

POLITECNICO DI TORINO

Master's Degree in Electronics Engineering /
Embedded Systems



Department of Electronics and Telecommunications

Development of Radiations tolerant Attitude Control Algorithm for small Satellites

Supervisor

Prof. Reyneri LEONARDO

Candidate

Zubair Ahmed JUNAID

December 2020

Abstract

Small satellites provide flexible and economical cost-effective solutions for research and development in the field of the space industry. It is possible to implement and test various mission algorithms with very few resources available. The stringent requirement of a smaller mass of spacecraft has almost eliminated the use of propulsion systems in small satellites. The smaller mass and size of the spacecraft, on the other hand, make it easier to control and adjust the attitude and orbit of the spacecraft with only magnetic control or inertial control. In this field, there are many attitude and orbit control algorithms and techniques that have already been explored.

As compared to the heavier satellites, the smaller satellites are equally exposed to the radiations and experience the same harsh environment in space but due to the low mass budgeting, it is not preferred to use redundant hardware or extra shielding for them. Nowadays there are many circuits available that are made hardened against radiations during manufacturing. Integrated circuits, processors, and micro-controllers during the design phase can also implement hardware-level radiation hardening techniques to tolerate radiation exposure but this solution leads to very expensive and relatively slow hardware, which does not align with the main motivation of small satellites, i.e. small, inexpensive, and efficient.

In order to make the satellite to work while exposed to a radiation environment, software methods can be used. Once it is studied that how often and in which manner the radiations can affect the normal operations of spacecraft, a flexible and robust software algorithm can be developed which can tolerate not all, but many radiation events. This can lead to using cheap commercial off-the-shelf devices without using redundant hardware and heavy shielding. This technique may cause the system to use more memory and perform computation slowly, but it will be much faster and effective as compared to the hardware hardening techniques.

The work presented in this report comprised of developing a closed-loop Attitude control flight software for small satellites, such as CubeSat, along with spacecraft dynamics, and environmental effects. The flight software that interacts with the sensors and actuators mathematical models, can make the satellite work into any of the predefined mission modes. The flight software is made robust and hardened against space radiations which can alter the

normal software operations and can result in malfunctioning. The radiation effects are modeled in the software and are simulated. A comparative analysis is also performed to express how a radiation-tolerant flight software performs in presence of radiation effects as compared to non-hardened flight software.

Table of Contents

List of Tables	IV
List of Figures	V
Acronyms	VII
1 Introduction	1
1.1 Aim of Study	1
1.2 Proposed Solutions	2
1.3 Previous Work	2
1.4 Thesis Organization	3
2 Definitions and Notations	4
2.1 Reference Systems	4
2.1.1 Earth Centered Inertial (ECI) Frame	4
2.1.2 Earth Centered Earth Fixed (ECEF) Frame	4
2.1.3 North East Down (NED) Frame	5
2.1.4 Orbit Reference Frame	6
2.1.5 Body Frame	6
2.2 Rotations	6
2.2.1 Rotation Matrix	7
2.2.2 Euler Angles	7
2.2.3 Quaternions	8
2.3 Space Radiations	9
2.3.1 Types of Space Radiations	10
2.3.2 Effects of Space Radiations	10
3 Attitude Control System	12
3.1 Mathematical Modeling	12

3.1.1	Mission Description	13
3.1.2	Orbital Parameters	14
3.2	Environment	15
3.2.1	Gravitational Model	15
3.2.2	Sun Position Model	15
3.2.3	Magnetic Field Model	16
3.2.4	Disturbances	16
3.3	Satellite Dynamics	18
3.4	Sensors	22
3.4.1	Magnetometer	22
3.4.2	Gyroscope	23
3.4.3	Sun Sensor	23
3.4.4	GPS	25
3.5	Actuators	26
3.5.1	Magnetorquer Rods	27
3.5.2	Reaction Wheels	28
3.6	Flight Software	30
3.6.1	Detumbling Mode	31
3.6.2	Sun Pointing Mode	32
3.6.3	Nadir Pointing Mode	35
3.6.4	Visualization	35
4	Radiations Hardened Software	38
4.1	Development of Flight Software	38
4.1.1	Data Structures	39
4.1.2	Software Hierarchy	39
4.2	Development of Radiation Hardened Flight Software	40
4.2.1	Radiation Effects on Software	40
4.2.2	Radiation Hardening Library	41
4.2.3	Data Structures	42
4.2.4	Software Hierarchy	42
4.2.5	Radiation Effects Sub-Routine	43
4.2.6	Radiation Tolerance	43
4.2.7	S-functions Implementation	43
5	Simulations and Results	45
5.1	Matlab & Simulink Results	45
5.1.1	Development Platforms & Tools	45

5.1.2	Mission Modes	45
5.2	Radiation Effects	48
5.2.1	Mission Modes Simulation Results	50
5.3	Radiation Hardened Software	52
5.3.1	Mission Modes Simulation Results	52
5.4	Comparative Analysis	55
6	Conclusion	59
6.1	Future Work	60
A	Flight Software Source Code	61
	Bibliography	111

List of Tables

3.1	Orbital Parameters Information	14
3.2	Gravitational Model Simulink Block Inputs & Outputs . . .	16
3.3	Sun Position Model Simulink Block Inputs & Outputs	17
3.4	Magnetic Field Model Simulink Block Inputs & Outputs . .	18
3.5	Vehicle Dynamics Simulink Block Inputs and Outputs	21
3.6	Magnetometer Simulink Block Inputs and Outputs	22
3.7	Gyroscope Simulink Block Inputs & Outputs	23
3.8	Sun Sensor Simulink Block Inputs & Outputs	25
3.9	GPS Simulink Block Inputs & Outputs	25
3.10	Magnetorquer Rods Simulink Block Inputs & Outputs . . .	27
3.11	Reaction wheels Simulink Block Inputs & Outputs	30
3.12	Detumbling Mode Simulink Block Inputs & Outputs	33
3.13	Sun Pointing Mode Simulink Block Inputs & Outputs	34
3.14	Nadir Pointing Mode Simulink Block Inputs & Outputs . . .	36
4.1	List Of Data Structures in C++ Flight Software	39
5.1	Detumbling mode performance comparison	56
5.2	Nadir Pointing mode performance comparison	57
5.3	Sun Pointing mode performance comparison	58

