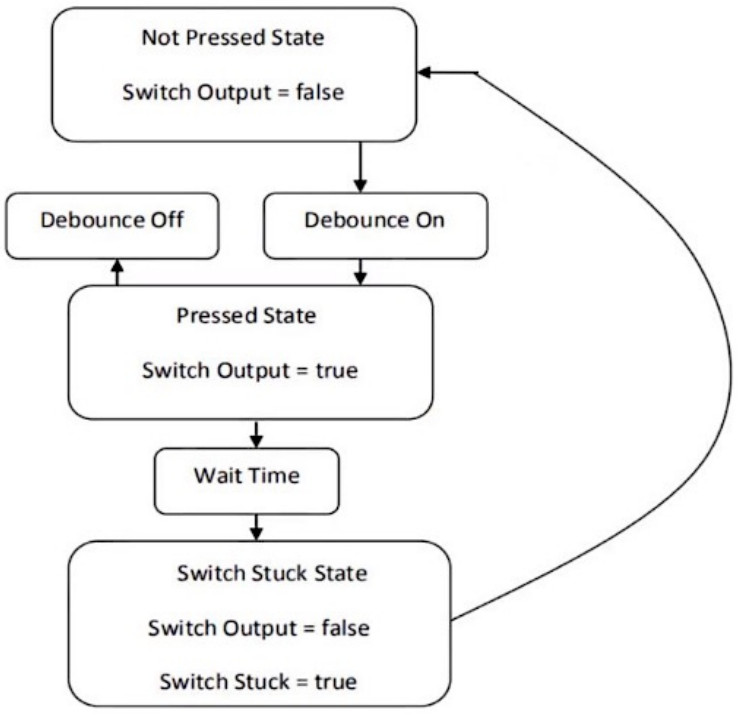
▪ 3.3 Functional Requirements (Physical Switch States):

Switch Position	Description	Tilt/ Telescopic Logical States
Neutral	go to Switch	TTSw_Off
Up	push switch Upward	TTSw_Up
Down	push switch Downward	TTSw_Down
Forward	push switch Forward	TTSw_Frwd
Rearward	push switch Rearward	TTSw_Rrwrd
Stuck	hardware/ user Error	TTSw_stuckswitch

▪ 3.4 Dignostic Management Reuirement Analysis:

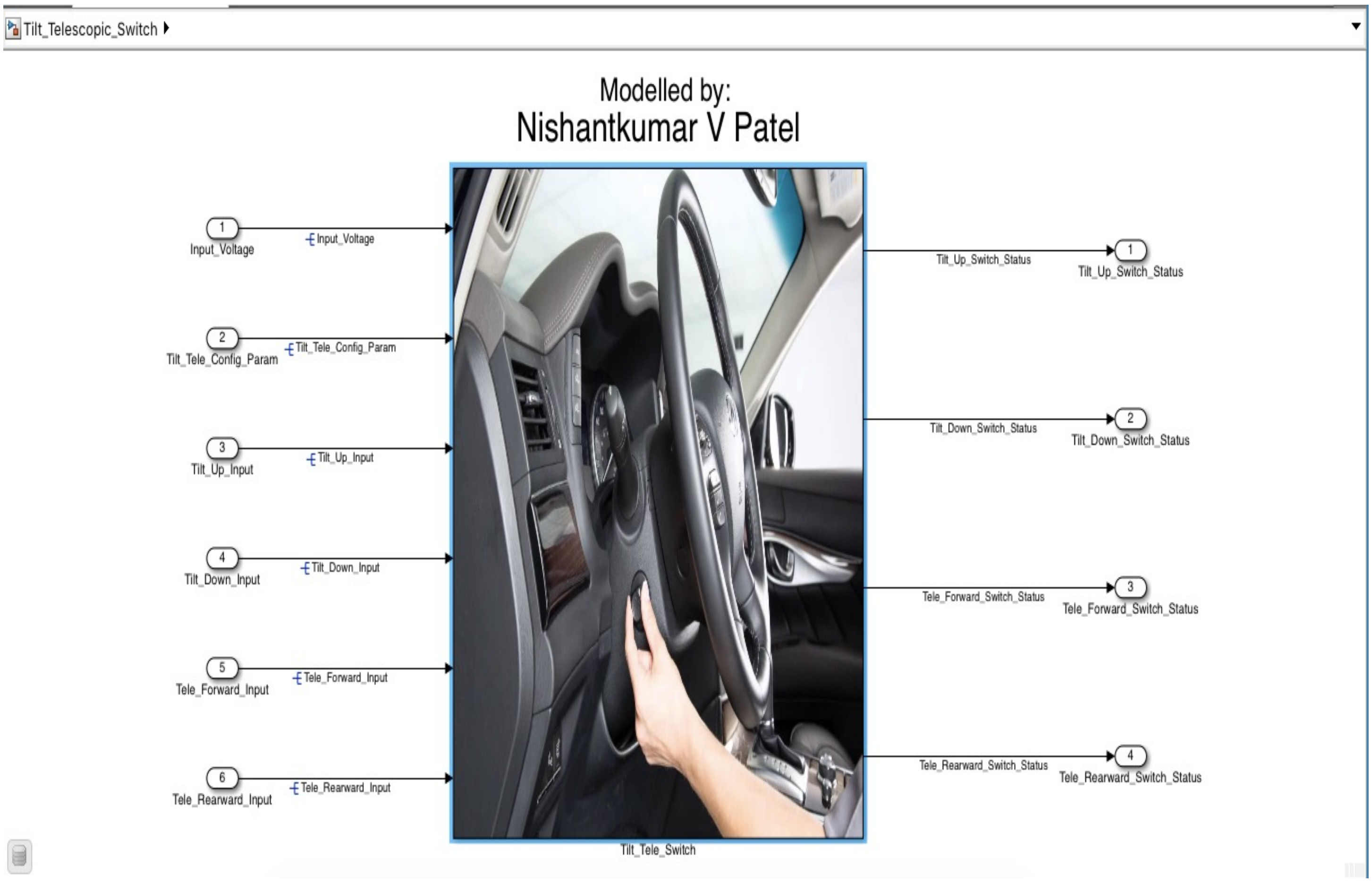
- The initial test for the switch interface shall start atleast 250 milliseconds after wakeup (200 ms powerup time+ 50 ms debounce time)
- A valid voltage drop is observed during <Switch Not Pressed> as it is electronically operated module.

■ 3.5 Functional Requirements (Debounce Work-flow):

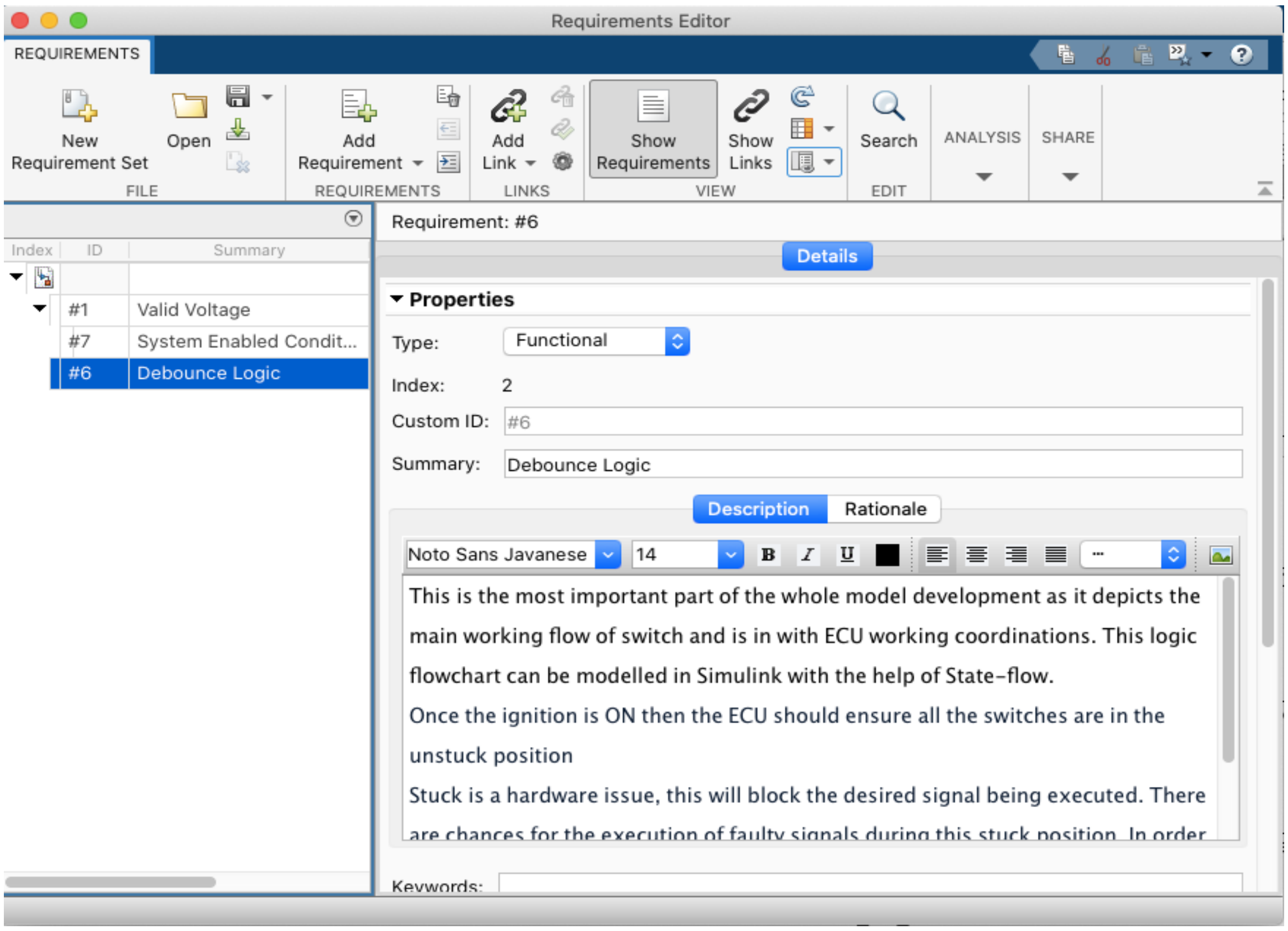


- This is the most important part of the whole model development as it depicts the main working flow of switch and is in with ECU working coordinations. This logic flowchart can be modelled in Simulink with the help of State-flow.
- Once the ignition is ON then the ECU should ensure all the switches are in the unstuck position
- Stuck is a hardware issue, this will block the desired signal being executed. There are chances for the execution of faulty signals during this stuck position. In order to avoid these errors debounce time is being introduced. This will avoid/fix all the input errors and the actual signal will be processed.

4. Model Development in Simulink



Requirement Editor:



ID	Summary
#1	Valid Voltage
#7	System Enabled Condit...
#6	Debounce Logic

Requirement: #6

Type: Functional

Index: 2

Custom ID: #6

Summary: Debounce Logic

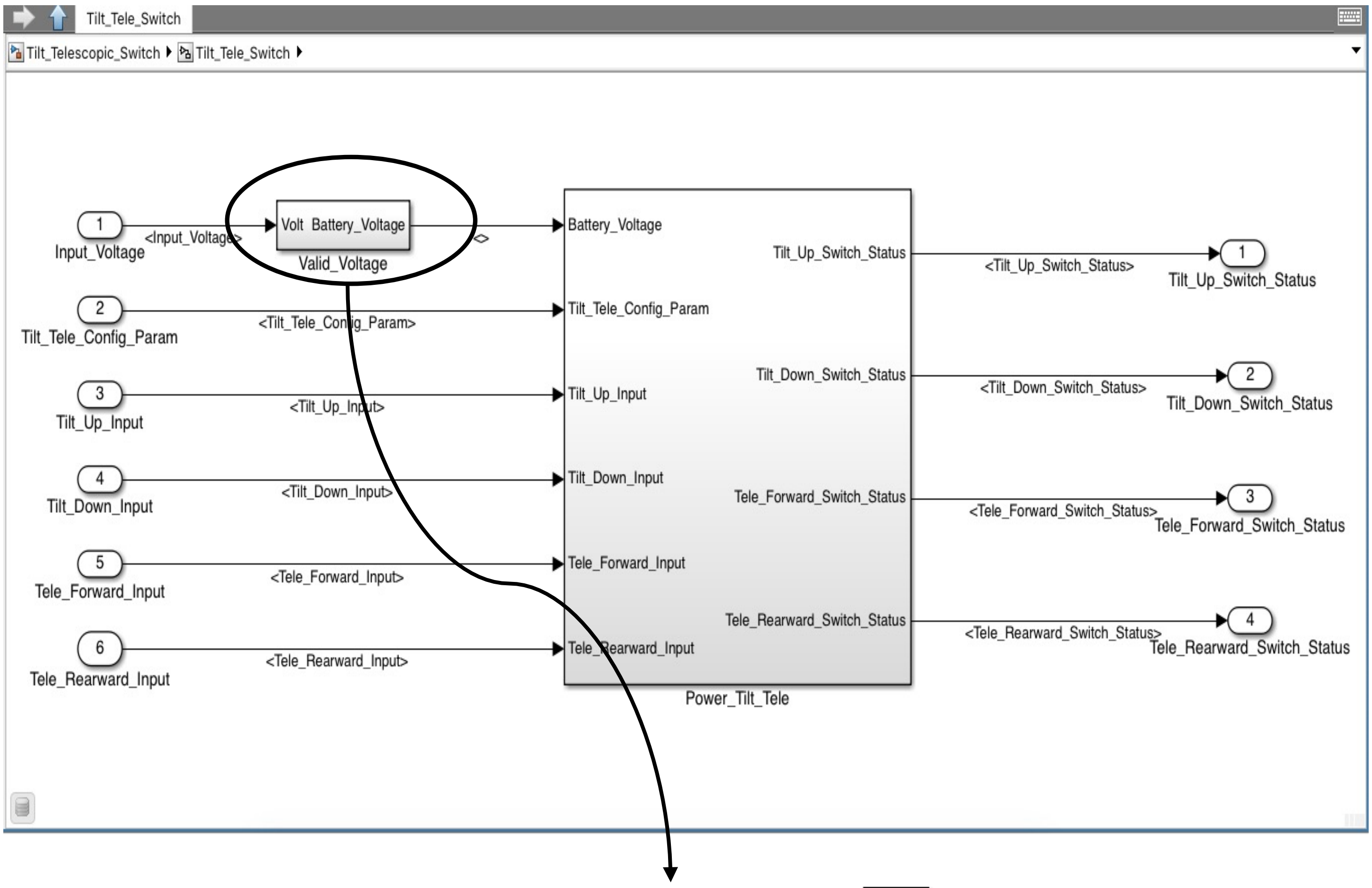
Noto Sans Javanese 14

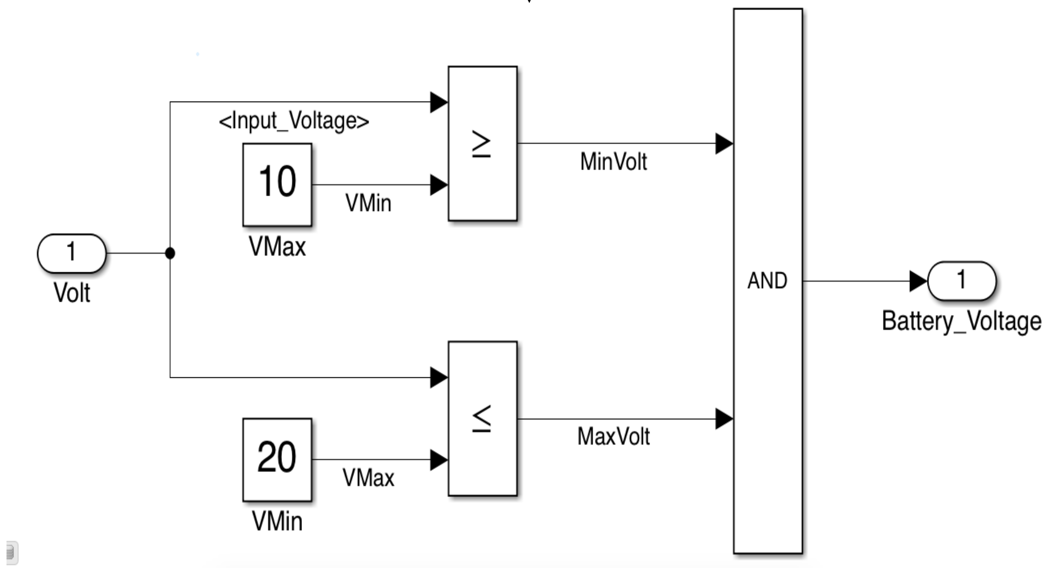
This is the most important part of the whole model development as it depicts the main working flow of switch and is in with ECU working coordinations. This logic flowchart can be modelled in Simulink with the help of State-flow.
Once the ignition is ON then the ECU should ensure all the switches are in the unstuck position
Stuck is a hardware issue, this will block the desired signal being executed. There are chances for the execution of faulty signals during this stuck position. In order

Keywords:

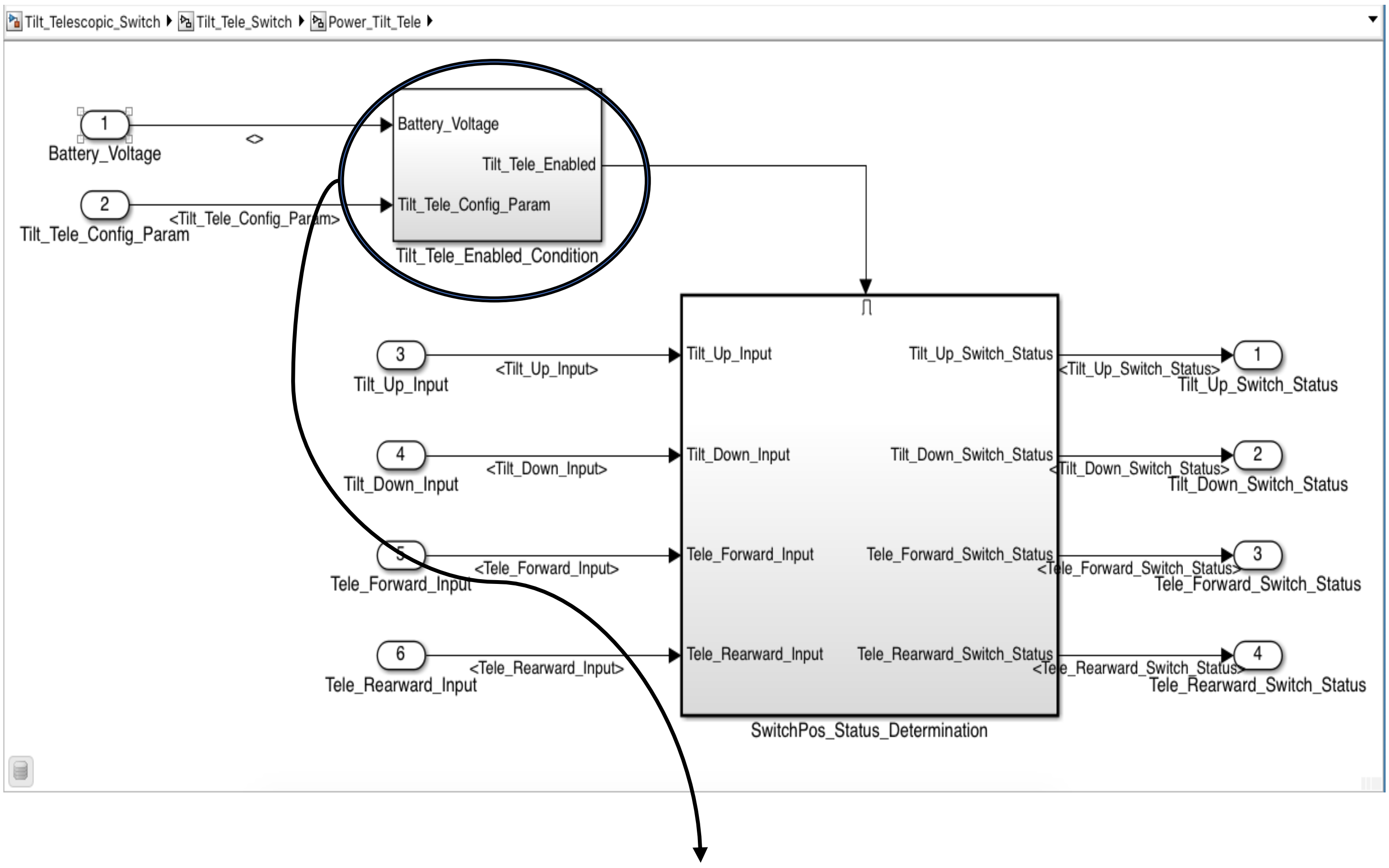
- I have attached the 3 major requirements regarding to this model. One is valid voltage seconds one is system enalbed configuration condition. The last one is about the main and most important requirement logic used behind the whole system.

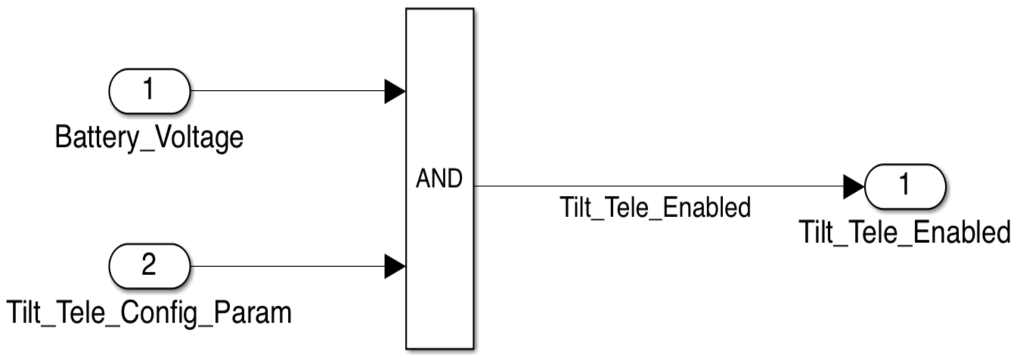
- (.slreqx) extension file includes the requirement with discriptions. Whereas (.slmx) extension file includes the links of that described requirements attached to their respective and corresponding block or subsystems in the model.



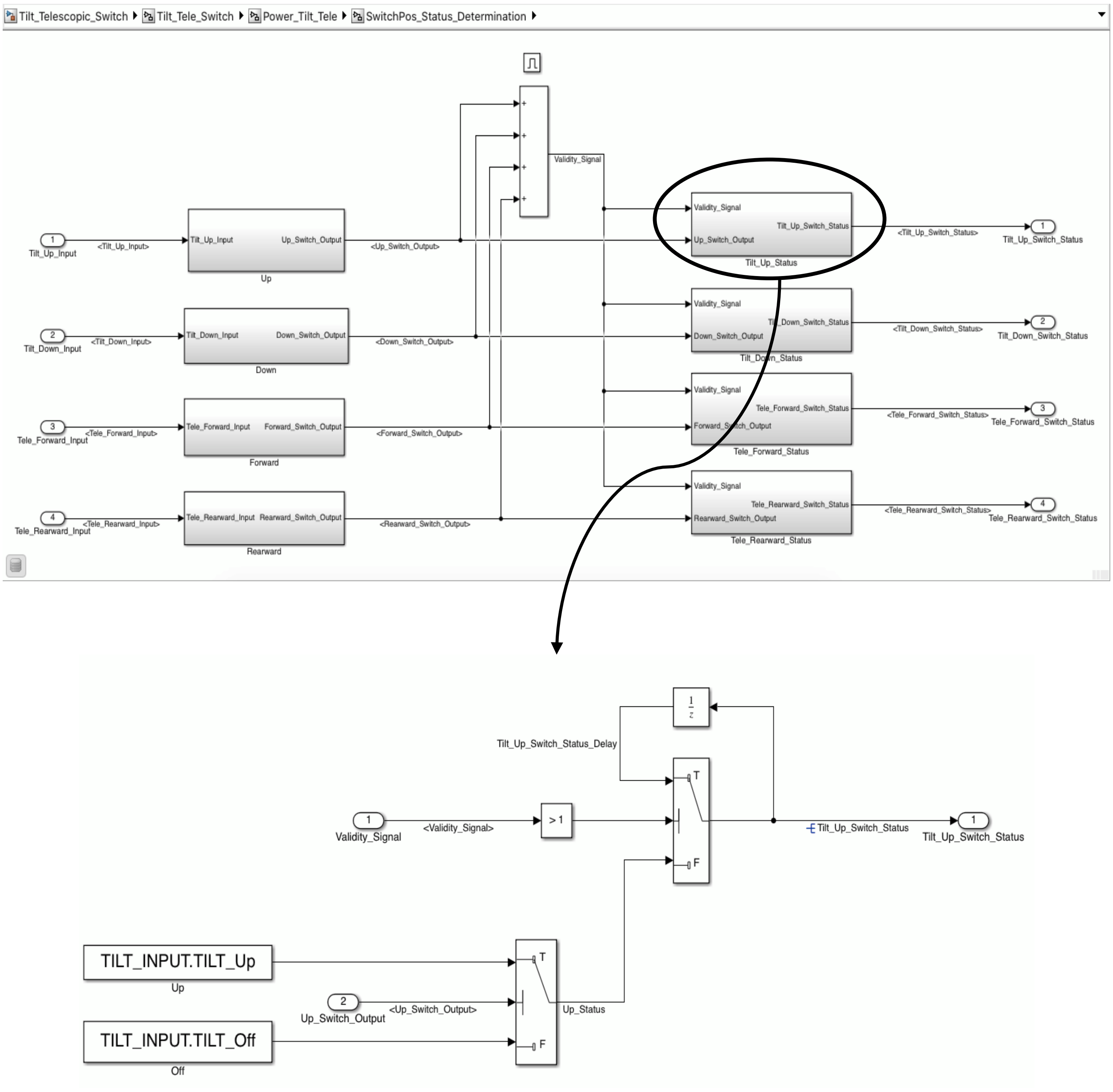


- This is the operating range of Battery voltage between 10 to 20 volts.

