**Project Title: Adaptive Cruise Control**

**Aim**

To create a Simulink model of Adaptive Cruise Control as per the Requirement data.

**General Overview:**

Adaptive Cruise Control Feature for passenger cars allows the host vehicle to adapt to the speed in line with the flow of traffic. Driving in heavy traffic or keeping a safe distance to the preceding vehicle calls for a high level of concentration. The Adaptive Cruise Control feature can reduce the stress on the driver by automatically controlling the vehicle speed & maintaining a predefined minimum distance to the preceding vehicle. As a consequence, the driver enjoys more comfort & can concentrate on the road little better.

A radar sensor is usually at the core of the Adaptive Cruise Control. Installed at the front of the vehicle, the system permanently monitors the road ahead. As long as the road ahead is clear, cruise control feature maintains the speed set by the driver. If the system spots a slower vehicle within its detection range, it gently reduces speed by releasing the accelerator or actively engaging the brake control system. If the vehicle ahead speeds up or changes lanes, the cruise control automatically accelerates to the driver’s desired speed.

Standard Adaptive Cruise Control can be activated from speeds of around 30 km/h (20 mph) upwards and supports the driver, primarily on cross-country journeys or on freeways. The cruise control stop & go variant is also active at speeds below 30 km/h (20 mph). It can maintain the set distance to the preceding vehicle even at very low speeds and can decelerate to a complete standstill. When the vehicle remains stopped longer, the driver needs only to reactivate the system, for example by briefly stepping on the gas pedal to return to cruise control mode. In this way, cruise control stop & go supports the driver even in heavy traffic and traffic jams.

Since Adaptive Cruise Control is a comfort and convenience system, brake interventions and vehicle acceleration only take place within defined limits. Even with Adaptive Cruise Control switched on, it remains the driver’s responsibility to monitor the speed and distance from the vehicle in front.

**Objective of Main Project:**

* Developing Adaptive Cruise Control feature as per the Requirement Document using MATLAB Simulink.
* Follow all the MBD related processes: Requirement Tagging & Traceability, SLDD creation, Configuration Parameter changes, Model Advisor check & Code Generation.
* In Configuration Parameters: enable “Support Floating Numbers” under Code Generation settings.
* Use Embedded Coder to generate the code.
* If choosing code generation, Storage class for Input signals: ImportedExtern; Storage class for Output signal: Export to File; Storage class for local signals: localizable; Storage class for calibration signals: Const.
* Choose sample time for all signals as 0.01s

**Requirement 1– Lead Vehicle:**

* Lead Vehicle is a vehicle which is driving in the road ahead of our drive vehicle. Two input signals (Signal Name: CameraInput\_LeadVehicle & RadarInput\_LeadVehicle).
* Ideally sensor fusion techniques will be deployed to process & analyze data from camera & radar. For complexity reasons, let’s not adapt to any such algorithms.
* We can simply add both the radar & camera inputs & the corresponding output is read as Speed profile output (Signal Name: LeadVehicle\_Speed).
* Speed data of the lead vehicle is critical in implementing the Adaptive Cruise Control algorithm.

**Requirement 2 – Drive Vehicle:**

* Drive Vehicle is the vehicle driven by the user & this is the vehicle which has ACC algorithm in it.
* Like the Lead Vehicle, Drive Vehicle algorithm also has 2 input signals (Signal Name: CameraInput\_DriveVehicle, RadarInput\_DriveVehicle) & one signal coming as an Input to this subsystem (Signal Name: Acceleration\_Mode) – three inputs into this requirement in total.
* Like the above requirement, sensor fusion techniques will also be deployed here, for complexity reasons we are ignoring them.
* Two output signals come from this subsystem (Signal Name: DriveVehicle\_Speed & LeadVehicle\_Detected).
* Signal DriveVehicle\_Speed is summation of three input signals mentioned above & LeadVehicle\_Detected is renamed from Input Signal RadarInput\_DriveVehicle by mere use of Signal Conversion block.

**Requirement 3 – Adaptive Cruise Control Algorithm:**

* Adaptive Cruise Control feature has 3 major modes of operation: OFF Mode, STANDBY Mode & ON Mode. This particular requirement has to be implemented as state machine logic in Simulink.
* The input signals to this state machine system are (Signal Name: Time\_Gap, Set\_Speed, Set\_Gap, CruiseSwitch, SetSwitch).
* Also, the output signals (Signal Name: DriveVehicle\_Speed & LeadVehicle\_Detected) from requirement-2 is fed back as an input signal into this state machine block.
* Additionally, output signal (Signal Name: LeadVehicle\_Speed) from requirement-1 is given as an input signal to this state machine block as well.
* Output from this subsystem is a signal (Signal Name: Acceleration\_Mode) which governs the vehicular speed of the drive vehicle which automatically adjusts its speed & velocity to match the lead vehicle.