List of Figures

2.1	ECI Coordinate System[3]	5
2.2	ECEF Coordinate System[3]	5
2.3	NED Coordinate System[4]	6
2.4	Orbit Reference frame and Body Coordinate System[2] . . .	7
2.5	Quaternion Representation of rotation [6]	9
3.1	Attitude Control System Block Diagram	13
3.2	Gravitational Forces acting on the Satellite	16
3.3	Sun Vector Model	17
3.4	Magnetic Field Model	18
3.5	Satellite Dynamics Simulink Library Block	19
3.6	Magnetometer Simulink Block	23
3.7	Gyroscope representation of Simulink Block	24
3.8	Sun Sensor representation of Simulink Block	24
3.9	GPS representation of Simulink Block	26
3.10	Magnetorquer Rods model Simulink representation	28
3.11	Tetrahedron configuration of Reaction Wheels[2]	29
3.12	Reaction wheels representation of Simulink Model	29
3.13	Flight Software Top Level hierarchy in Simulink	31
3.14	Detumbling Mode Matlab/Simulink Block	33
3.15	Sun Pointing Mode Simulink Model	34
3.16	Nadir Pointing Mode Simulink Model	35
3.17	Three-Dimensional Visualization of satellite in Orbit	37
4.1	Radiation Hardened Software Hierarchy	42
4.2	S-function Builder Representation of Flight Software	44
5.1	Detumbling Mode Simulation Results: Angular rates along xyz body axes	46

5.2	Nadir Pointing Mode Attitude Error: Euler angles	47
5.3	Nadir Pointing Mode Attitude Error steady state: Euler angles	48
5.4	Sun Pointing Mode Attitude Error: Euler angles	49
5.5	Sun Pointing Mode Attitude Error steady state: Euler angles	49
5.6	Radiations affected Detumbling Mode Simulation Results: Angular rates along xyz body axes	50
5.7	Radiations affected Nadir Pointing Mode Simulation Results: Euler angles	51
5.8	Radiations affected Sun Pointing Mode Simulation Results: Euler angles	52
5.9	Radiation Hardened Detumbling Mode Simulation Results: Angular rates along xyz body axes	53
5.10	Radiation Hardened Nadir Pointing Mode Simulation Results: Euler angles	54
5.11	Radiation Hardened Nadir Pointing Mode steady state Simu- lation Results: Euler angles	55
5.12	Radiation Hardened Sun Pointing Mode Simulation Results: Euler angles	56
5.13	Radiation Hardened Sun Pointing Mode steady state Simula- tion Results: Euler angles	57

Acronyms

ECI

Earth Centered Inertial co-ordinate Frame

ECEF

Earth Centered Earth fixed co-ordinate Frame

NED

North East Down co-ordinate Frame

ORF

Orbit Reference Frame

PID

Proportional-Integral-Derivative

LEO

Low Earth Orbit

SEU

Single Event Upset

Chapter 1

Introduction

For more than the past decade, small satellites known as CubeSats and nanosats have become an area of research and development. The enhancements in the scientific equipment of satellite payload, as well as the platform, have made the possibility of size and mass reduction of the satellite. Many academic institutes and commercial companies are doing research and experimenting with new innovations in terms of software algorithms and also in terms of hardware. In terms of orbit and attitude control of the satellite, due to the reduced mass and volume, the forces and torques needed to control the position and three-dimensional orientation of the satellite have also reduced. This has led to the almost elimination of propulsion systems from small satellites, leaving the control only to the magnetic or inertial control. Many techniques have been developed and explored in this regard. Unfortunately in space, despite having different sizes and masses, the space radiations act on all satellites the same way. It can cause transient bit flipping to permanent fault induction in hardware and memory element or can cause short-circuit or open-circuit resulting in malfunctioning of a device or set of devices or in the worst case the mission failure. Also, the longer exposure to radiation can tear-off the parts of satellites causing physical damage to the spacecraft.

1.1 Aim of Study

The work presented in this report is focused on the development of a basic attitude and orbit control algorithm which can withstand radiation effects. The study aimed at the development of a closed-loop mathematical model in order to simulate a set of control algorithms for a complete mission of a

small satellite. The other part of the study focused on making these control algorithms radiations tolerant. Then the comparative analysis is performed between hardened flight software and un-hardened flight software against radiation effects.

1.2 Proposed Solutions

With the help of the MathWorks modeling tool, Matlab/Simulink, it is possible to model and simulate the attitude and orbit subsystem of a satellite. The attitude control subsystem with different environmental effects can be simulated along with sensors and actuators' mathematical models. The satellite can be made to point to different targets or to damp its angular rotations based on the mission control modes. The whole system can be visualized in three-dimensional space with the help of a virtual reality add-on available in Matlab/Simulink. The flight software model can be converted into C or C++ language in order to model and analyze the effects of radiations. Some hardening techniques can be used to make the flight software hardened and more robust against radiation effects. The hardened software can be plugged back into the Simulink model by using the s-function capability of Matlab which allows a low-level language developed function or set of functions to run as a part of the Simulink model. The comparative analysis can be performed in the end to evaluate the study and simulation results.

1.3 Previous Work

There is an already developed C++ library that implements the triple redundancy on the level of the variables. It stores three private copies of each defined data type variable and over-rides the basic operators, providing a seamless view to the user. The ideology is such that during the radiation event, it is common that Single Event Upsets happen. This phenomenon not only cause the bit-flipping in memories but also can affect the execution of program by corrupting the control variables or any vital data. Defining three copies of same variable will most probably cause the corruption of only one version of stored copy, leaving the other two versions unchanged. Upon reading the variable, a voting is performed and the correct version of data is retrieved [1].