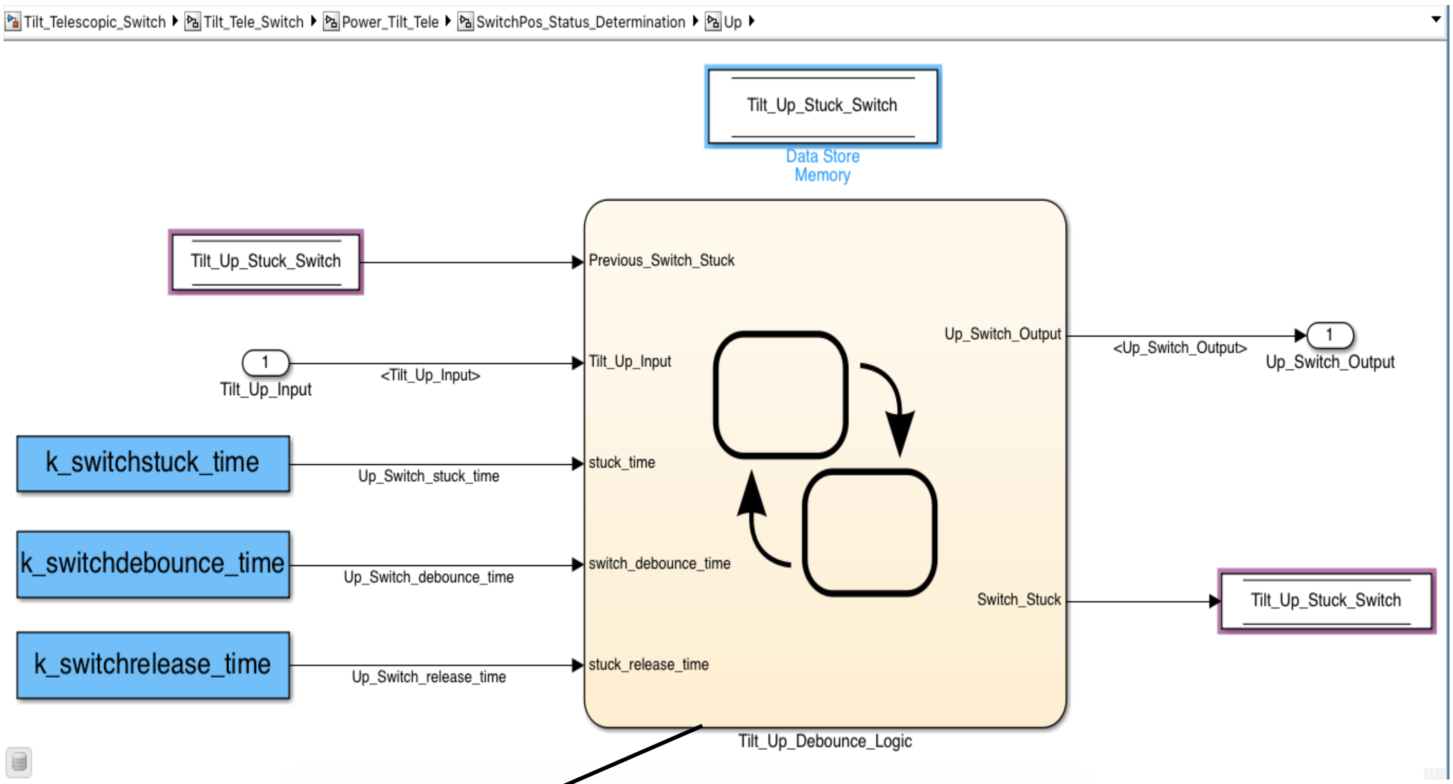


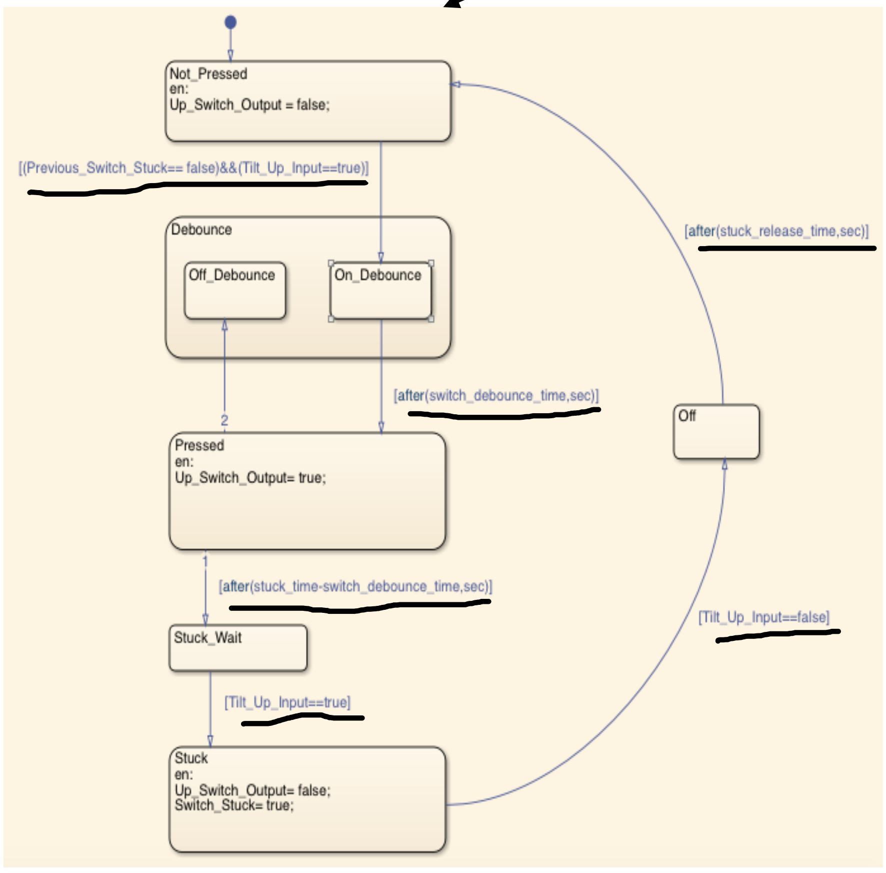


- This is the general erequirement condition that if battery voltage as well as configuration parameters both are satisfied then only the whole subsystem for switch able to activated/ enabled.

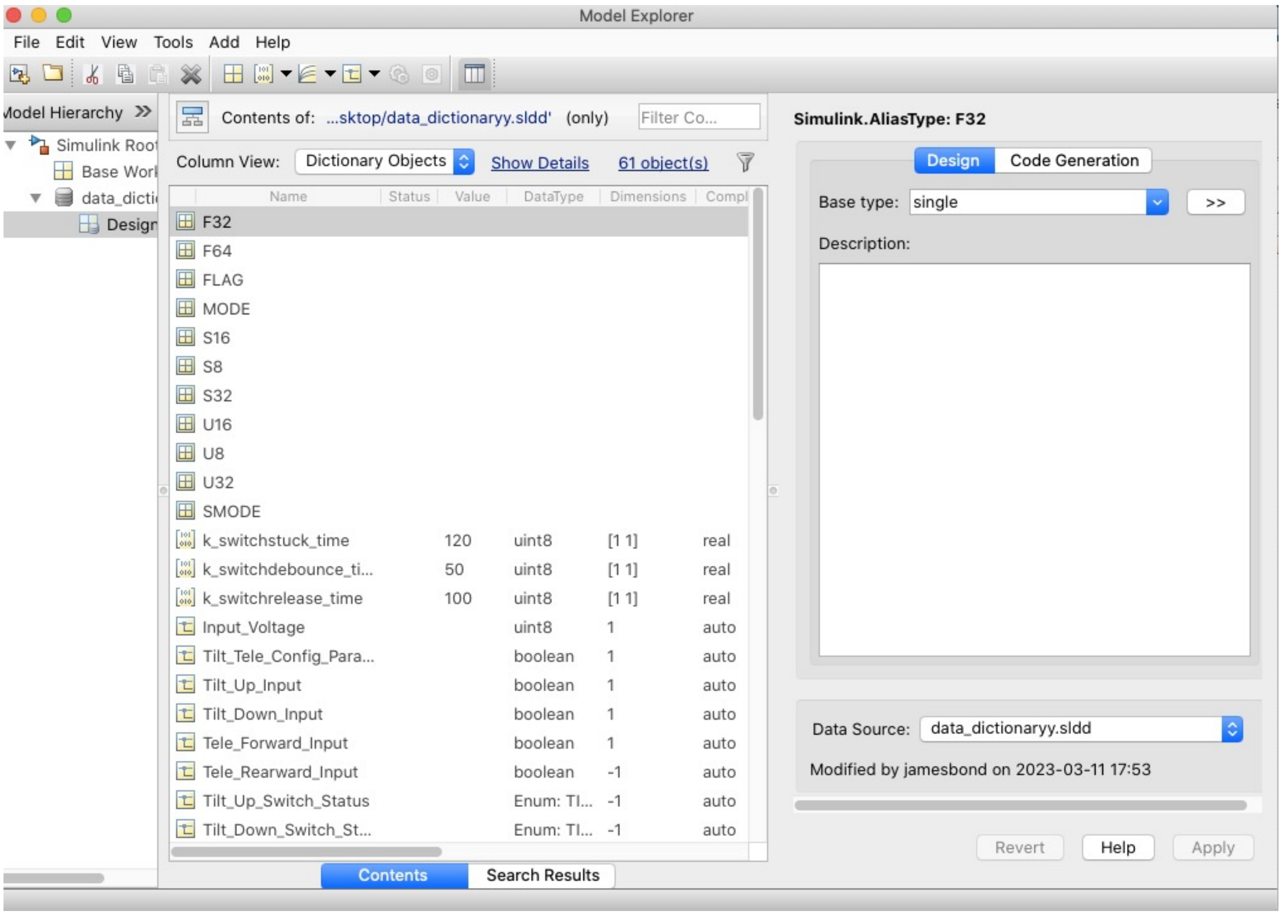


- If the validity signal>1 then Tilt Up switch status delay hase to be implemented. Where as if it is false then the new else if condition is introduced with another switch block.


Tilt_Telescopic_Switch ▸ Tilt_Tele_Switch ▸ Power_Tilt_Tele ▸ SwitchPos_Status_Determination ▸ Up ▸
Tilt_Up_Stuck_Switch
Data Store Memory
Tilt_Up_Stuck_Switch
Previous_Switch_Stuck
1
Tilt_Up_Input
<Tilt_Up_Input>
Tilt_Up_Input
k_switchstuck_time
Up_Switch_stuck_time
stuck_time
k_switchdebounce_time
Up_Switch_debounce_time
switch_debounce_time
k_switchrelease_time
Up_Switch_release_time
stuck_release_time
Up_Switch_Output
<Up_Switch_Output>
1
Up_Switch_Output
Switch_Stuck
Tilt_Up_Stuck_Switch
Tilt_Up_Debounce_Logic


Not_Pressed
en:
Up_Switch_Output = false;
[(Previous_Switch_Stuck== false)&&(Tilt_Up_Input==true)]
Debounce
Off_Debounce
On_Debounce
[after(stuck_release_time,sec)]
[after(switch_debounce_time,sec)]
2
Off
Pressed
en:
Up_Switch_Output= true;
1
[after(stuck_time-switch_debounce_time,sec)]
Stuck_Wait
[Tilt_Up_Input==false]
[Tilt_Up_Input==true]
Stuck
en:
Up_Switch_Output= false;
Switch_Stuck= true;

<u>Simulink Data Dictionary (SLDD File):</u>



Contents of: ...sktop/data_dictionaryy.sldd' (only)

Column View: Dictionary Objects Show Details 61 object(s)

Name	Status	Value	DataType	Dimensions	Compl
F32					
F64					
FLAG					
MODE					
S16					
S8					
S32					
U16					
U8					
U32					
SMODE					
k_switchstuck_time		120	uint8	[1 1]	real
k_switchdebounce_ti...		50	uint8	[1 1]	real
k_switchrelease_time		100	uint8	[1 1]	real
Input_Voltage			uint8	1	auto
Tilt_Tele_Config_Para...			boolean	1	auto
Tilt_Up_Input			boolean	1	auto
Tilt_Down_Input			boolean	1	auto
Tele_Forward_Input			boolean	1	auto
Tele_Rearward_Input			boolean	-1	auto
Tilt_Up_Switch_Status			Enum: Tl...	-1	auto
Tilt_Down_Switch_St...			Enum: Tl...	-1	auto

Simulink.AliasType: F32

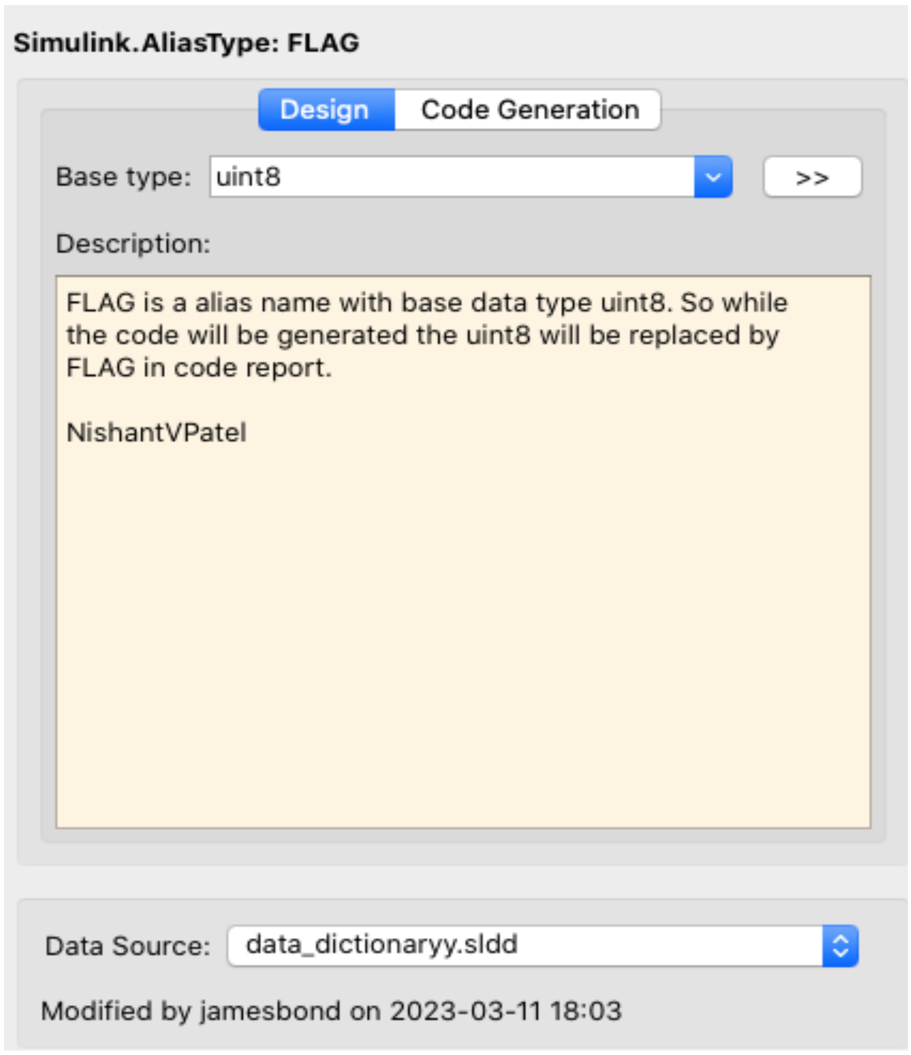
Base type: single

Description:

Data Source: data_dictionaryy.sldd

Modified by jamesbond on 2023-03-11 17:53

- This is the screenshot I have taken form my developed sldd file. Data dictionary file contains <u>Alias type signal, signals, Simulink parameters for calibrated values, Enumerate type</u> with different and appropriate ***Design*** and ***Code Generation*** settings.
- Then this sldd file has to be linked with target and relevant slx simulink file in order for the code generation.
- Sldd helps to create the signals traceability and tagging as well as to create the final model code.
- By creating sldd file one do not have to create the variable in workspace.