***Requirement 3a – ACC OFF MODE state logic:***

* This is the default state inside state machine logic. Output signal Acceleration\_Mode is at value 0 in this state.
* This state is governed by input signal CruiseSwitch.
* If CruiseSwitch is equal to 1, state ACC STANDBY mode will get activated. If CruiseSwitch is equal to 0, state ACC OFF mode will get activated, from either ACC ON mode or ACC STANDBY mode

***Requirement 3b – ACC STANDBY MODE state logic:***

* This is the second activated state inside state machine logic. Output signal Acceleration\_Mode is at value 1 in this state.
* This state is governed by both input signals CruiseSwitch & SetSwitch.
* If CruiseSwitch is equal to 1, state ACC STANDBY mode will get activated. If CruiseSwitch is equal to 0, state ACC OFF mode will get activated, from either ACC ON mode or ACC STANDBY mode
* If SetSwitch is equal to 1, state ACC ON mode will get activated. If SetSwitch is equal to 0, state ACC STANDBY mode will get activated.

***Requirement 3c – ACC ON MODE state logic:***

This state will be activated when input signal SetSwitch is equal to 1. There are 6 sub states to this state logic: They are:

* LeadVehicle\_Detected\_Follow (Default)
* LeadVehicle\_Not\_Detected
* LeadVehicle\_Detected\_Resume
* LeadVehicle\_Not\_Detected\_Resume
* LeadVehicle\_Speed\_lessthan\_Set\_Speed

***Requirement 3c (i) – LeadVehicle\_Detected\_Follow (ACC ON MODE):***

* This is the default sub state inside ACC ON MODE state. Output signal Acceleration\_Mode is equal to 2.
* Condition to transit from LeadVehicle\_Detected\_Follow to LeadVehicle\_Not\_Detected; Input signal condition LeadVehicle\_Detected == 0.
* Condition to transit from LeadVehicle\_Detected\_Follow to LeadVehicle\_Speed\_lessthan\_Set\_Speed; Input Signals condition (LeadVehicle\_Detected == 1) && (LeadVehicle\_Speed < Set\_Speed) || (Time\_Gap < Set\_Gap).

***Requirement 3c (ii) – LeadVehicle\_Not\_Detected (ACC ON MODE):***

* Output signal Acceleration\_Mode is equal to 1.
* Condition to transit from LeadVehicle\_Not\_Detected to LeadVehicle\_Detected\_Follow; Input signals condition [(LeadVehicle\_Detected==1) && (DriveVehicle\_Speed == Set\_Speed) && (LeadVehicle\_Speed >= Set\_Speed) && (Time\_Gap >= Set\_Gap)]
* Condition to transit from LeadVehicle\_Not\_Detected to LeadVehicle\_Speed\_lessthan\_Set\_Speed; Input signals condition [(LeadVehicle\_Detected == 1) && (LeadVehicle\_Speed < Set\_Speed) || (Time\_Gap < Set\_Gap)]

***Requirement 3c (iii) – LeadVehicle\_Detected\_Resume (ACC ON MODE):***

* Output signal Acceleration\_Mode is equal to 3.
* Condition to transit from LeadVehicle\_Detected\_Resume to LeadVehicle\_Detected\_Follow; Input signals condition [(DriveVehicle\_Speed == Set\_Speed) && (LeadVehicle\_Speed >= Set\_Speed) && (Time\_Gap >= Set\_Gap)]
* Condition to transit from LeadVehicle\_Detected\_Resume to LeadVehicle\_Not\_Detected\_Resume; Input signal condition LeadVehicle\_Detected==0.
* Condition to transit from LeadVehicle\_Detected\_Resume to LeadVehicle\_Speed\_equal\_Set\_Speed; Input Signal condition [(DriveVehicle\_Speed < Set\_Speed) && (LeadVehicle\_Speed > DriveVehicle\_Speed) && (Time\_Gap >= Set\_Gap)]

***Requirement 3c (iv) - LeadVehicle\_Not\_Detected\_Resume (ACC ON MODE):***

* Output signal Acceleration\_Mode is equal to 1.