Defining the triple data can be seen as:

$$\textit{HardenedData} :: \textit{TripleData} < \textit{DataType} > \textit{Variable}; \quad (1.1)$$

Accessing any of the private copy is not the goal of normal software operations, but can be done for the sake of radiation noise injection in the following way:

```
Variable.a();  
Variable.b();  
Variable.c();
```

There is also another Library provided, written in C++ language, with the name of *SingleData.cpp*, that overrides the basic operations just like triple data, but keeps only one copy of the data. The purpose of this file is to provide a flexibility at the development level that the user can choose to use radiation hardened software or un-hardened software in order to analyze the effects of radiations.

1.4 Thesis Organization

Chapter 1 discusses the brief introduction to the topic with the aim defined to conduct this study. A short summary of the proposed solution is also mentioned which is discussed in further detail in the following chapters. The previous work done which is used as a starting point of this thesis is also described.

Chapter 2 discusses the basic notations and definitions and concepts used in the whole study. A basic definition of various coordinate transformations and rotation matrices is discussed. The concept of space radiations is also discussed.

Chapter 3 discusses the Attitude Control System in detail with the description of Matlab/Simulink various blocks. The flight software is also presented in detail.

Chapter 4 discusses the methodology for the radiation hardening of control algorithms. Development platform and building blocks of Radiations hardening software are also presented.

Chapter 5 discusses the simulation results and concludes the comparative analysis of radiation-tolerant and intolerant software. This chapter concludes the study and highlights all the results achieved and summarizes them.

Chapter 6 discusses the conclusion of the report and the presented work.

Chapter 2

Definitions and Notations

2.1 Reference Systems

Reference systems are extremely important when defining and studying the space craft motion and orientation in space. There are numerous co-ordinate frames used in the report which are summarized below.

2.1.1 Earth Centered Inertial (ECI) Frame

This is an inertial reference frame with the origin lies at the center of earth. The x-y plane is the equatorial plane with the z-axis points towards the north pole of earth.

The x-axis points towards the vernal equinox and y-axis completes the right-hand Cartesian coordinate system[2]. The nutation and precession motions of earth are not considered in this report for the sake of simplicity.

2.1.2 Earth Centered Earth Fixed (ECEF) Frame

Earth centered Earth fixed frame,as the name suggests, rotates with the rotation of earth. The origin is again at the center of the earth, the z-axis points towards the North pole and is aligned with the rotational axis of earth. The x-axis points toward the intersection of Greenwich meridian with the equator. The y-axis completes the right handed system.[3]

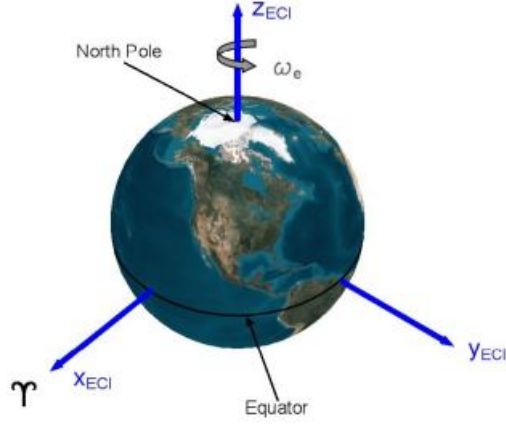


Figure 2.1: ECI Coordinate System[3]

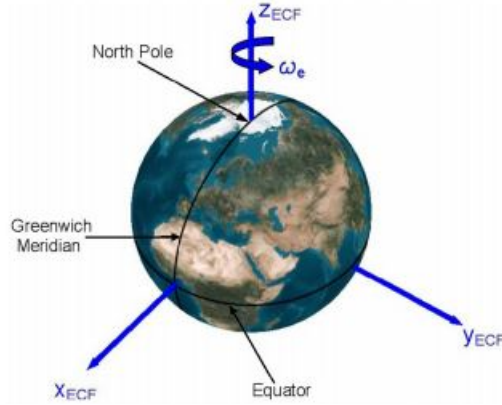


Figure 2.2: ECEF Coordinate System[3]

2.1.3 North East Down (NED) Frame

The North East Down coordinate system defines a reference frame on the body of spacecraft. The x-axis points parallel towards the north pole of earth, the y-axis points parallel towards the latitude curve of earth and z-axis points towards the center of earth, completing the right hand rule.

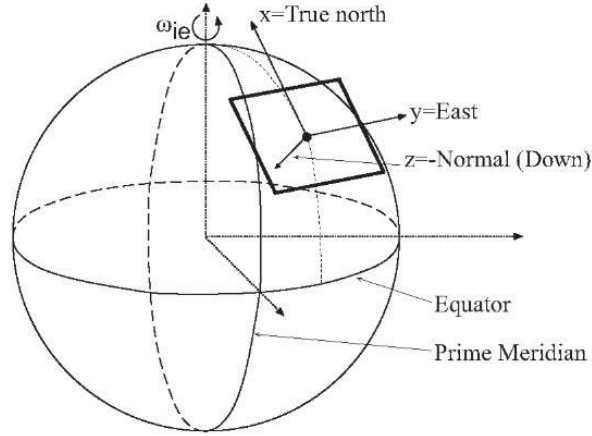


Figure 2.3: NED Coordinate System[4]

2.1.4 Orbit Reference Frame

The origin of Orbit reference frame lies on the center of mass of space craft, while z-axis points towards the center of the Earth. X-axis points in the direction of motion tangentially to the orbit [2].

2.1.5 Body Frame

The Body coordinate frame moves and rotates with the body movement. The origin lies on the center of mass of satellite. The z-axis points towards the face which is usually required to face towards earth during earth-pointing mode, where the scientific equipment of a LEO satellite is placed. The x-axis points along the solar panels, considering the deploy-able solar panels. The y-axis completes the right-hand rule.