Simulink.AliasType: FLAG

Base type: uint8

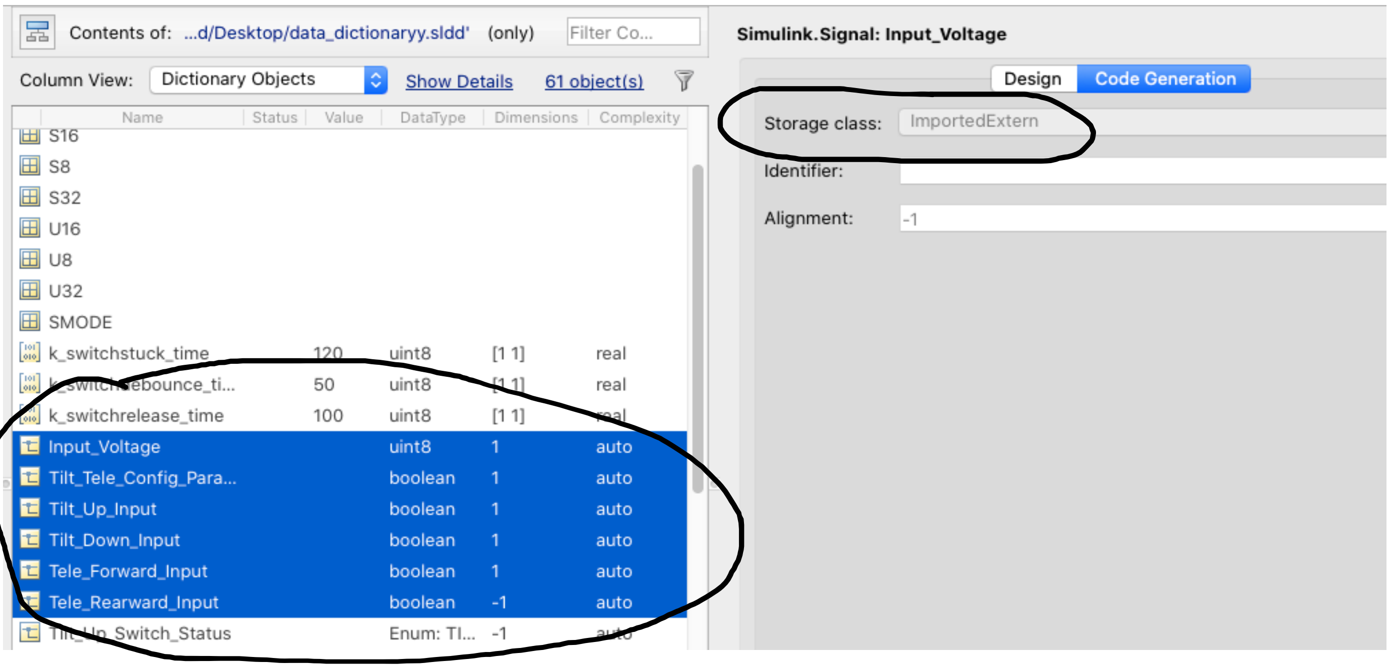
Description:

FLAG is a alias name with base data type uint8. So while the code will be generated the uint8 will be replaced by FLAG in code report.

NishantVPatel

Data Source: data_dictionaryy.sldd

Modified by jamesbond on 2023-03-11 18:03



Contents of: ...d/Desktop/data_dictionaryy.sldd' (only) Filter Co...

Column View: Dictionary Objects Show Details 61 object(s)

Name	Status	Value	DataType	Dimensions	Complexity
S16					
S8					
S32					
U16					
U8					
U32					
SMODE					
k_switchstuck_time		120	uint8	[1 1]	real
k_switchdebounce_ti...		50	uint8	[1 1]	real
k_switchrelease_time		100	uint8	[1 1]	real
Input_Voltage			uint8	1	auto
Tilt_Tele_Config_Para...			boolean	1	auto
Tilt_Up_Input			boolean	1	auto
Tilt_Down_Input			boolean	1	auto
Tele_Forward_Input			boolean	1	auto
Tele_Rearward_Input			boolean	-1	auto
Tilt_Up_Switch_Status			Enum: TI...	-1	auto

Simulink.Signal: Input_Voltage

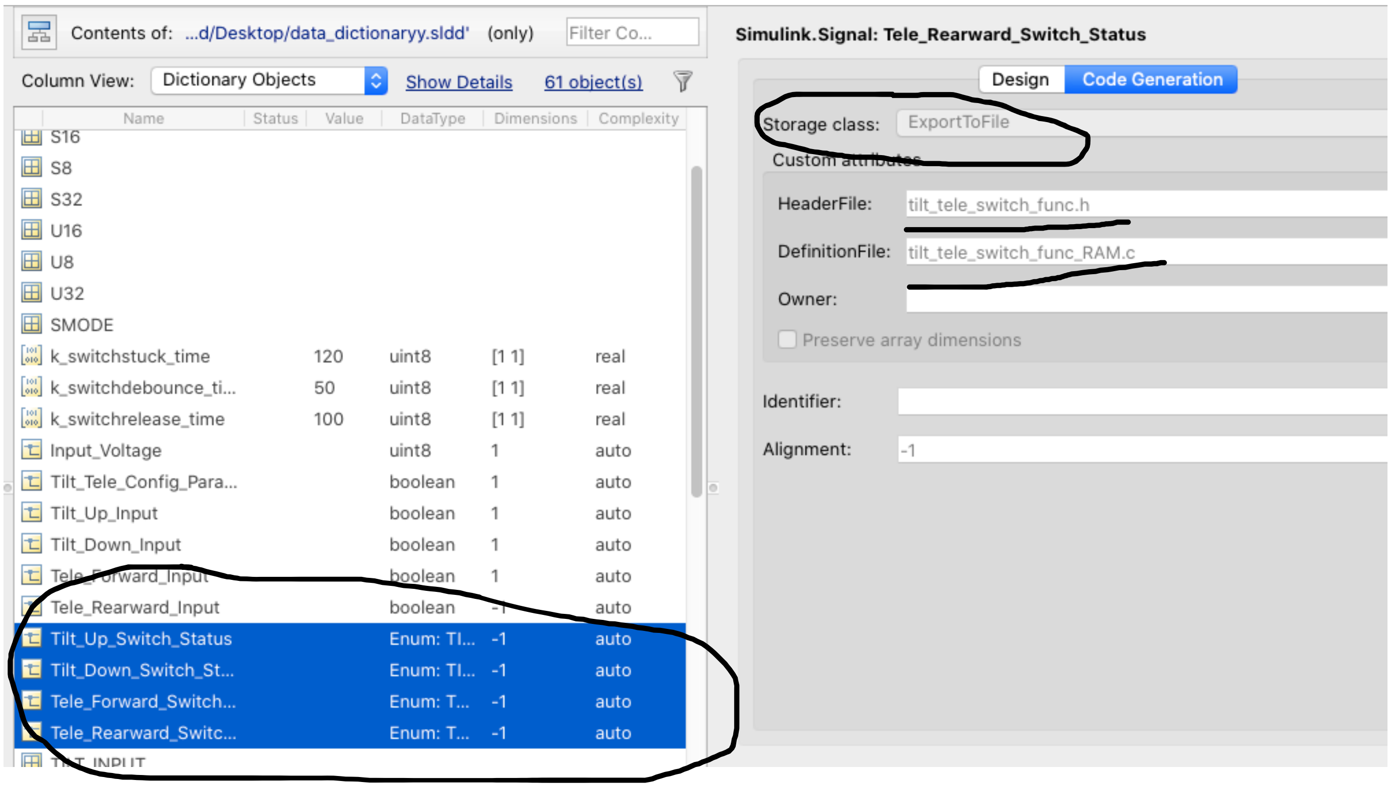
Design | Code Generation

Storage class: ImportedExtern

Identifier:

Alignment: -1

- The input signals are come under the *ImportedExtern* storage class.



Contents of: ...d/Desktop/data_dictionaryy.sldd' (only) Filter Co...

Column View: Dictionary Objects Show Details 61 object(s)

Name	Status	Value	DataType	Dimensions	Complexity
S16					
S8					
S32					
U16					
U8					
U32					
SMODE					
k_switchstuck_time		120	uint8	[1 1]	real
k_switchdebounce_ti...		50	uint8	[1 1]	real
k_switchrelease_time		100	uint8	[1 1]	real
Input_Voltage			uint8	1	auto
Tilt_Tele_Config_Para...			boolean	1	auto
Tilt_Up_Input			boolean	1	auto
Tilt_Down_Input			boolean	1	auto
Tele_Forward_Input			boolean	1	auto
Tele_Rearward_Input			boolean	-1	auto
Tilt_Up_Switch_Status			Enum: TI...	-1	auto
Tilt_Down_Switch_St...			Enum: TI...	-1	auto
Tele_Forward_Switch...			Enum: T...	-1	auto
Tele_Rearward_Switc...			Enum: T...	-1	auto
TILT_INPUT					

Simulink.Signal: Tele_Rearward_Switch_Status

Design | Code Generation

Storage class: ExportToFile

Custom attributes

HeaderFile: tilt_tele_switch_func.h

DefinitionFile: tilt_tele_switch_func_RAM.c

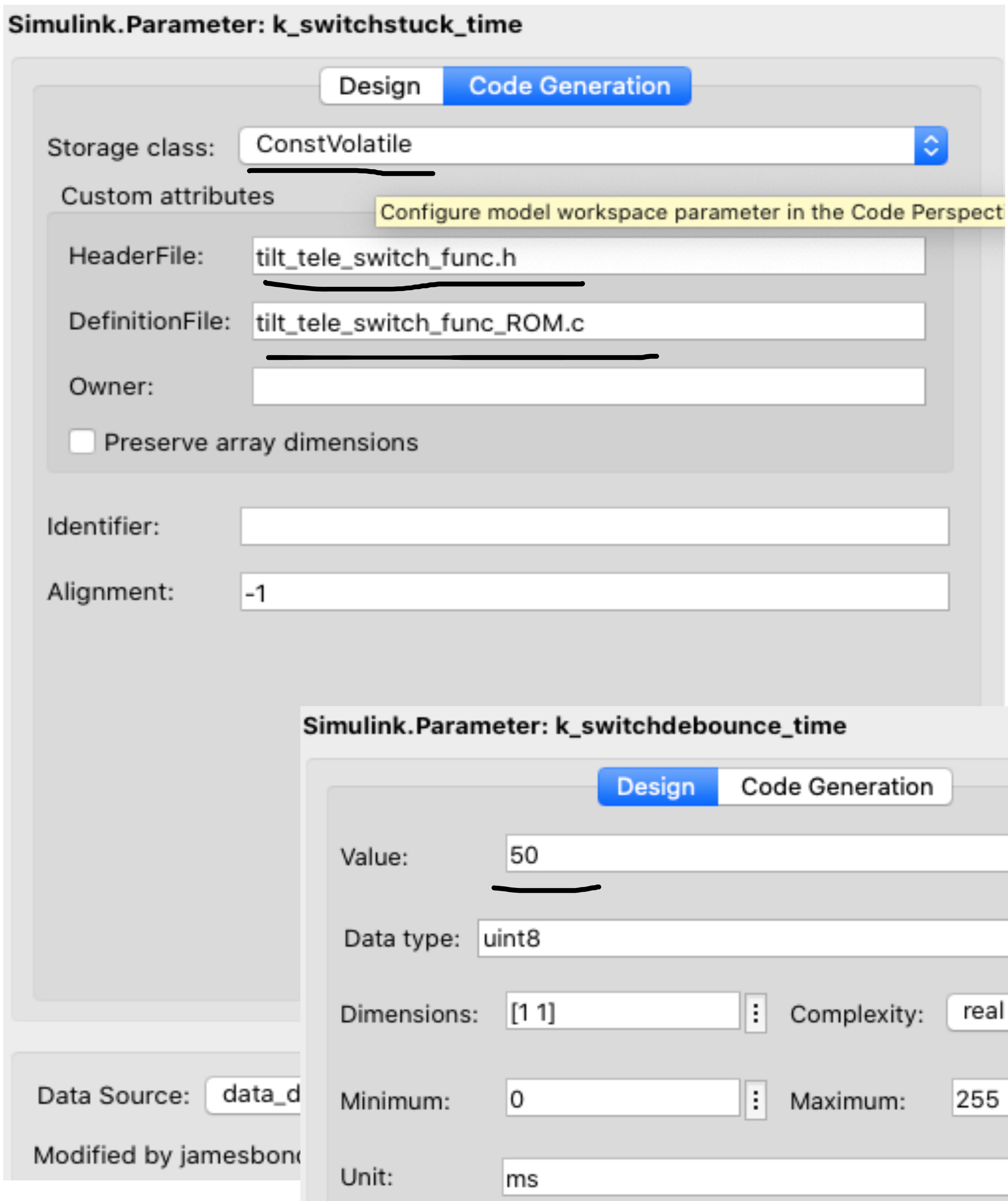
Owner:

☐ Preserve array dimensions

Identifier:

Alignment: -1

- The output signals are come under the *ExportToFile* storage class which means the signal data has to be exported to mentioned header and c files.