***Requirement 3c (v) - LeadVehicle\_Speed\_lessthan\_Set\_Speed (ACC ON MODE):***

* Output signal Acceleration\_Mode is equal to 4.
* Condition to transit from LeadVehicle\_Speed\_lessthan\_Set\_Speed to LeadVehicle\_Not\_Detected; Input signal condition [(LeadVehicle\_Detected == 0) && (DriveVehicle\_Speed == Set\_Speed)]
* Condition to transit from LeadVehicle\_Speed\_lessthan\_Set\_Speed to LeadVehicle\_Speed\_equal\_Set\_Speed; Input signals condition [((LeadVehicle\_Speed\*1.25>=DriveVehicle\_Speed) && (LeadVehicle\_Speed \* 0.75<=DriveVehicle\_Speed)) && (DriveVehicle\_Speed < Set\_Speed) && ((Time\_Gap<=1.25\*Set\_Gap) && (Time\_Gap >=0.75\*Set\_Gap))]

***Requirement 3c (vi) - LeadVehicle\_Speed\_equal\_Set\_Speed (ACC ON MODE):***

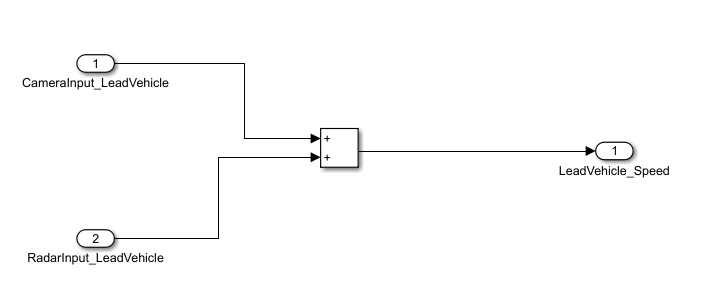
* Output signal Acceleration\_Mode is equal to 5.
* Condition to transit from LeadVehicle\_Speed\_equal\_Set\_Speed to LeadVehicle\_Not\_Detected\_Resume; Input signal conditions is [(LeadVehicle\_Detected == 0) || (DriveVehicle\_Speed <= Set\_Speed)]
* Condition to transit from LeadVehicle\_Speed\_equal\_Set\_Speed to LeadVehicle\_Detected\_Resume; Input signal conditions [(DriveVehicle\_Speed < Set\_Speed) && (LeadVehicle\_Speed > DriveVehicle\_Speed) || (Time\_Gap >= Set\_Gap)]
* Condition to transit from LeadVehicle\_Speed\_equal\_Set\_Speed to LeadVehicle\_Speed\_lessthan\_Set\_Speed; Input signals conditions [(LeadVehicle\_Speed<Set\_Speed) && (LeadVehicle\_Speed<DriveVehicle\_Speed) || (Time\_Gap==0.75\*Set\_Gap)]

**Signals & Calibration Data List:**

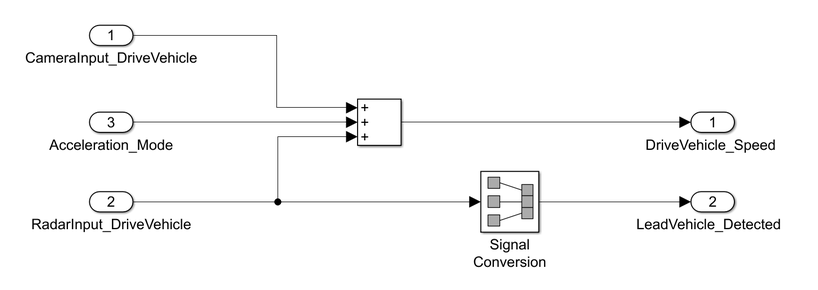
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***Signal / Calibration Name*** | ***Signal Type*** | ***Data Type*** | ***Dimension*** | ***Min*** | ***Max*** | ***Initial Value*** | ***Units*** |
| CameraInput\_LeadVehicle | Input | Uint8 | 1 | 0 | 255 | - | - |
| RadarInput\_LeadVehicle | Input | Uint8 | 1 | 0 | 255 | - | - |
| CameraInput\_DriveVehicle | Input | Uint8 | 1 | 0 | 255 | - | - |
| RadarInput\_DriveVehicle | Input | Uint8 | 1 | 0 | 255 | - | - |
| Time\_Gap | Input | Uint8 | 1 | 0 | 255 | - | - |
| Set\_Speed | Input | Uint8 | 1 | 0 | 255 | - | - |
| Set\_Gap | Input | Uint8 | 1 | 0 | 255 | - | - |
| CruiseSwitch | Input | Boolean | 1 | 0 | 1 | - | - |
| SetSwitch | Input | Boolean | 1 | 0 | 1 | - | - |
| Acceleration\_Mode | Output | Uint8 | 1 | 0 | 255 | - | - |
| LeadVehicle\_Speed | Output | Uint8 | 1 | 0 | 255 | - | - |
| DriveVehicle\_Speed | Output | Uint8 | 1 | 0 | 255 | - | - |
| LeadVehicle\_Detected | Output | Uint8 | 1 | 0 | 255 | - | - |