2.2 Rotations

In order to represent the orientation of space craft in different frame of references and to perform different operations on vectors that are described in different frames of reference, the rotations of vectors is necessary.

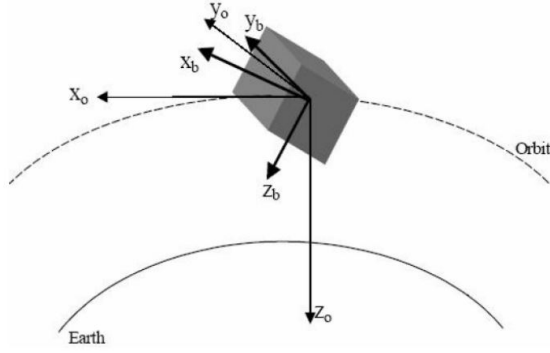


Figure 2.4: Orbit Reference frame and Body Coordinate System[2]

2.2.1 Rotation Matrix

A rotation matrix is a matrix that represents the rotations of two frame of references in Euclidean Space. The rotation of a vector from one frame to another frame can be performed as follows:

$$v^y = R_x^y \times v^x \quad (2.1)$$

2.2.2 Euler Angles

The successive angular rotations about the three orthogonal body axes are defined as the Euler angle rotations[2]. Roll(ϕ), pitch(θ) and yaw(ψ) are respectively the rotations around x, y and z axis.

The rotation around x-axis can be seen as the x-axis component of source and target vector does not change,

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \quad (2.2)$$

The rotation around y-axis, which is called pitch, can be seen as the y-axis component of source and transformed vector does not get affected.

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \quad (2.3)$$

When the rotation around z-axis is performed, which is called yaw, can be seen in the following equation as the z-axis component of source and target vector remains unchanged.

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.4)$$

2.2.3 Quaternions

Quaternions are an alternate way to describe orientation and rotations in three dimensional space using a set of four numbers. They have the ability to uniquely describe any three-dimensional rotation about an arbitrary axis and do not suffer from gimbal lock [5].

quaternions are represented as:

$$\underline{q} = q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} + q_4 \quad (2.5)$$

It is convenient and preferable to define the attitude of the satellite using quaternions. It is a generalized complex number, composed of the sum of a real number q_4 and a 3D vector q_{1-3} [3].

The closest approximation of understanding that can be developed to grasp the concept of quaternions is that it is a rotation between two reference frames on an *arbitrary axis*. The scalar part represents the amount of rotation which is the angle, and the vector part represents the axis of rotation.

An important property of the quaternions is that sum of squares of all four components is always 1 [3].

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1 \quad (2.6)$$

The quaternions can be used to transform one quaternion orientation to another quaternion using multiplying them together. For example, quaternion rotation between reference frame \mathcal{B} and \mathcal{I} can be represented as multiplying the quaternion rotations that represent rotations between \mathcal{B} to \mathcal{A} and \mathcal{A} to \mathcal{I} .

$$\underline{q}_{31} = \underline{q}_{34}\underline{q}_{41} \quad (2.7)$$

During the pointing control modes of space craft, the attitude error between the reference attitude quaternion and the current attitude quaternion is also

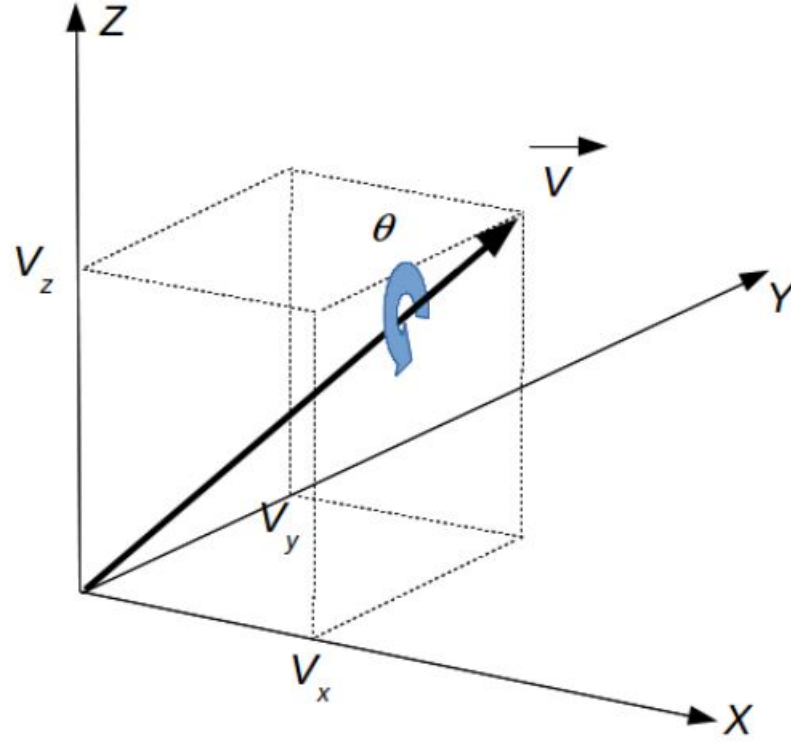


Figure 2.5: Quaternion Representation of rotation [6]

calculated in the similar way.

$$\underline{q}_{error} = \underline{q}_{ref} \underline{q}_{attitude} \quad (2.8)$$

2.3 Space Radiations

Space weather is a phenomenon of effects of solar activity on the near-earth space environment. The energized charged particles can not only bombard the exposed surfaces of satellites but also can penetrate through sometimes hardware devices and circuits and cause hardware and software malfunctioning. The unexplainable errors in functions of sensitive parts and also the degradation of structural materials are known effects of harsh space weather[7].

2.3.1 Types of Space Radiations

There are three types of space radiations widely considered for space craft modeling.

The Trapped Particles

The charged particles from solar wind are trapped in the earth's surrounding Van Allen radiation belts and travel along the lines of magnetic field. The inner belt contains protons with energy exceeding 10 MeV, while the outer belt mostly contains charged electrons. The satellites experience the radiation effects caused by these trapped particles[7].