Simulink.Parameter: k_switchstuck_time

Storage class: ConstVolatile

Custom attributes

Configure model workspace parameter in the Code Perspect

HeaderFile: tilt_tele_switch_func.h

DefinitionFile: tilt_tele_switch_func_ROM.c

Owner:

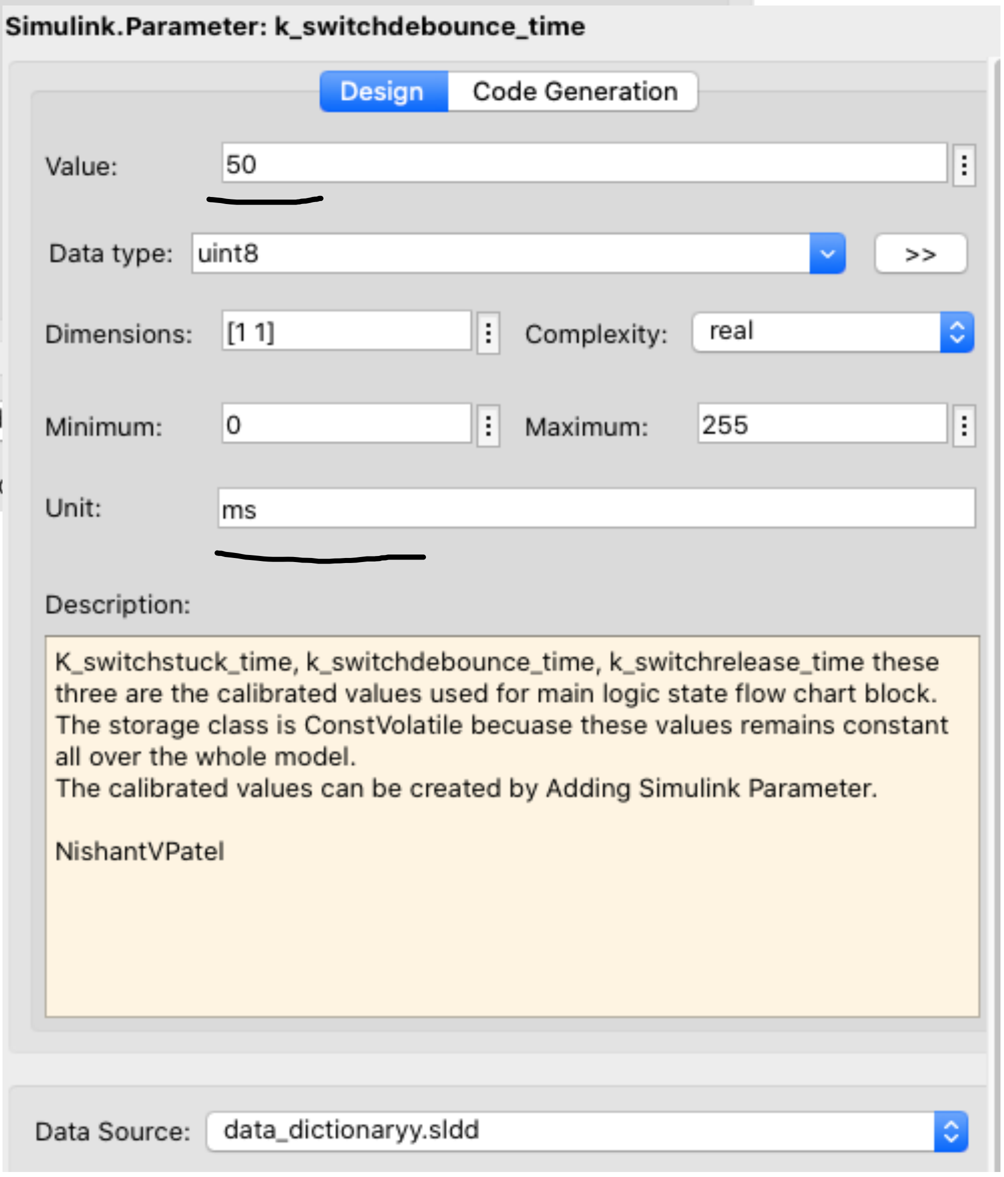
Preserve array dimensions

Identifier:

Alignment: -1

Data Source: data_d

Modified by jamesbon


Simulink.Parameter: k_switchdebounce_time

Value: 50

Data type: uint8

Dimensions: [1 1]

Complexity: real

Minimum: 0

Maximum: 255

Unit: ms

Description:

K_switchstuck_time, k_switchdebounce_time, k_switchrelease_time these three are the calibrated values used for main logic state flow chart block.
The storage class is ConstVolatile becuase these values remains constant all over the whole model.
The calibrated values can be created by Adding Simulink Parameter.

NishantVPatel

Data Source: data_dictionaryy.sldd

Requirement Traceability Report

- Requirement traceability report simple show the combinations of .slreqx and .slmx files.
- It show the subsystem attahced to which requirements and their descriptions as well.
- This all can be done by the ***Requirement Editor*** app.

Requirements Report for Tilt_Telescopic_Switch

Table of Contents

List of Tables

Chapter 1. Model Information for "Tilt_Telescopic_Switch"

Table 1.1. Tilt_Telescopic_Switch

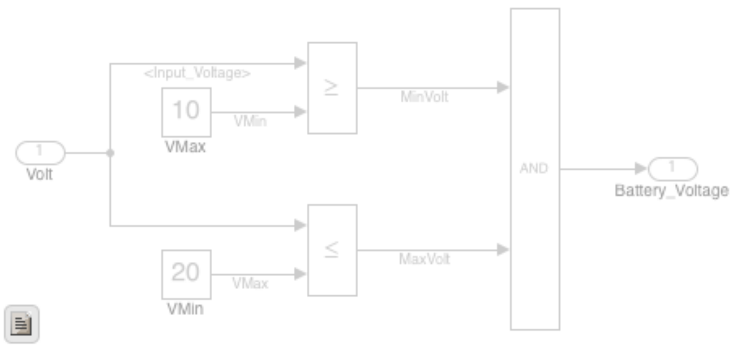
ModelVersion	1.46	*ConfigurationManager*	N/A
Created	Thu Mar 09 17:03:35 2023	*Creator*	jamesbond
LastModifiedDate	Wed Mar 15 01:22:35 2023	*LastModifiedBy*	jamesbond

Chapter 2. Traceability Summary for "Tilt_Telescopic_Switch"

Table 2.1. Artifacts linked in model

ID	Artifact names stored by RMI	Last modified	# links
DOC1	tile_tele_req_analysis.slreqx	Wed Mar 15 01:31:16 2023	3

Chapter 3. System - Valid_Voltage



Show in Simulink

Table 3.1. Tilt_Telescopic_Switch/Tilt_Tele_Switch/Valid_Voltage Requirements Data

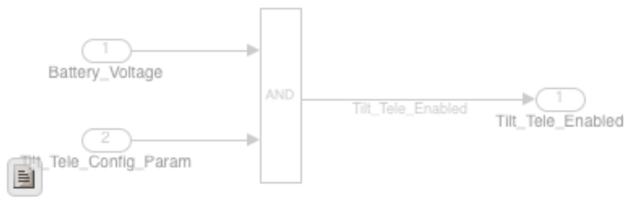
Link#	Link Description	Link Target (document name and location ID)

Link#	Link Description	Link Target (document name and location ID)
1.	"Valid Voltage" ------- Details from tile_tele_req_analysis.slreqx: ------- Description: The operating voltage range should be between 10 to 20 V.	DOC1, at "1"

Table 3.2. Objects in "Valid_Voltage" that are not linked to requirements

Name	Type
Battery_Voltage	Outport
Logical Operator	Logic
Relational Operator	RelationalOperator
Relational Operator1	RelationalOperator
VMax	Constant
VMin	Constant
Volt	Inport

Chapter 4. System - Tilt_Tele_Enabled_Condition



Show in Simulink

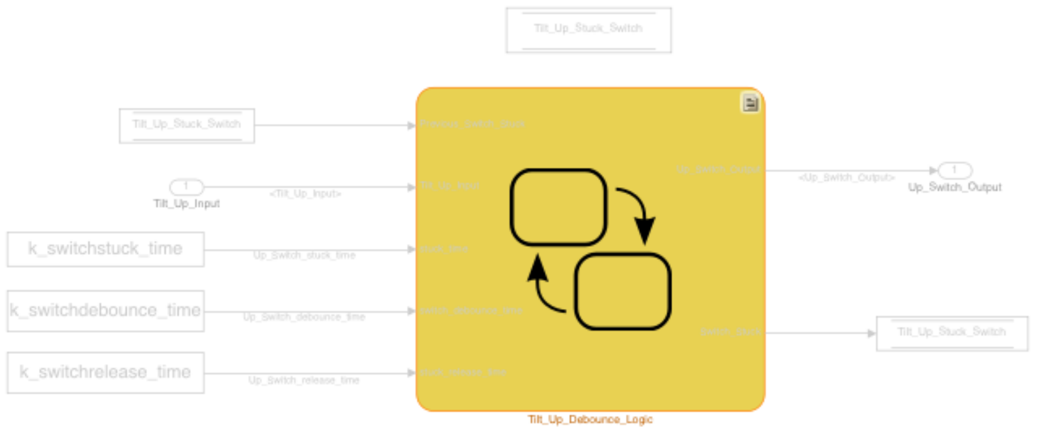
Table 4.1. Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/Tilt_Tele_Enabled_Condition Requirements Data

Link#	Link Description	Link Target (document name and location ID)
1.	"System Enabled Condition" ------- Details from tile_tele_req_analysis.slreqx: ------- Description: The valid voltage and configuration parameters which came from ECU through CAN signlas, If these both 2 condition are true then only the system will be enabled. Those configuration parameters will be like power on the voltage drop for few milli seconds.	DOC1, at "7"

Table 4.2. Objects in "Tilt_Tele_Enabled_Condition" that are not linked to requirements

Name	Type
Battery_Voltage	Inport
Logical Operator	Logic
Tilt_Tele_Config_Param	Inport
Tilt_Tele_Enabled	Outport

Chapter 5. System - Up



Show in Simulink

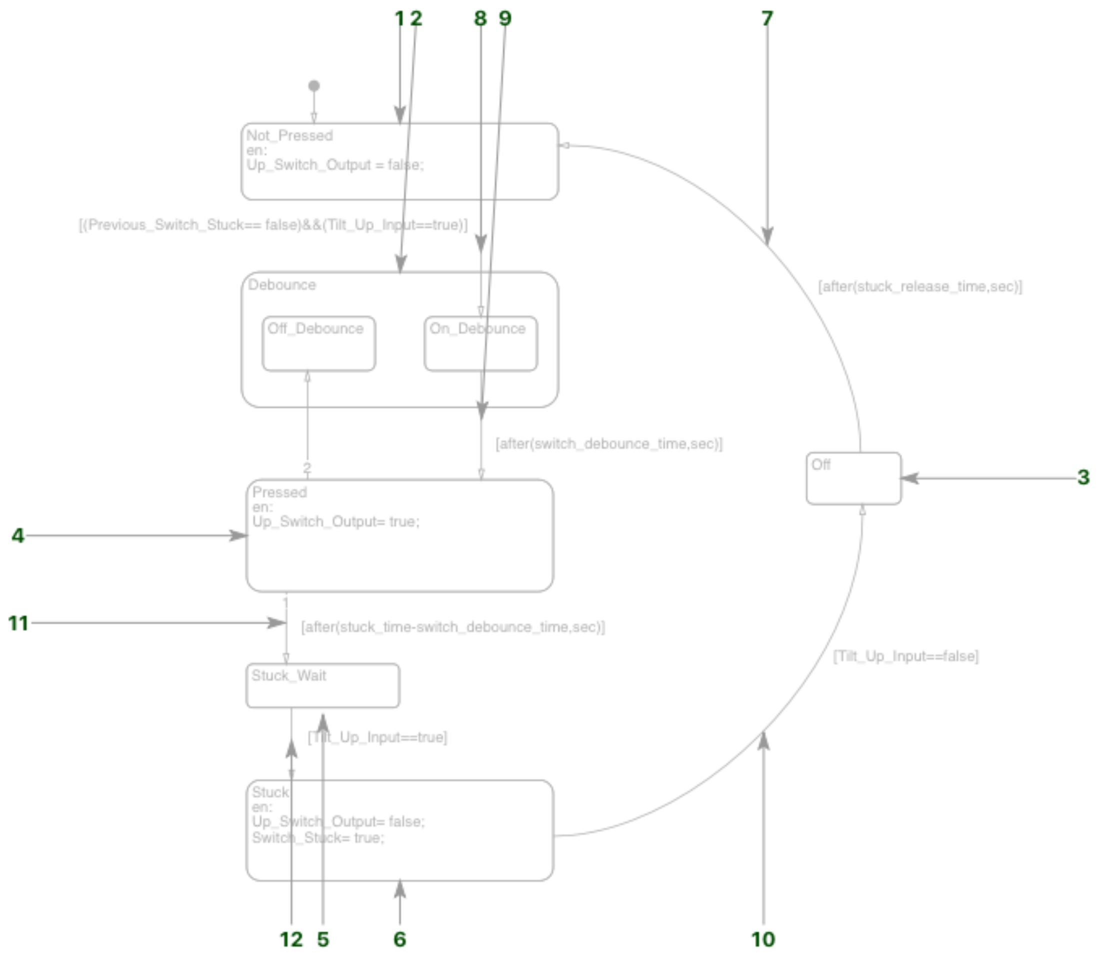
Table 5.1. Objects in Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/Up that have Requirement Links

Linked Object	Requirements Data		
Tilt_Up_Debounce_Logic	1.	"Debounce Logic" ------- Details from tile_tele_req_analysis.slreqx: ------- Description: This is the most important part of the whole model development as it depicts the main working flow of switch and is in with ECU working coordinations. This logic flowchart can be modelled in Simulink with the help of State-flow. Once the ignition is ON then the ECU should ensure all the switches are in the unstuck position Stuck is a hardware issue, this will block the desired signal being executed. There are chances for the execution of faulty signals during this stuck position. In order to avoid these errors debounce time is being introduced. This will avoid/fix all the input errors and the actual signal will be processed.	DOC1, at "6"