**Model Screenshots**

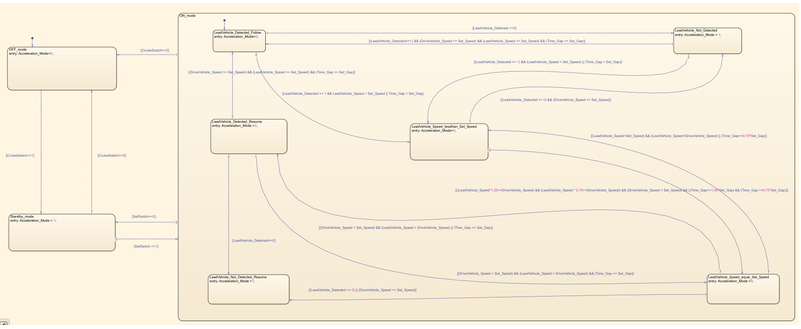
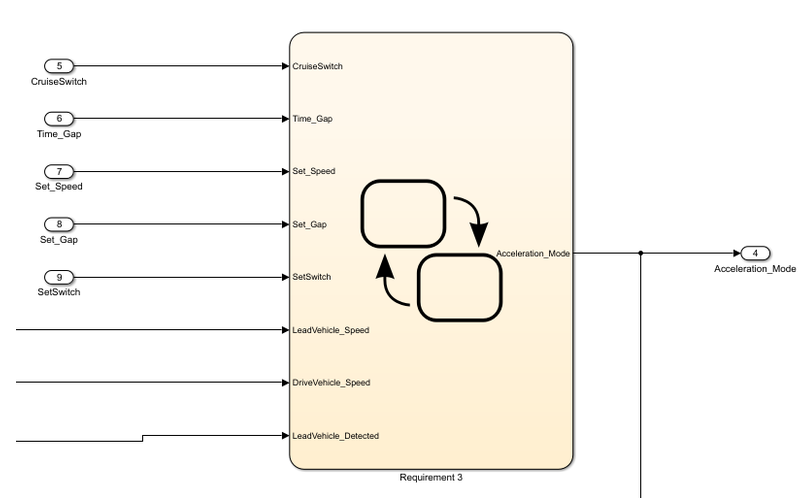
***Lead Vehicle***