The Transient Particles

The transient particles are mainly the reason of radiation events such as solar wind, solar flares, coronal mass ejections (CME)[7]. The large scale disturbances caused by transient particles can impact on space crafts normal operations.

Galactic Cosmic Radiation

These radiations originate from outside solar system. They consist of ionized atoms ranging from a single proton up to a uranium nucleus[7]. The flux of this radiation type is low, Earth's magnetic field provides natural shielding against it. Therefore, as the altitude increases, the exposure of GCRs increases.

2.3.2 Effects of Space Radiations

When a satellite is injected into its orbit, the space weather affects its normal mission life in various ways. The most common effects are:

- ***spacecraft charging***: It can be surface charging which can cause structural damage to the space craft or it can be internal dielectric charging which might cause the abnormal operations and damage of sensitive electronics equipment.
- ***Total Ionizing Dose***: It defines the total energy that ionization processes create and deposit in materials when energized particles pass

through it [7]. It can degrade the structure of satellite and can cause mission failure also.

- ***Displacement Damage***: An incoming energetic particle may collides with the lattice atoms and can result in atoms displacement. The properties of bulk semiconductor material may alter which might cause the silicon level defect.
- ***Single Event Effects***: It is the effect when a single incident ionizing particle depositing enough energy to cause an abnormal effect[7]. There are four types of single event effects:
 - *SEU*: Single Event Upset
 - *SEL*: Single Event Latch-up
 - *SEB*: Single Event Burn-out
 - *SEGR*: Single Event Gate Rupture

SEUs result in change of logic state of a part of sensitive micro-electronic device. It can result in bit flipping of some part of data while it is residing in the memory or during execution. This can cause the software to consider the wrong values of data and can result in program corruption. An even more severe case of SEU is Single Event Function Interrupt (SEFI), which can affect the control of software and can put the system in test mode, halt or any undefined state. This will then require a full reset of the system.

In this report, the SEU bit flipping is mainly targeted, modeled and treated.

Chapter 3

Attitude Control System

The three dimensional orientation of spacecraft in space is termed as attitude of the satellite. Having continuous, or in some cases periodic knowledge, and having the ability to control it under desired thresholds is of vital importance to every satellite mission. Few main purpose of attitude controlling of a space craft can be described as:

- To point the scientific equipment of satellite in right direction (under threshold desired accuracy) so that mission objectives can be successfully achieved.
- To keep the sensitive sensors and scientific equipment to face the direct sunlight and in some cases direct earth to avoid the earth albedo effect, so that they might not get damaged.
- To keep pointing the solar panels towards sun as maximum as possible in order not to drain out the batteries power.

3.1 Mathematical Modeling

The Attitude Control System is modeled and simulated through Mathworks Matlab/Simulink mathematical modeling tool. It provides a powerful platform to not only simulate the mathematical equations for space craft but also it can run the C/C++ software codes completely integrated with the Simulink blocks and seamless to the user. The extensive Simulink library provides various blocks with plug-n-place capability to make the modeling easier.

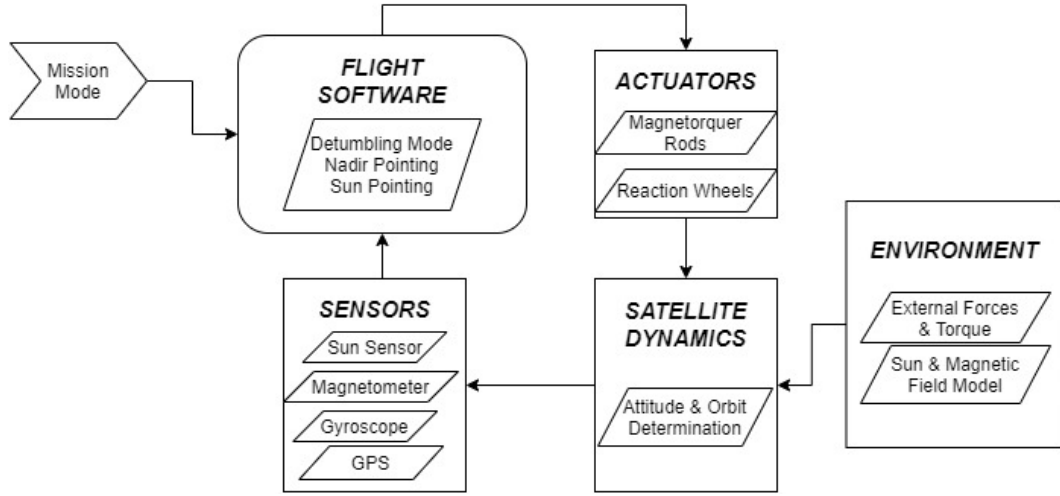


Figure 3.1: Attitude Control System Block Diagram

A closed-loop CubeSat simulation library is provided with the Matlab/Simulink Aerospace Blockset which lets the user to model, simulate, analyze, and visualize the motion and dynamics of CubeSats and nano satellites, which are miniaturized spacecraft designed for space research[8].

A simplified block diagram of the Attitude Control System presented in this report is given. The details of different blocks of Simulink used for closed loop simulation of Attitude Control System are described in this chapter.

3.1.1 Mission Description

In the work presented here, a LEO satellite mission with a general imaging payload is considered. For each of such kind of missions, three basic maneuvers or attitude controls are required to be implemented for the completion of basic mission functionalities. The following working modes are considered in this work.

- **Detumbling Mode:** When the satellite is ejected from the launcher and is injected into its orbit, the initial angular velocities in each of its body axes with respect to any of the inertial frame of references are very large and sometimes in the order of 10's of degrees per second. It is very crucial to implement a control law that can help slow down the body angular velocities under some threshold. In this study, this control mode is implemented with the name of *Detumbling Mode*, that relies on the

magnetometers and magnetorquer rods to damp the satellite body rates. Another purpose of this modes is to *de-saturate* the reaction wheels of satellite because when the reaction wheels are required to slow down their speed, they induce the angular torques in the body which are also damped via this mode.