Table 5.2. Objects in "Up" that are not linked to requirements

Name	Type
Constant	Constant
Constant1	Constant
Constant2	Constant
Data Store Memory	DataStoreMemory
Data Store Read	DataStoreRead
Data Store Write	DataStoreWrite
Tilt_Up_Input	Inport
Up_Switch_Output	Outport

Chapter 6. Chart - Tilt_Up_Debounce_Logic



(1) Not_Pressed
(2) Debounce
(3) Off
(4) Pressed
(5) Stuck_Wait
(6) Stuck
(7) [after(stuck_release_time,sec)]
(8) [(Previous_Switch_Stuck== false)&&(Tilt_Up_Input==true)]
(9) [after(switch_debounce_time,sec)]
(10) [Tilt_Up_Input==false]
(11) [after(stuck_time-switch_debounce_time,sec)]
(12) [Tilt_Up_Input==true]

Table 6.1. Chart Requirements

Link#	Link Description	Link Target (document name and location ID)
1.	"Debounce Logic" ------- Details from tile_tele_req_analysis.slreqx: ------- Description: This is the most important part of the whole model development as it depicts the main working flow of switch and is in with ECU working coordinations. This logic flowchart can be modelled in Simulink with the help of State-flow. Once the ignition is ON then the ECU should ensure all the switches are in the unstuck position Stuck is a hardware issue, this will block the desired signal being executed. There are chances for the execution of faulty signals during this stuck position. In order to avoid these errors debounce time is being introduced. This will avoid/fix all the input errors and the actual signal will be processed.	DOC1, at "6"

Show in Simulink

Table 6.2. Objects in "Tilt_Up_Debounce_Logic" that are not linked to requirements

Name	Type
Not_Pressed	State
Debounce	State
Off	State
Pressed	State
Stuck_Wait	State
Stuck	State
Default Transition 2	Transition
[after(stuck_release_time,sec)]	Transition
[(Previous_Switch_Stuck== false)&&(Tilt_Up_Input==true)]	Transition
Default Transition 19	Transition
[after(switch_debounce_time,sec)]	Transition
[Tilt_Up_Input==false]	Transition
[after(stuck_time-switch_debounce_time,sec)]	Transition
[Tilt_Up_Input==true]	Transition
Off_Debounce	State
On_Debounce	State

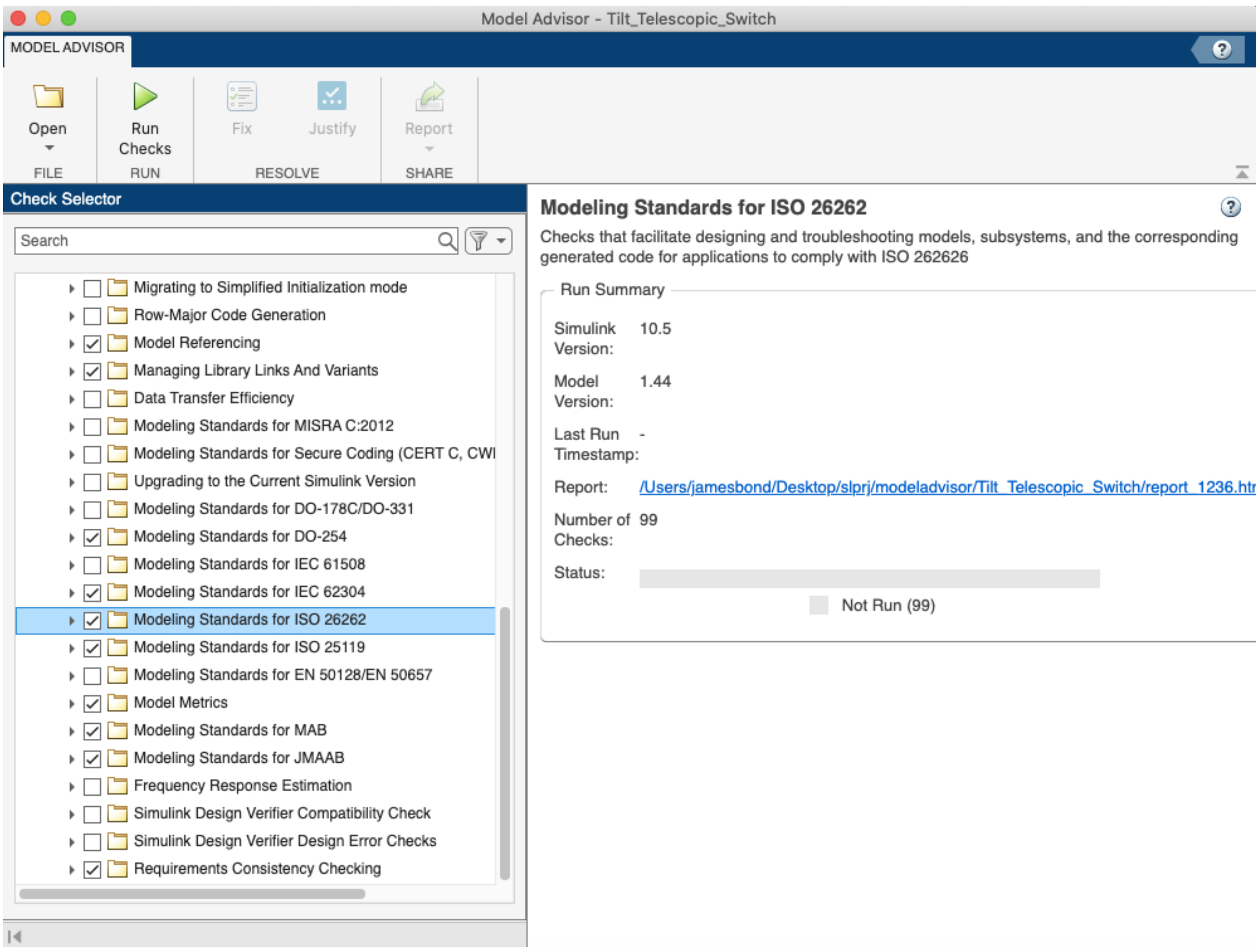
Chapter 7. Systems in "Tilt_Telescopic_Switch" that have no links to requirements

Table 7.1. Systems and subsystem blocks in "Tilt_Telescopic_Switch" that have no links to requirements

Model or subsystem block	Children with links
Tilt_Telescopic_Switch	None
Tilt_Telescopic_Switch/Tilt_Tele_Switch	1 out of 12
Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele	1 out of 12
Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination	None
Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/Down	None
Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/Forward	None
Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/Rearward	None
Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/Tele_Forward_Status	None
Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/Tele_Rearward_Status	None
Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/Tilt_Down_Status	None
Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/Tilt_Up_Status	None
Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/Up	1 out of 9
Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/Tele_Forward_Status/Compare To Constant	None
Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/Tele_Rearward_Status/Compare To Constant	None
Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/Tilt_Down_Status/Compare To Constant	None
Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/Tilt_Up_Status/Compare To Constant	None

Model Advisor

- Model advisor is used to check and inspect and the different error and warning exist in your developed model.
- It also helps to verify that does this developed model satisfy the different ISO standards like ISO 26262, and guidelines like JMAAB (Japanese MAAB), MAAB (Mathworks Automotive Advisory Board)
- Model Advisor app significatnly helps to change the modify the configuration settings for particular model betterment.



- ☐ Migrating to Simplified Initialization mode
- ☐ Row-Major Code Generation
- ☑ Model Referencing
- ☑ Managing Library Links And Variants
- ☐ Data Transfer Efficiency
- ☐ Modeling Standards for MISRA C:2012
- ☐ Modeling Standards for Secure Coding (CERT C, CWI
- ☐ Upgrading to the Current Simulink Version
- ☐ Modeling Standards for DO-178C/DO-331
- ☑ Modeling Standards for DO-254
- ☐ Modeling Standards for IEC 61508
- ☑ Modeling Standards for IEC 62304
- ☑ Modeling Standards for ISO 26262
- ☑ Modeling Standards for ISO 25119
- ☐ Modeling Standards for EN 50128/EN 50657
- ☑ Model Metrics
- ☑ Modeling Standards for MAB
- ☑ Modeling Standards for JMAAB
- ☐ Frequency Response Estimation
- ☐ Simulink Design Verifier Compatibility Check
- ☐ Simulink Design Verifier Design Error Checks
- ☑ Requirements Consistency Checking

Modeling Standards for ISO 26262

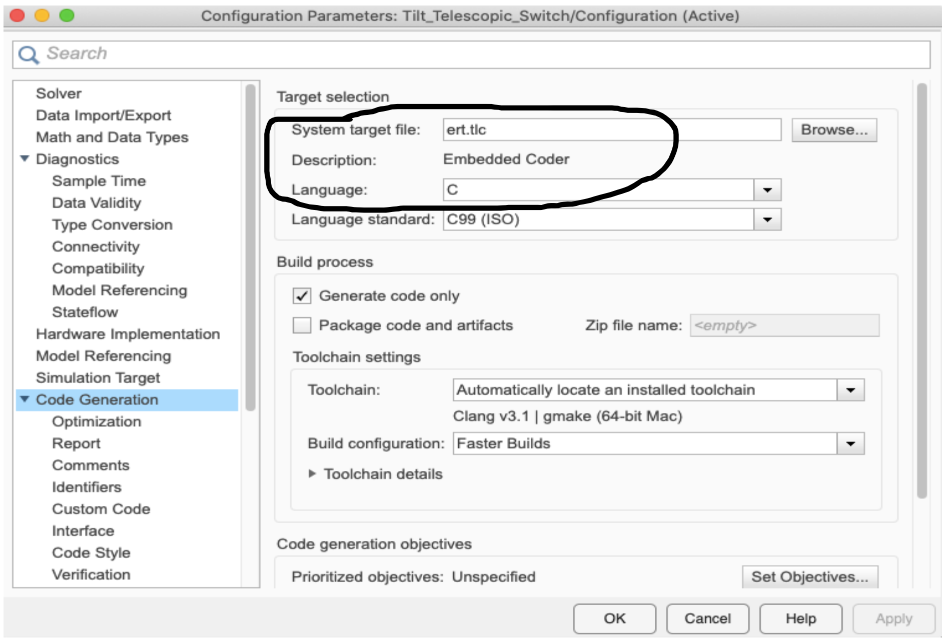
Checks that facilitate designing and troubleshooting models, subsystems, and the corresponding generated code for applications to comply with ISO 262626

Run Summary

Simulink Version:	10.5
Model Version:	1.44
Last Run Timestamp:	-
Report:	/Users/jamesbond/Desktop/slprj/modeladvisor/Tilt_Telescopic_Switch/report_1236.htr
Number of Checks:	99
Status:	

Not Run (99)

Model Configuration Settings


Configuration Parameters: Tilt_Telescopic_Switch/Configuration (Active)

Code Generation

Target selection

- System target file: ert.tlc
- Description: Embedded Coder
- Language: C
- Language standard: C99 (ISO)

Build process

- [✓] Generate code only
- [ ] Package code and artifacts Zip file name: <empty>

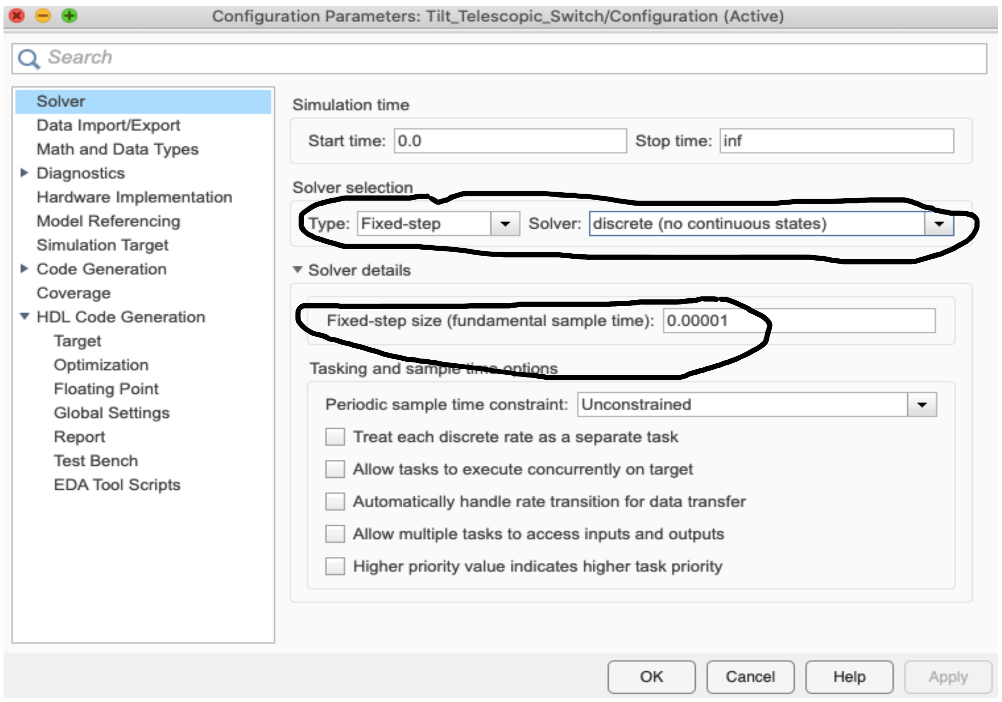
Toolchain settings

- Toolchain: Automatically locate an installed toolchain
- Clang v3.1 | gmake (64-bit Mac)
- Build configuration: Faster Builds

Code generation objectives

- Prioritized objectives: Unspecified

- Embedded code or AUTOSAR code is ideal for ECU. Here I choose embedded coder for my model.