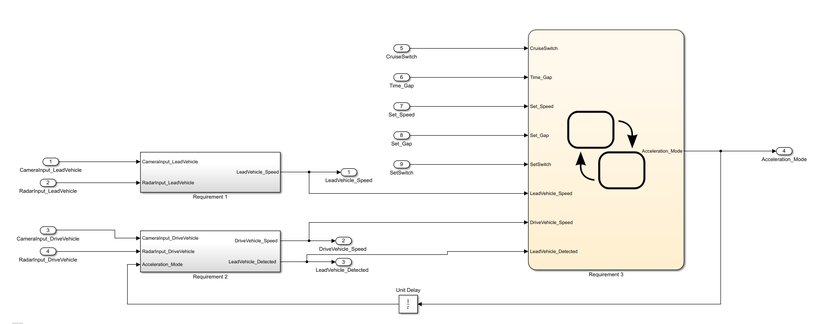


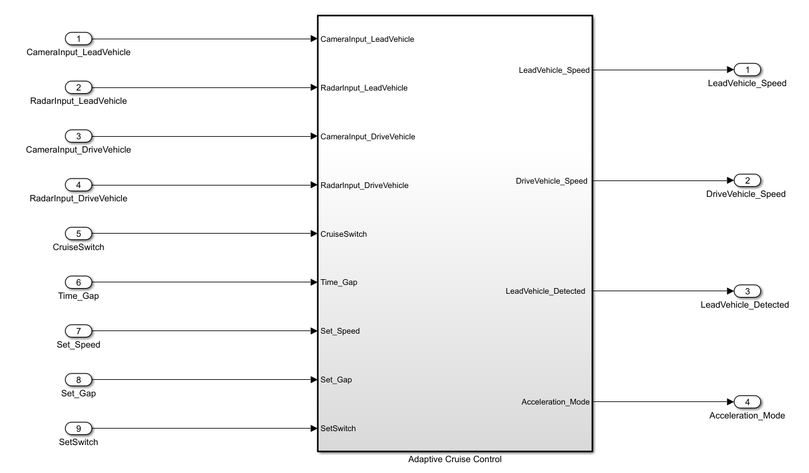
***Drive Vehicle***



***Adaptive Cruise Control Algorithm***

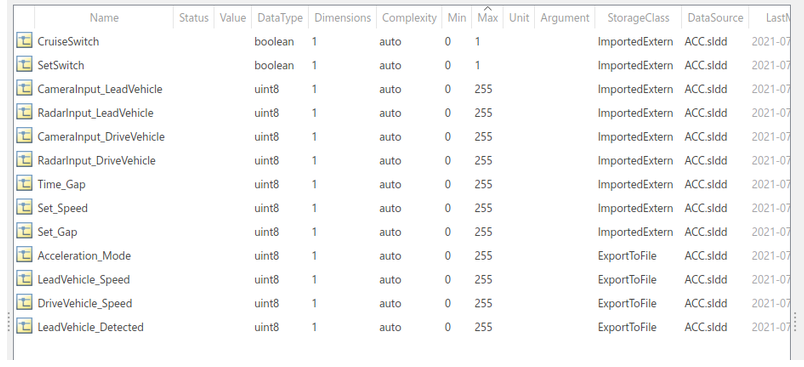
 

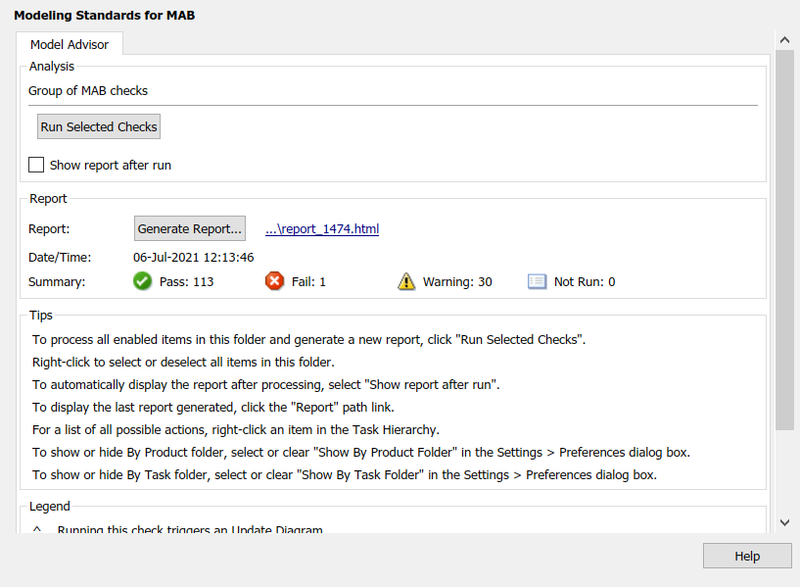




**SLDD File**

The SLDD file is created to store the calibration values, input, and output signals.



**Model Advisor Report**

**Code generation**

/\*

\* File: ACC\_model.c

\*

\* Code generated for Simulink model 'ACC\_model'.

\*

\* Model version : 1.1

\* Simulink Coder version : 9.5 (R2021a) 14-Nov-2020

\* C/C++ source code generated on : Fri Jul 2 13:47:36 2021

\*

\* Target selection: ert.tlc

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

\* Code generation objectives: Unspecified

\* Validation result: Not run

\*/

#include "ACC\_model.h"

#include "ACC\_model\_private.h"

/\* Named constants for Chart: '<S1>/Chart' \*/

#define ACC\_IN\_LeadVehicle\_Not\_Detected ((uint8\_T)3U)

#define ACC\_model\_IN\_NO\_ACTIVE\_CHILD ((uint8\_T)0U)

#define ACC\_model\_IN\_OFF\_mode ((uint8\_T)1U)

#define ACC\_model\_IN\_ON\_mode ((uint8\_T)2U)

#define ACC\_model\_IN\_Standby\_mode ((uint8\_T)3U)