- **Sun Pointing Mode:** To keep charging the batteries of satellite, it is necessary to keep pointing the solar panels towards the sun. Therefore, a specific pointing mode is designed in which satellite points its face, that contains the solar panels, towards the sun. In the study presented here, the sun pointing mode is considered as a separate mission mode which relies on the sun sensors and attitude information, and gives commands to reaction wheels in order to point towards the sun.
- **Nadir Pointing Mode:** The mission mode in which satellite is required to point its scientific payload towards its target is a basic necessary mission mode. In the study presented here, the satellite is made to point towards the center of earth. This mode relies on the GPS data and attitude information, and controls the attitude using reaction wheels.

3.1.2 Orbital Parameters

The mission is considered as a general LEO satellite in order to simulate and understand the attitude control software and radiation hardening implementation. The summary of mission geometry can be sum up as following:

Keplerian Orbital Elements	Values
Semi-major Axis (a)	6786233.13
Eccentricity (e)	0.0010537
Inclination (i)	51.7519
Right Ascension of Ascending Node (Ω)	95.2562
Argument of periapsis (ω)	93.4872
True Anomaly (v)	302.9234
Epoch Date & Time	4-January-2019-12:00

Table 3.1: Orbital Parameters Information

3.2 Environment

The first component of closed-loop simulator is the environment model which simulates the environment effects that act on the satellite at each time stamp. These blocks are not directly the part of Attitude Control system, but they are necessary for the system to work in closed loop manner. The environmental forces and torques acting on the body along with the magnetic field intensity and sun position model are the main components of environmental effects modeling.

3.2.1 Gravitational Model

The aerospace blockset provides a Spherical Harmonic Gravity Model that implements the mathematical representation of spherical harmonic gravity based on the earth gravitational potential. It provides a convenient way to describe the earth's gravitational field outside of its surface in spherical harmonic expansion[9].

The gravitational force acting on the satellite can be described as:

$$F = G \frac{Mm}{r^2} \quad (3.1)$$

where G is the gravitational constant and its value is $6.67430 \times 10^{-11} \frac{Nm^2}{kg^2}$. This block takes the position of satellite at each instant in ECEF coordinate frame and calculates the gravity values in ECEF coordinate frame. These values are then multiplied with the mass of satellite and transformed into the body frame in order to represent the gravitational force acting on the satellite in Body Coordinate frame, according to the Newton's second law of motion. The precession of the earth is ignored for the sake of simplicity.

$$F = ma \quad (3.2)$$

The Simulink block of *Gravitational Model's* inputs and outputs are given below.

3.2.2 Sun Position Model

The sun model calculates the sun vector at each time stamp provided. The model is part of the aerospace block-set celestial phenomenon. The input is

Input	Unit	Output	Unit
xECEF (ECEF Position)	meters	EnvForceBody (Environmental Force on Body)	Newton

Table 3.2: Gravitational Model Simulink Block Inputs & Outputs

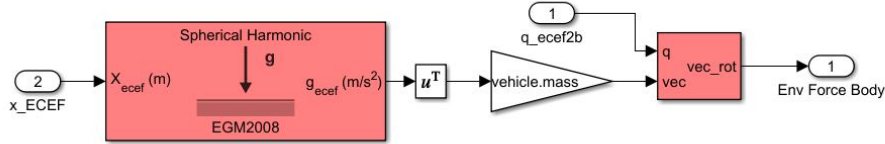


Figure 3.2: Gravitational Forces acting on the Satellite

the current timestamp in Julian Date format. The output is the sun vector calculated in the ECI frame of reference.

The Simulink block of *Sun Position Model*'s inputs and outputs are given below.

3.2.3 Magnetic Field Model

The magnetic field model is also the part of aerospace block-set environment. It implements the 12th degree International Geomagnetic Reference Field model. The inputs are the position of space craft in terms of latitude, longitude and altitude. Also the start date time is also required to be provided. The output, which is of interest, is the Magnetic Field intensity vector in NED coordinate frame. The output unit is nano-tesla.

The Simulink block of *Magnetic Field Model*'s inputs and outputs are given below.

3.2.4 Disturbances

When the satellite is injected into its orbit, it experiences several environmental disturbances. These disturbances in some cases must be considered and modeled properly in order to calculate the proper torque commands for actuators. In this study, for the sake of simplicity of model and unit inertia

Input	Unit	Output	Unit
JD (Julian Date converted from Simulation time)	sec	xSumECI (Sun Vector in ECI Frame)	meters

Table 3.3: Sun Position Model Simulink Block Inputs & Outputs

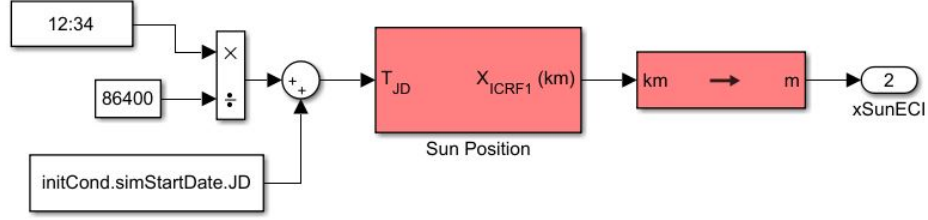


Figure 3.3: Sun Vector Model

matrix consideration, the disturbances are not taken into account. Generally, the environmental disturbances that can act on the body of space craft are following:

Gravitational Torque

Gravitational force acting on the satellite body causes a gravitational torque on the body when its mass is not symmetrically distributed. This non-zero torque is due to the different amount of gravitational force acting on the different parts of satellite. It can be modeled as[10]:

$$T_G = (F_2 - F_1) \frac{l}{2} \sin \alpha \quad (3.3)$$

Aerodynamic torque

The air density at higher altitude start to lower. Some LEO satellites may face some drag but it is in quite lower amount. The drag force acting on the satellite body can be modeled as[10]:

$$F_{AD} = \frac{\rho C_D A V_o^2}{2} \quad (3.4)$$

Input	Unit	Output	Unit
Lat-Lon-Alt (Latitude, Longitude and Altitude of satellite)	deg, deg & meter	Bned (Magnetic Field Intensity Vector in NED Frame)	Tesla