Configuration Parameters: Tilt_Telescopic_Switch/Configuration (Active)

Solver

Simulation time

- Start time: 0.0
- Stop time: inf

Solver selection

- Type: Fixed-step
- Solver: discrete (no continuous states)

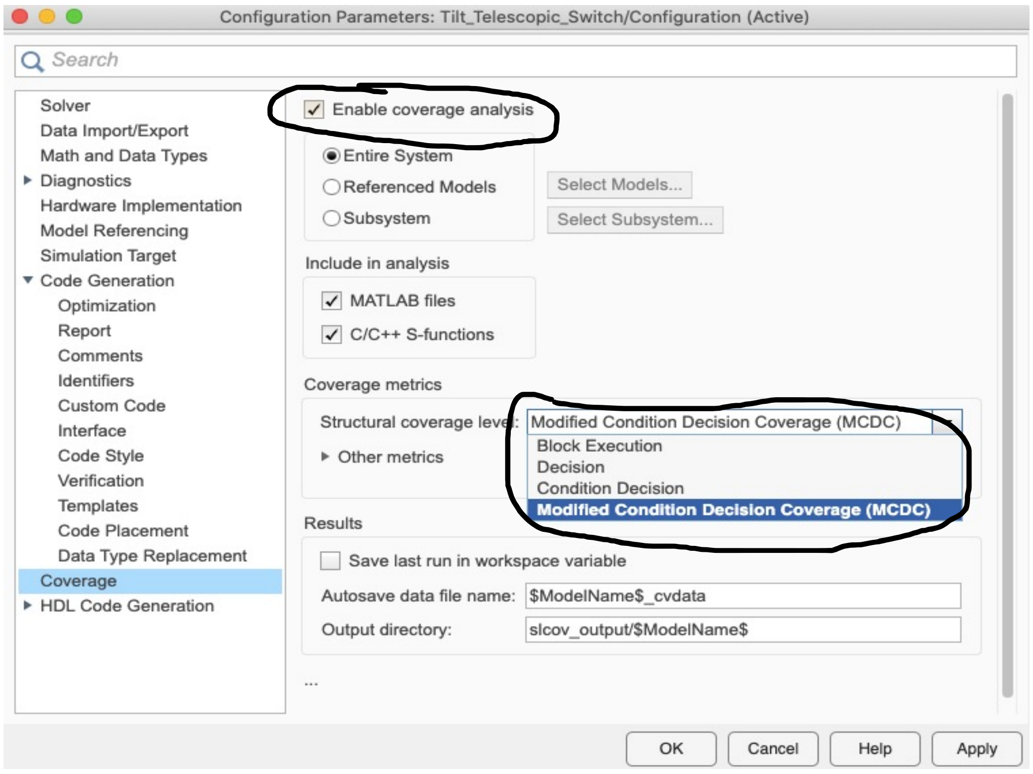
Solver details

- Fixed-step size (fundamental sample time): 0.00001

Tasking and sample time options

- Periodic sample time constraint: Unconstrained
- [ ] Treat each discrete rate as a separate task
- [ ] Allow tasks to execute concurrently on target
- [ ] Automatically handle rate transition for data transfer
- [ ] Allow multiple tasks to access inputs and outputs
- [ ] Higher priority value indicates higher task priority

- Solver type should be *Fixed-step* for code generation purpose.
- Solver is *Discrete (no continouous states)*



Configuration Parameters: Tilt_Telescopic_Switch/Configuration (Active)

Search

- Solver
- Data Import/Export
- Math and Data Types
- Diagnostics
- Hardware Implementation
- Model Referencing
- Simulation Target
- Code Generation
  - Optimization
  - Report
  - Comments
  - Identifiers
  - Custom Code
  - Interface
  - Code Style
  - Verification
  - Templates
  - Code Placement
  - Data Type Replacement
- Coverage
- HDL Code Generation

☑ Enable coverage analysis

● Entire System
○ Referenced Models — Select Models...
○ Subsystem — Select Subsystem...

Include in analysis

☑ MATLAB files
☑ C/C++ S-functions

Coverage metrics

Structural coverage level: Modified Condition Decision Coverage (MCDC)
- Block Execution
- Decision
- Condition Decision
- **Modified Condition Decision Coverage (MCDC)**

Other metrics

Results

☐ Save last run in workspace variable

Autosave data file name: $ModelName$_cvdata

Output directory: slcov_output/$ModelName$

...

- The coverage analysis emphasizes on the given structural level that how much percentage of coverage is possible and occurred in the present developed model. The logical operators, relational operators, switch block are the examples which plays an important role during coverage analysis.
- On genral basis, *MCDC* option is good to go for analysis.

5. Code Generation Report

NOTE

- This Code Generation Report is a C-language codings for given model with header and c-files.
- The code generation report includes number of different files.
- This code is generated based on Embedded Coder
- Here I only show main file (ert.tlc) which is simulink target file, model files (.c and .h files) so in total 3 files.

ert.tlc (simulink target file/ main file)

Tilt_Tele_Steering_Switch.c (c file/ model file)

Tilt_Tele_Steering_Switch.h (header file/ model file)

Code Generation Report for 'Tilt_Telescopic_Switch'

Model Information

Author	jamesbond
Last Modified By	jamesbond
Model Version	1.43
Tasking Mode	SingleTasking

Configuration settings at time of code generation

Code Information

System Target File	ert.tlc
Hardware Device Type	Intel->x86-64 (Windows64)
Simulink Coder Version	9.7 (R2022a) 13-Nov-2021
Timestamp of Generated Source Code	Sun Mar 12 03:15:13 2023
Location of Generated Source Code	/Users/jamesbond/Desktop/MATLAB: Simulink examples/Tilt_Telescopic_Switch_ert_rtw
Type of Build	Model
Objectives Specified	**Unspecified**

Additional Information

Code Generation Advisor	Not run

ert_main.c

```c
/*

 * Academic License - for use in teaching, academic research, and meeting

 * course requirements at degree granting institutions only.  Not for

 * government, commercial, or other organizational use.

 *

 * File: ert_main.c

 *

 * Code generated for Simulink model 'Tilt_Telescopic_Switch'.

 *

 * Model version                  : 1.46

 * Simulink Coder version         : 9.7 (R2022a) 13-Nov-2021

 * C/C++ source code generated on : Tue Mar 21 15:30:45 2023

 *

 * Target selection: ert.tlc

 * Embedded hardware selection: Intel->x86-64 (Windows64)

 * Code generation objectives: Unspecified

 * Validation result: Not run

 */


#include <stddef.h>

#include <stdio.h>            /* This example main program uses printf/fflush */

#include "Tilt_Telescopic_Switch.h"    /* Model header file */


/*

 * Associating rt_OneStep with a real-time clock or interrupt service routine

 * is what makes the generated code "real-time".  The function rt_OneStep is

 * always associated with the base rate of the model.  Subrates are managed

 * by the base rate from inside the generated code.  Enabling/disabling

 * interrupts and floating point context switches are target specific.  This

 * example code indicates where these should take place relative to executing

 * the generated code step function.  Overrun behavior should be tailored to

 * your application needs.  This example simply sets an error status in the

 * real-time model and returns from rt_OneStep.

 */
```

```
void rt_OneStep(void);
void rt_OneStep(void)
{
  static boolean_T OverrunFlag = false;

  /* Disable interrupts here */

  /* Check for overrun */
  if (OverrunFlag) {
    return;
  }

  OverrunFlag = true;

  /* Save FPU context here (if necessary) */
  /* Re-enable timer or interrupt here */
  /* Set model inputs here */

  /* Step the model */
  Tilt_Telescopic_Switch_step();

  /* Get model outputs here */

  /* Indicate task complete */
  OverrunFlag = false;

  /* Disable interrupts here */
  /* Restore FPU context here (if necessary) */
  /* Enable interrupts here */
}

/*
 * The example main function illustrates what is required by your
 * application code to initialize, execute, and terminate the generated code.
 * Attaching rt_OneStep to a real-time clock is target specific. This example
 * illustrates how you do this relative to initializing the model.
 */
int_T main(int_T argc, const char *argv[])
{
  /* Unused arguments */
  (void)(argc);
  (void)(argv);

  /* Initialize model */
  Tilt_Telescopic_Switch_initialize();

  /* Attach rt_OneStep to a timer or interrupt service routine with
   * period 1.0E-5 seconds (base rate of the model) here.
   * The call syntax for rt_OneStep is
   *
   *  rt_OneStep();
   */
  printf("Warning: The simulation will run forever. "
         "Generated ERT main won't simulate model step behavior. "
         "To change this behavior select the 'MAT-file logging' option.\n");
  fflush((NULL));
  while (1) {
    /*  Perform application tasks here */
  }

  /* The option 'Remove error status field in real-time model data structure'
   * is selected, therefore the following code does not need to execute.
   */

  /* Terminate model */
  Tilt_Telescopic_Switch_terminate();
  return 0;
}

/*
```

```
* File trailer for generated code.
*
* [EOF]
*/
```

```
        /* Entry 'Not_Pressed': '<S21>:1' */
        Tilt_Telescopic_Switch_B.Up_Switch_Output = false;
      } else {
        switch (Tilt_Telescopic_Switch_DW.is_c3_Tilt_Telescopic_Switch) {
         case Tilt_Telescopic_Swi_IN_Debounce:
          /* During 'Debounce': '<S21>:3' */
          if ((Tilt_Telescopic_Switch_DW.is_Debounce !=
               Tilt_Telescopic_IN_Off_Debounce) &&
              (Tilt_Telescopic_Switch_DW.temporalCounter_i1 >= ((uint32_T)
                (k_switchdebounce_time * 100)))) {
            /* During 'On_Debounce': '<S21>:4' */
            /* Transition: '<S21>:11' */
            Tilt_Telescopic_Switch_DW.is_Debounce =
              Tilt_Telesco_IN_NO_ACTIVE_CHILD;
            Tilt_Telescopic_Switch_DW.is_c3_Tilt_Telescopic_Switch =
              Tilt_Telescopic_Swit_IN_Pressed;
            Tilt_Telescopic_Switch_DW.temporalCounter_i1 = 0U;

            /* Entry 'Pressed': '<S21>:6' */
            Tilt_Telescopic_Switch_B.Up_Switch_Output = true;
          } else {
            /* During 'Off_Debounce': '<S21>:5' */
          }
          break;

         case Tilt_Telescopic__IN_Not_Pressed:
          Tilt_Telescopic_Switch_B.Up_Switch_Output = false;

          /* During 'Not_Pressed': '<S21>:1' */
          if ((!Tilt_Up_Stuck_Switch) && Tilt_Up_Input) {
            /* Transition: '<S21>:10' */
            Tilt_Telescopic_Switch_DW.is_c3_Tilt_Telescopic_Switch =
              Tilt_Telescopic_Swi_IN_Debounce;
```

```c
      Tilt_Telescopic_Switch_DW.is_Debounce =
        Tilt_Telescopic__IN_On_Debounce;
      Tilt_Telescopic_Switch_DW.temporalCounter_i1 = 0U;
    }
    break;

   case Tilt_Telescopic_Switch_IN_Off:
    /* During 'Off': '<S21>:9' */
    if (Tilt_Telescopic_Switch_DW.temporalCounter_i1 >= ((uint32_T)
          (k_switchrelease_time * 100))) {
      /* Transition: '<S21>:17' */
      Tilt_Telescopic_Switch_DW.is_c3_Tilt_Telescopic_Switch =
        Tilt_Telescopic__IN_Not_Pressed;

      /* Entry 'Not_Pressed': '<S21>:1' */
      Tilt_Telescopic_Switch_B.Up_Switch_Output = false;
    }
    break;

   case Tilt_Telescopic_Swit_IN_Pressed:
    Tilt_Telescopic_Switch_B.Up_Switch_Output = true;

    /* During 'Pressed': '<S21>:6' */
    tmp = (k_switchstuck_time - k_switchdebounce_time) * 100;
    if (tmp < 0) {
      tmp = 0;
    }

    if (Tilt_Telescopic_Switch_DW.temporalCounter_i1 >= ((uint32_T)tmp)) {
      /* Transition: '<S21>:18' */
      Tilt_Telescopic_Switch_DW.is_c3_Tilt_Telescopic_Switch =
        Tilt_Telescopic_S_IN_Stuck_Wait;
    } else {
      /* Transition: '<S21>:19' */
      Tilt_Telescopic_Switch_DW.is_c3_Tilt_Telescopic_Switch =
        Tilt_Telescopic_Swi_IN_Debounce;
      Tilt_Telescopic_Switch_DW.is_Debounce =
        Tilt_Telescopic_IN_Off_Debounce;
    }
    break;

   case Tilt_Telescopic_Switch_IN_Stuck:
    Tilt_Telescopic_Switch_B.Up_Switch_Output = false;
    Tilt_Telescopic_Switch_B.Switch_Stuck = true;

    /* During 'Stuck': '<S21>:8' */
    if (!Tilt_Up_Input) {
      /* Transition: '<S21>:16' */
      Tilt_Telescopic_Switch_DW.is_c3_Tilt_Telescopic_Switch =
        Tilt_Telescopic_Switch_IN_Off;
      Tilt_Telescopic_Switch_DW.temporalCounter_i1 = 0U;
    }
    break;

   default:
    /* During 'Stuck_Wait': '<S21>:7' */
    if (Tilt_Up_Input) {
      /* Transition: '<S21>:15' */
      Tilt_Telescopic_Switch_DW.is_c3_Tilt_Telescopic_Switch =
        Tilt_Telescopic_Switch_IN_Stuck;

      /* Entry 'Stuck': '<S21>:8' */
      Tilt_Telescopic_Switch_B.Up_Switch_Output = false;
      Tilt_Telescopic_Switch_B.Switch_Stuck = true;
    }
    break;
  }
}

/* End of Chart: '<S13>/Tilt_Up_Debounce_Logic' */
/* DataStoreWrite: '<S13>/Data Store Write' */
Tilt_Up_Stuck_Switch = Tilt_Telescopic_Switch_B.Switch_Stuck;

/* Switch: '<S9>/Switch' incorporates:
 *  Constant: '<S17>/Constant'
 *  RelationalOperator: '<S17>/Compare'
 *  Sum: '<S4>/Add'
 *  Switch: '<S10>/Switch'
 *  Switch: '<S11>/Switch'
 *  Switch: '<S12>/Switch'
 *  Switch: '<S9>/Switch1'
 */
if (((int32_T)(((((uint32_T)Tilt_Telescopic_Switch_B.Up_Switch_Output) +
                 Tilt_Telescopic_Switch_B.Down_Switch_Output) +
                Tilt_Telescopic_Switch_B.Forward_Switch_Output) +
               Tilt_Telescopic_Switch_B.Rearward_Switch_Output)) >
    ((uint8_T)1U)) {
  /* Switch: '<S9>/Switch' incorporates:
   *  UnitDelay: '<S9>/Unit Delay'
   */
  Tele_Forward_Switch_Status = Tele_Forward_Switch_Status_Delay;

  /* Switch: '<S10>/Switch' incorporates:
   *  UnitDelay: '<S10>/Unit Delay'
   */
  Tele_Rearward_Switch_Status = Tele_Rearward_Switch_Status_Delay;

  /* Switch: '<S11>/Switch' incorporates:
   *  UnitDelay: '<S11>/Unit Delay'
   */
  Tilt_Down_Switch_Status = Tilt_Down_Switch_Status_Delay;

  /* Switch: '<S12>/Switch' incorporates:
   *  UnitDelay: '<S12>/Unit Delay'
```

```
     */
    Tilt_Up_Switch_Status = Tilt_Up_Switch_Status_Delay;
  } else {
    if (Tilt_Telescopic_Switch_B.Forward_Switch_Output) {
      /* Switch: '<S9>/Switch1' incorporates:
       *  Constant: '<S9>/Forward'
       *  Switch: '<S9>/Switch'
       */
      Tele_Forward_Switch_Status = TELE_Forward;
    } else {
      /* Switch: '<S9>/Switch' incorporates:
       *  Constant: '<S9>/Off'
       *  Switch: '<S9>/Switch1'
       */
      Tele_Forward_Switch_Status = TELE_Off;
    }

    /* Switch: '<S10>/Switch1' */
    if (Tilt_Telescopic_Switch_B.Rearward_Switch_Output) {
      /* Switch: '<S10>/Switch' incorporates:
       *  Constant: '<S10>/Rearward'
       */
      Tele_Rearward_Switch_Status = TELE_Rearward;
    } else {
      /* Switch: '<S10>/Switch' incorporates:
       *  Constant: '<S10>/Off'
       */
      Tele_Rearward_Switch_Status = TELE_Off;
    }

    /* End of Switch: '<S10>/Switch1' */

    /* Switch: '<S11>/Switch1' */
    if (Tilt_Telescopic_Switch_B.Down_Switch_Output) {
      /* Switch: '<S11>/Switch' incorporates:
```

```
     *  Constant: '<S11>/Down'
     */
    Tilt_Down_Switch_Status = TILT_Down;
  } else {
    /* Switch: '<S11>/Switch' incorporates:
     *  Constant: '<S11>/Off'
     */
    Tilt_Down_Switch_Status = TILT_Off;
  }

  /* End of Switch: '<S11>/Switch1' */

  /* Switch: '<S12>/Switch1' */
  if (Tilt_Telescopic_Switch_B.Up_Switch_Output) {
    /* Switch: '<S12>/Switch' incorporates:
     *  Constant: '<S12>/Up'
     */
    Tilt_Up_Switch_Status = TILT_Up;
  } else {
    /* Switch: '<S12>/Switch' incorporates:
     *  Constant: '<S12>/Off'
     */
    Tilt_Up_Switch_Status = TILT_Off;
  }

  /* End of Switch: '<S12>/Switch1' */
}

/* End of Switch: '<S9>/Switch' */

/* Update for UnitDelay: '<S9>/Unit Delay' */
Tele_Forward_Switch_Status_Delay = Tele_Forward_Switch_Status;

/* Update for UnitDelay: '<S10>/Unit Delay' */
Tele_Rearward_Switch_Status_Delay = Tele_Rearward_Switch_Status;
```

```
      /* Update for UnitDelay: '<S11>/Unit Delay' */
      Tilt_Down_Switch_Status_Delay = Tilt_Down_Switch_Status;

      /* Update for UnitDelay: '<S12>/Unit Delay' */
      Tilt_Up_Switch_Status_Delay = Tilt_Up_Switch_Status;
    }

    /* End of Logic: '<S5>/Logical Operator' */
    /* End of Outputs for SubSystem: '<S2>/SwitchPos_Status_Determination' */
  }

  rate_scheduler();
}

/* Model initialize function */
void Tilt_Telescopic_Switch_initialize(void)
{
  /* Registration code */

  /* block I/O */

  /* custom signals */
  Tilt_Down_Switch_Status = TILT_Up;
  Tilt_Up_Switch_Status = TILT_Up;
  Tele_Forward_Switch_Status = TELE_Forward;
  Tele_Rearward_Switch_Status = TELE_Forward;

  /* states (dwork) */

  /* custom states */
  Tele_Forward_Switch_Status_Delay = TELE_Forward;
  Tele_Rearward_Switch_Status_Delay = TELE_Forward;
  Tilt_Down_Switch_Status_Delay = TILT_Up;
  Tilt_Up_Switch_Status_Delay = TILT_Up;
  /* SystemInitialize for Enabled SubSystem: '<S2>/SwitchPos_Status_Determination' */
  /* InitializeConditions for UnitDelay: '<S9>/Unit Delay' */
  Tele_Forward_Switch_Status_Delay = TELE_Off;

  /* InitializeConditions for UnitDelay: '<S10>/Unit Delay' */
  Tele_Rearward_Switch_Status_Delay = TELE_Off;

  /* InitializeConditions for UnitDelay: '<S11>/Unit Delay' */
  Tilt_Down_Switch_Status_Delay = TILT_Off;

  /* InitializeConditions for UnitDelay: '<S12>/Unit Delay' */
  Tilt_Up_Switch_Status_Delay = TILT_Off;

  /* End of SystemInitialize for SubSystem: '<S2>/SwitchPos_Status_Determination' */
}

/* Model terminate function */
void Tilt_Telescopic_Switch_terminate(void)
{
  /* (no terminate code required) */
}

/*
 * File trailer for generated code.
 *
 * [EOF]
 */
```
/