#define IN\_LeadVehicle\_Detected\_Follow ((uint8\_T)1U)

#define IN\_LeadVehicle\_Detected\_Resume ((uint8\_T)2U)

#define IN\_LeadVehicle\_Not\_Detected\_Res ((uint8\_T)4U)

#define IN\_LeadVehicle\_Speed\_equal\_Set\_ ((uint8\_T)5U)

#define IN\_LeadVehicle\_Speed\_lessthan\_S ((uint8\_T)6U)

/\* Block states (default storage) \*/

DW\_ACC\_model\_T ACC\_model\_DW;

/\* External inputs (root inport signals with default storage) \*/

ExtU\_ACC\_model\_T ACC\_model\_U;

/\* External outputs (root outports fed by signals with default storage) \*/

ExtY\_ACC\_model\_T ACC\_model\_Y;

/\* Real-time model \*/

static RT\_MODEL\_ACC\_model\_T ACC\_model\_M\_;

RT\_MODEL\_ACC\_model\_T \*const ACC\_model\_M = &ACC\_model\_M\_;

/\* Model step function \*/

void ACC\_model\_step(void)

{

/\* Outputs for Atomic SubSystem: '<Root>/Subsystem2' \*/

/\* Sum: '<S4>/Add' incorporates:

\* Inport: '<Root>/CameraInput\_DriveVehicle'

\* Inport: '<Root>/RadarInput\_DriveVehicle'

\* UnitDelay: '<S1>/Unit Delay'

\*/

ACC\_model\_Y.DriveVehicle\_Speed = (ACC\_model\_U.CameraInput\_DriveVehicle +

ACC\_model\_Y.Acceleration\_Mode) + ACC\_model\_U.RadarInput\_DriveVehicle;

/\* Sum: '<S3>/Add' incorporates:

\* Inport: '<Root>/CameraInput\_LeadVehicle'

\* Inport: '<Root>/RadarInput\_LeadVehicle'

\*/

ACC\_model\_Y.LeadVehicle\_Speed = ACC\_model\_U.CameraInput\_LeadVehicle +

ACC\_model\_U.RadarInput\_LeadVehicle;

/\* Chart: '<S1>/Chart' incorporates:

\* Inport: '<Root>/CruiseSwitch'

\* Inport: '<Root>/RadarInput\_DriveVehicle'

\* Inport: '<Root>/SetSwitch'

\* Inport: '<Root>/Set\_Gap'

\* Inport: '<Root>/Set\_Speed'

\* Inport: '<Root>/Time\_Gap'

\* UnitDelay: '<S1>/Unit Delay'

\*/

if (ACC\_model\_DW.is\_active\_c3\_ACC\_model == 0U) {

ACC\_model\_DW.is\_active\_c3\_ACC\_model = 1U;

ACC\_model\_DW.is\_c3\_ACC\_model = ACC\_model\_IN\_OFF\_mode;

ACC\_model\_Y.Acceleration\_Mode = 0.0;

} else {

switch (ACC\_model\_DW.is\_c3\_ACC\_model) {

case ACC\_model\_IN\_OFF\_mode:

ACC\_model\_Y.Acceleration\_Mode = 0.0;

if (ACC\_model\_U.CruiseSwitch == 1.0) {

ACC\_model\_DW.is\_c3\_ACC\_model = ACC\_model\_IN\_Standby\_mode;

ACC\_model\_Y.Acceleration\_Mode = 1.0;

}

break;

case ACC\_model\_IN\_ON\_mode:

if (ACC\_model\_U.CruiseSwitch == 0.0) {

ACC\_model\_DW.is\_ON\_mode = ACC\_model\_IN\_NO\_ACTIVE\_CHILD;

ACC\_model\_DW.is\_c3\_ACC\_model = ACC\_model\_IN\_OFF\_mode;

ACC\_model\_Y.Acceleration\_Mode = 0.0;

} else if (ACC\_model\_U.SetSwitch == 0.0) {

ACC\_model\_DW.is\_ON\_mode = ACC\_model\_IN\_NO\_ACTIVE\_CHILD;

ACC\_model\_DW.is\_c3\_ACC\_model = ACC\_model\_IN\_Standby\_mode;

ACC\_model\_Y.Acceleration\_Mode = 1.0;

} else {

switch (ACC\_model\_DW.is\_ON\_mode) {

case IN\_LeadVehicle\_Detected\_Follow:

ACC\_model\_Y.Acceleration\_Mode = 2.0;

if (ACC\_model\_U.RadarInput\_DriveVehicle == 0.0) {

ACC\_model\_DW.is\_ON\_mode = ACC\_IN\_LeadVehicle\_Not\_Detected;

ACC\_model\_Y.Acceleration\_Mode = 1.0;

} else if (((ACC\_model\_U.RadarInput\_DriveVehicle == 1.0) &&

(ACC\_model\_Y.LeadVehicle\_Speed < ACC\_model\_U.Set\_Speed)) ||

(ACC\_model\_U.Time\_Gap < ACC\_model\_U.Set\_Gap)) {

ACC\_model\_DW.is\_ON\_mode = IN\_LeadVehicle\_Speed\_lessthan\_S;

ACC\_model\_Y.Acceleration\_Mode = 4.0;

}

break;

case IN\_LeadVehicle\_Detected\_Resume:

ACC\_model\_Y.Acceleration\_Mode = 3.0;

if ((ACC\_model\_Y.DriveVehicle\_Speed < ACC\_model\_U.Set\_Speed) &&

(ACC\_model\_Y.LeadVehicle\_Speed > ACC\_model\_Y.DriveVehicle\_Speed) &&

(ACC\_model\_U.Time\_Gap >= ACC\_model\_U.Set\_Gap)) {

ACC\_model\_DW.is\_ON\_mode = IN\_LeadVehicle\_Speed\_equal\_Set\_;

ACC\_model\_Y.Acceleration\_Mode = 5.0;

} else if ((ACC\_model\_Y.DriveVehicle\_Speed == ACC\_model\_U.Set\_Speed) &&

(ACC\_model\_Y.LeadVehicle\_Speed >= ACC\_model\_U.Set\_Speed) &&

(ACC\_model\_U.Time\_Gap >= ACC\_model\_U.Set\_Gap)) {

ACC\_model\_DW.is\_ON\_mode = IN\_LeadVehicle\_Detected\_Follow;

ACC\_model\_Y.Acceleration\_Mode = 2.0;

} else if (ACC\_model\_U.RadarInput\_DriveVehicle == 0.0) {

ACC\_model\_DW.is\_ON\_mode = IN\_LeadVehicle\_Not\_Detected\_Res;

ACC\_model\_Y.Acceleration\_Mode = 1.0;

}

break;

case ACC\_IN\_LeadVehicle\_Not\_Detected:

ACC\_model\_Y.Acceleration\_Mode = 1.0;

if ((ACC\_model\_U.RadarInput\_DriveVehicle == 1.0) &&

(ACC\_model\_Y.DriveVehicle\_Speed == ACC\_model\_U.Set\_Speed) &&

(ACC\_model\_Y.LeadVehicle\_Speed >= ACC\_model\_U.Set\_Speed) &&

(ACC\_model\_U.Time\_Gap >= ACC\_model\_U.Set\_Gap)) {

ACC\_model\_DW.is\_ON\_mode = IN\_LeadVehicle\_Detected\_Follow;

ACC\_model\_Y.Acceleration\_Mode = 2.0;

} else if (((ACC\_model\_U.RadarInput\_DriveVehicle == 1.0) &&

(ACC\_model\_Y.LeadVehicle\_Speed < ACC\_model\_U.Set\_Speed)) ||

(ACC\_model\_U.Time\_Gap < ACC\_model\_U.Set\_Gap)) {

ACC\_model\_DW.is\_ON\_mode = IN\_LeadVehicle\_Speed\_lessthan\_S;

ACC\_model\_Y.Acceleration\_Mode = 4.0;

}

break;

case IN\_LeadVehicle\_Not\_Detected\_Res:

ACC\_model\_Y.Acceleration\_Mode = 1.0;

break;

case IN\_LeadVehicle\_Speed\_equal\_Set\_:

ACC\_model\_Y.Acceleration\_Mode = 5.0;

if ((ACC\_model\_U.RadarInput\_DriveVehicle == 0.0) ||

(ACC\_model\_Y.DriveVehicle\_Speed <= ACC\_model\_U.Set\_Speed)) {

ACC\_model\_DW.is\_ON\_mode = IN\_LeadVehicle\_Not\_Detected\_Res;

ACC\_model\_Y.Acceleration\_Mode = 1.0;