Table 3.4: Magnetic Field Model Simulink Block Inputs & Outputs

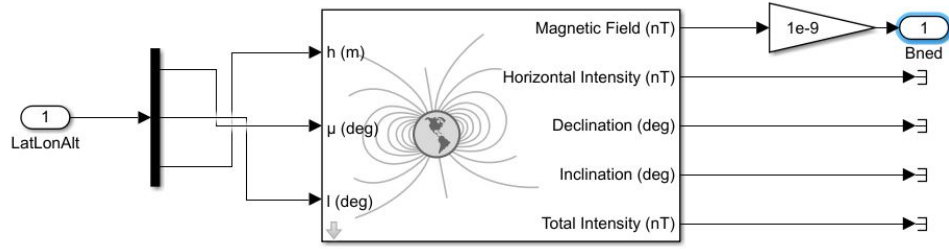


Figure 3.4: Magnetic Field Model

Magnetic Torque

The scenario when the spacecraft has non-zero magnetic moment \mathbf{m} due to hysteresis and other currents flowing, the external magnetic field flux \mathbf{B} causes a torque to be produced in the body.

$$\vec{T}_m = \vec{m} \vec{B} \quad (3.5)$$

Solar Radiation Pressure

The solar radiation pressure, as the name suggests, is the force of solar radiations acting on the body. This force, when acting on a moment arm of satellite, creates a torque on the body.

$$\vec{T}_s = \vec{r}_{sp} \times \vec{F}_s \vec{F}_s = (1 + K) \rho_s A_{\perp} \quad (3.6)$$

3.3 Satellite Dynamics

Simulink provides a powerful library block for the analysis of satellite dynamics equation and calculations of various transformation matrices. The aerospace blockset equations of motions' 6DOF (6 Degrees of Freedom) represents the orientation of satellite in space using quaternions representation

defined in ECEF frame. The block takes the initial position, velocity, orientation, and angular rates along with the space craft mass and inertia as the initial conditions. Then it propagates the output vectors at each time stamp based on the previous states. The block also takes the forces and torques acting on the satellite from the environment, and also the torques acting on the body from the actuators. This block provides the inputs for

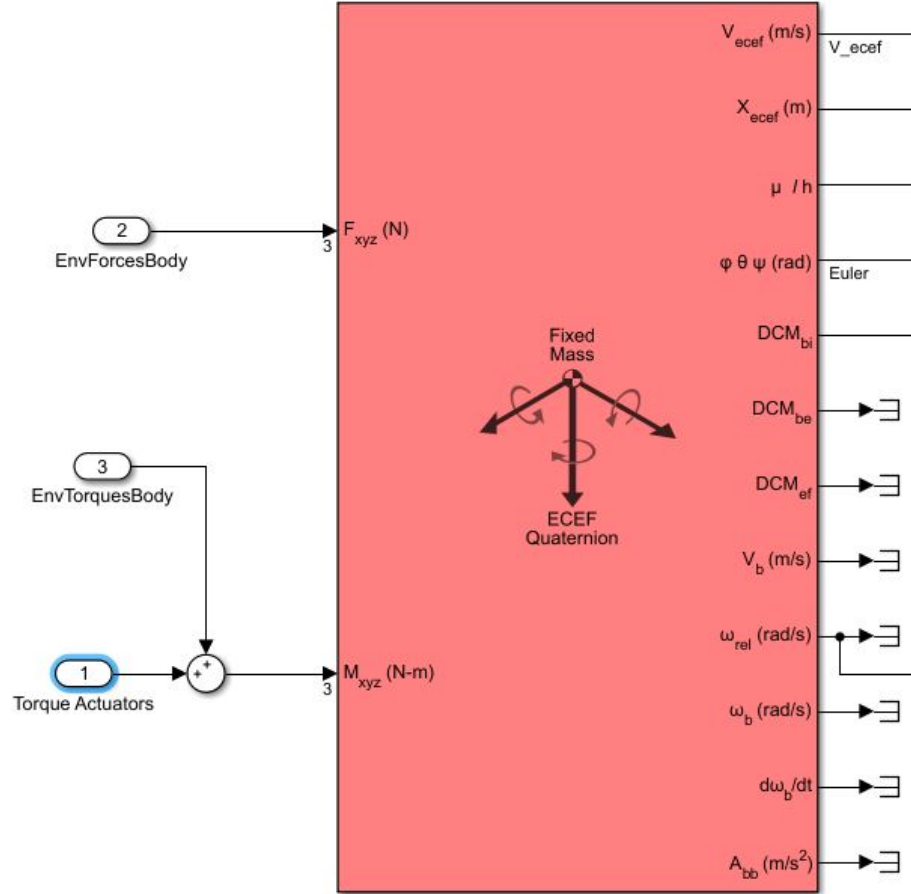


Figure 3.5: Satellite Dynamics Simulink Library Block

the sensors and after few transformations, the quaternions representing the attitude of space craft can also be computed. The quaternion angles from body to ECEF frame and from body to ECI are also computed. Moreover, the rotation matrix from body to NED frame is also possible to calculate. The rates in the principal axes of body are also calculated.