Tilt_Telescopic_Switch.h

```
/*

 * Academic License - for use in teaching, academic research, and meeting

 * course requirements at degree granting institutions only.  Not for

 * government, commercial, or other organizational use.

 *

 * File: Tilt_Telescopic_Switch.h

 *

 * Code generated for Simulink model 'Tilt_Telescopic_Switch'.

 *

 * Model version                  : 1.46

 * Simulink Coder version         : 9.7 (R2022a) 13-Nov-2021

 * C/C++ source code generated on : Tue Mar 21 15:30:45 2023

 *

 * Target selection: ert.tlc

 * Embedded hardware selection: Intel->x86-64 (Windows64)

 * Code generation objectives: Unspecified

 * Validation result: Not run

 */


#ifndef RTW_HEADER_Tilt_Telescopic_Switch_h_

#define RTW_HEADER_Tilt_Telescopic_Switch_h_

#ifndef Tilt_Telescopic_Switch_COMMON_INCLUDES_

#define Tilt_Telescopic_Switch_COMMON_INCLUDES_

#include "rtwtypes.h"

#endif                             /* Tilt_Telescopic_Switch_COMMON_INCLUDES_ */


#include "Tilt_Telescopic_Switch_types.h"

#include "enum_types.h"


/* Includes for objects with custom storage classes */

#include "tilt_tele_switch_func.h"


/* Block signals (default storage) */

typedef struct {
  boolean_T Up_Switch_Output;          /* '<S13>/Tilt_Up_Debounce_Logic' */
  boolean_T Switch_Stuck;              /* '<S13>/Tilt_Up_Debounce_Logic' */
  boolean_T Rearward_Switch_Output;    /* '<S8>/Tele_Rearward_Debounce_Logic' */
  boolean_T Switch_Stuck_j;            /* '<S8>/Tele_Rearward_Debounce_Logic' */
  boolean_T Forward_Switch_Output;     /* '<S7>/Tele_Forward_Debounce_Logic' */
  boolean_T Switch_Stuck_l;            /* '<S7>/Tele_Forward_Debounce_Logic' */
  boolean_T Down_Switch_Output;        /* '<S6>/Tilt_Down_Debounce_Logic' */
  boolean_T Switch_Stuck_i;            /* '<S6>/Tilt_Down_Debounce_Logic' */
} B_Tilt_Telescopic_Switch_T;

/* Block states (default storage) for system '<Root>' */
typedef struct {
  uint32_T temporalCounter_i1;         /* '<S13>/Tilt_Up_Debounce_Logic' */
  uint32_T temporalCounter_i1_d;       /* '<S8>/Tele_Rearward_Debounce_Logic' */
  uint32_T temporalCounter_i1_g;       /* '<S7>/Tele_Forward_Debounce_Logic' */
  uint32_T temporalCounter_i1_p;       /* '<S6>/Tilt_Down_Debounce_Logic' */
  uint8_T is_active_c3_Tilt_Telescopic_Sw;/* '<S13>/Tilt_Up_Debounce_Logic' */
  uint8_T is_c3_Tilt_Telescopic_Switch;/* '<S13>/Tilt_Up_Debounce_Logic' */
  uint8_T is_Debounce;                 /* '<S13>/Tilt_Up_Debounce_Logic' */
  uint8_T is_active_c4_Tilt_Telescopic_Sw;/* '<S8>/Tele_Rearward_Debounce_Logic' */
  uint8_T is_c4_Tilt_Telescopic_Switch;/* '<S8>/Tele_Rearward_Debounce_Logic' */
  uint8_T is_Debounce_o;               /* '<S8>/Tele_Rearward_Debounce_Logic' */
  uint8_T is_active_c2_Tilt_Telescopic_Sw;/* '<S7>/Tele_Forward_Debounce_Logic' */
  uint8_T is_c2_Tilt_Telescopic_Switch;/* '<S7>/Tele_Forward_Debounce_Logic' */
  uint8_T is_Debounce_f;               /* '<S7>/Tele_Forward_Debounce_Logic' */
  uint8_T is_active_c1_Tilt_Telescopic_Sw;/* '<S6>/Tilt_Down_Debounce_Logic' */
  uint8_T is_c1_Tilt_Telescopic_Switch;/* '<S6>/Tilt_Down_Debounce_Logic' */
  uint8_T is_Debounce_a;               /* '<S6>/Tilt_Down_Debounce_Logic' */
} DW_Tilt_Telescopic_Switch_T;

/* Real-time Model Data Structure */
struct tag_RTM_Tilt_Telescopic_Switc_T {
  /*
   * Timing:
   * The following substructure contains information regarding
```

```c
   * the timing information for the model.
   */
  struct {
    struct {
      uint16_T TID[2];
    } TaskCounters;
  } Timing;
};

/* Block signals (default storage) */
extern B_Tilt_Telescopic_Switch_T Tilt_Telescopic_Switch_B;

/* Block states (default storage) */
extern DW_Tilt_Telescopic_Switch_T Tilt_Telescopic_Switch_DW;

/* Model entry point functions */
extern void Tilt_Telescopic_Switch_initialize(void);
extern void Tilt_Telescopic_Switch_step(void);
extern void Tilt_Telescopic_Switch_terminate(void);

/* Real-time Model object */
extern RT_MODEL_Tilt_Telescopic_Swit_T *const Tilt_Telescopic_Switch_M;

/*-
 * The generated code includes comments that allow you to trace directly
 * back to the appropriate location in the model.  The basic format
 * is <system>/block_name, where system is the system number (uniquely
 * assigned by Simulink) and block_name is the name of the block.
 *
 * Use the MATLAB hilite_system command to trace the generated code back
 * to the model.  For example,
 *
 * hilite_system('<S3>')    - opens system 3
 * hilite_system('<S3>/Kp') - opens and selects block Kp which resides in S3
 *
 * Here is the system hierarchy for this model

 *

 * '<Root>' : 'Tilt_Telescopic_Switch'

 * '<S1>'   : 'Tilt_Telescopic_Switch/Tilt_Tele_Switch'

 * '<S2>'   : 'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele'

 * '<S3>'   : 'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Valid_Voltage'

 * '<S4>'   :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination'

 * '<S5>'   :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/Tilt_Tele_Enabled_Condition'

 * '<S6>'   :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/
Down'

 * '<S7>'   :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/
Forward'

 * '<S8>'   :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/
Rearward'

 * '<S9>'   :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/
Tele_Forward_Status'

 * '<S10>'  :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/
Tele_Rearward_Status'

 * '<S11>'  :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/
Tilt_Down_Status'

 * '<S12>'  :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/
Tilt_Up_Status'

 * '<S13>'  :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/
Up'

 * '<S14>'  :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/
Down/Tilt_Down_Debounce_Logic'

 * '<S15>'  :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/
Forward/Tele_Forward_Debounce_Logic'

 * '<S16>'  :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/
Rearward/Tele_Rearward_Debounce_Logic'

 * '<S17>'  :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/
Tele_Forward_Status/Compare To Constant'

 * '<S18>'  :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/
Tele_Rearward_Status/Compare To Constant'
```

```
 * '<S19>'  :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/
Tilt_Down_Status/Compare To Constant'

 * '<S20>'  :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/
Tilt_Up_Status/Compare To Constant'

 * '<S21>'  :
'Tilt_Telescopic_Switch/Tilt_Tele_Switch/Power_Tilt_Tele/SwitchPos_Status_Determination/
Up/Tilt_Up_Debounce_Logic'

 */

#endif                                 /* RTW_HEADER_Tilt_Telescopic_Switch_h_ */



/*

 * File trailer for generated code.

 *

 * [EOF]

 */
/
```