} else if (((ACC\_model\_Y.DriveVehicle\_Speed < ACC\_model\_U.Set\_Speed) &&

(ACC\_model\_Y.LeadVehicle\_Speed >

ACC\_model\_Y.DriveVehicle\_Speed)) || (ACC\_model\_U.Time\_Gap

>= ACC\_model\_U.Set\_Gap)) {

ACC\_model\_DW.is\_ON\_mode = IN\_LeadVehicle\_Detected\_Resume;

ACC\_model\_Y.Acceleration\_Mode = 3.0;

} else if (((ACC\_model\_Y.LeadVehicle\_Speed < ACC\_model\_U.Set\_Speed) &&

(ACC\_model\_Y.LeadVehicle\_Speed <

ACC\_model\_Y.DriveVehicle\_Speed)) || (0.75 \*

ACC\_model\_U.Set\_Gap == ACC\_model\_U.Time\_Gap)) {

ACC\_model\_DW.is\_ON\_mode = IN\_LeadVehicle\_Speed\_lessthan\_S;

ACC\_model\_Y.Acceleration\_Mode = 4.0;

}

break;

default:

/\* case IN\_LeadVehicle\_Speed\_lessthan\_Set\_Speed: \*/

ACC\_model\_Y.Acceleration\_Mode = 4.0;

if ((ACC\_model\_U.RadarInput\_DriveVehicle == 0.0) &&

(ACC\_model\_Y.DriveVehicle\_Speed == ACC\_model\_U.Set\_Speed)) {

ACC\_model\_DW.is\_ON\_mode = ACC\_IN\_LeadVehicle\_Not\_Detected;

ACC\_model\_Y.Acceleration\_Mode = 1.0;

} else if ((ACC\_model\_Y.LeadVehicle\_Speed \* 1.25 >=

ACC\_model\_Y.DriveVehicle\_Speed) &&

(ACC\_model\_Y.LeadVehicle\_Speed \* 0.75 <=

ACC\_model\_Y.DriveVehicle\_Speed) &&

(ACC\_model\_Y.DriveVehicle\_Speed < ACC\_model\_U.Set\_Speed) &&

(ACC\_model\_U.Time\_Gap <= 1.25 \* ACC\_model\_U.Set\_Gap) &&

(ACC\_model\_U.Time\_Gap >= 0.75 \* ACC\_model\_U.Set\_Gap)) {

ACC\_model\_DW.is\_ON\_mode = IN\_LeadVehicle\_Speed\_equal\_Set\_;

ACC\_model\_Y.Acceleration\_Mode = 5.0;

}

break;

}

}

break;

default:

/\* case IN\_Standby\_mode: \*/

ACC\_model\_Y.Acceleration\_Mode = 1.0;

if (ACC\_model\_U.CruiseSwitch == 0.0) {

ACC\_model\_DW.is\_c3\_ACC\_model = ACC\_model\_IN\_OFF\_mode;

ACC\_model\_Y.Acceleration\_Mode = 0.0;

} else if (ACC\_model\_U.SetSwitch == 1.0) {

ACC\_model\_DW.is\_c3\_ACC\_model = ACC\_model\_IN\_ON\_mode;

ACC\_model\_DW.is\_ON\_mode = IN\_LeadVehicle\_Detected\_Follow;

ACC\_model\_Y.Acceleration\_Mode = 2.0;

}

break;

}

}

/\* End of Chart: '<S1>/Chart' \*/

/\* End of Outputs for SubSystem: '<Root>/Subsystem2' \*/

/\* Outport: '<Root>/LeadVehicle\_Detected ' incorporates:

\* Inport: '<Root>/RadarInput\_DriveVehicle'

\*/

ACC\_model\_Y.LeadVehicle\_Detected = ACC\_model\_U.RadarInput\_DriveVehicle;

}

/\* Model initialize function \*/

void ACC\_model\_initialize(void)

{

/\* (no initialization code required) \*/

}

/\* Model terminate function \*/

void ACC\_model\_terminate(void)

{

/\* (no terminate code required) \*/

}

/\*

\* File trailer for generated code.

\*

\* [EOF]

\*/

**Steps followed for code generation**

* Right click on the sub system
* Select block parameters (subsystem)
* Checkmark on ‘treat as atomic size’.
* Go to Model explorer settings
* Configure the solver to fixed and change the code generation to ert.tlc
* Then go to embedded coder app.
* Select ‘Generate code’ in the build option.

**Conclusion**

* Successfully created a model of Adaptive cruise control system using the requirement data.
* Created a Simulink data dictionary for the input, output, and calibration signals.
* Did the model advisor check and generated the report.
* Generated the code using the embedded coder app.