The inputs and the outputs generated by this block which are used by further blocks are summarized below:

Input	Unit	Output	Unit
Forces (Environmental Forces acting on the body)	N (Newton)	Vecef (3×1 Vector. Velocity of Satellite in ECEF Frame)	m/s (meters/second)
Torques (Environmental Torques and Actuators Torques acting on the body)	N-m (Newton meter)	Xecef (3×1 Vector. Position of satellite in ECEF Frame)	m (meters)
		LatLonAlt (3×1 Vector. Latitude, Longitude & Altitude of satellite)	Deg,Deg & meters
		DCM_{bi} (3×3 Vector. Direction Cosine Matrix between ECI to Body Frame)	-
		DCM_{be} (3×3 Vector. Direction Cosine Matrix between ECI to NED Frame)	-
		DCM_{ef} (3×3 Vector. Direction Cosine Matrix between ECEF to NED Frame)	-
		ω_{rel} (3×1 Vector. Angular rates of the body with respect to NED Frame)	rad/sec (radians per second)

Table 3.5: Vehicle Dynamics Simulink Block Inputs and Outputs

3.4 Sensors

Sensors are the eyes and ears of a space craft for controlling and estimating the attitude of satellite. There are few sensors that are modeled in this work that support the attitude control system. The purpose of sensor models is to represent a close to reality mathematical model which incorporates the noises and irregularities of sensor' readings along with the necessary axes transformations.

Actual sensors' model needs to take input as the physical stimuli quantities and gives the outputs which represent the real sensors' outputs like currents or voltages. Then, it should be the job of *flight software* to take current or voltage signals from sensors' model and perform ADC conversions to get the digital quantized readings of sensors and calculate the vectors and matrices in respective coordinate frames. But, due to the sake of simplicity and limited scope of this work, the sensors' models provide the vectors and matrices in respective coordinate frames.

3.4.1 Magnetometer

The magnetometer takes the magnetic field being calculated from environment block and transform it into body axes. It also induces the noises and bias that is faced by the real sensor. The sensor model is supposed to give magnetic field values in Body coordinate frame.

The Simulink representation of *Magnetometer* is shown in figure.

Input	Unit	Output	Unit
MagEarth (3×1 Magnetic Field Intensity Vector in NED Frame)	Tesla	B_{MGM} (3×1 Magnetic Field Intensity Vector in Body Frame)	Tesla
NED2Body (3×3 NED to Body transformation Vector)	-		

Table 3.6: Magnetometer Simulink Block Inputs and Outputs

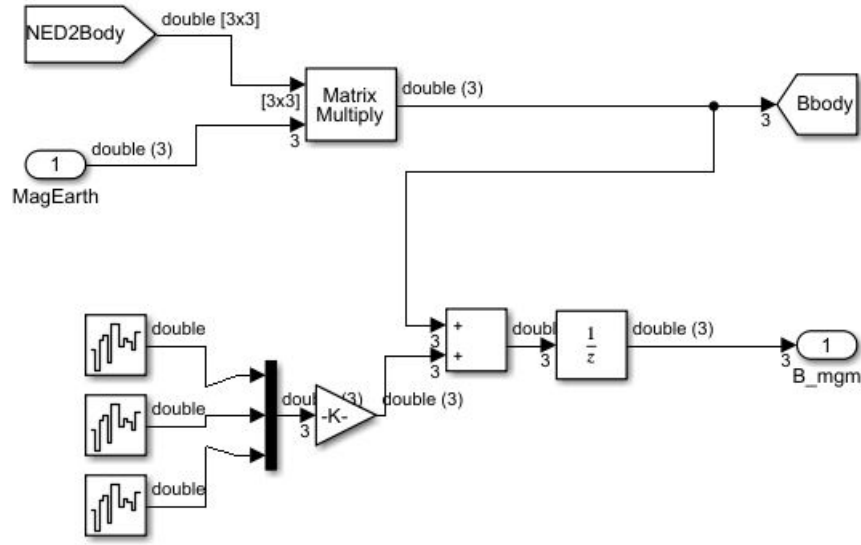


Figure 3.6: Magnetometer Simulink Block

3.4.2 Gyroscope

The purpose of gyroscope is to measure the angular velocities, which are also called body rates, in the body coordinate frame of satellite. In order to simulate the near-to-real effects of gyroscope, the colored noise model for bias is added with the readings coming from the dynamics block (ω_{rel}).

Input	Unit	Output	Unit
ω (3×1 Angular velocity in body axes of satellite)	rad/sec (radians per second)	ω_{SENS} (3×1 Angular velocity in body axes affected by noises)	rad/sec (radians per second)

Table 3.7: Gyroscope Simulink Block Inputs & Outputs

3.4.3 Sun Sensor

The purpose of sun sensor is to give the sun position vector in body coordinate frame of reference. In order to develop a mathematical model for the sun

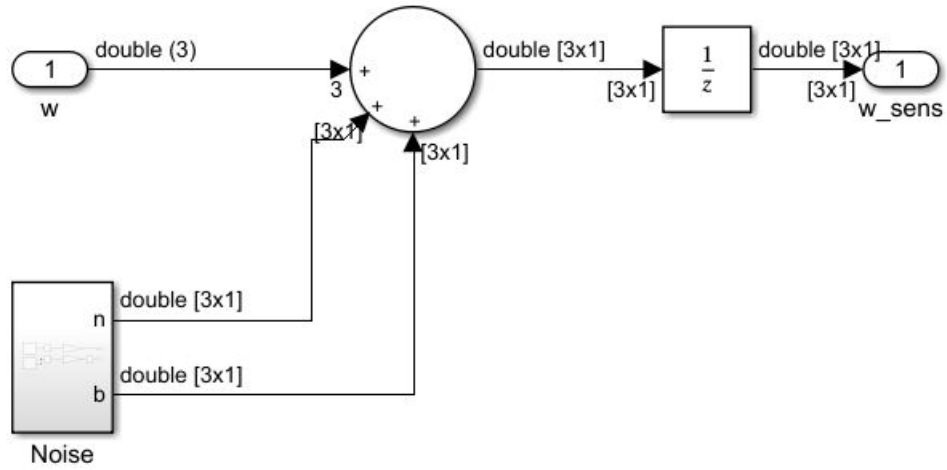


Figure 3.7: Gyroscope representation of Simulink Block

sensor, the sun position vector at the given time stamp is generated by sun position model and added with bias noises to represent the more realistic measurement model.

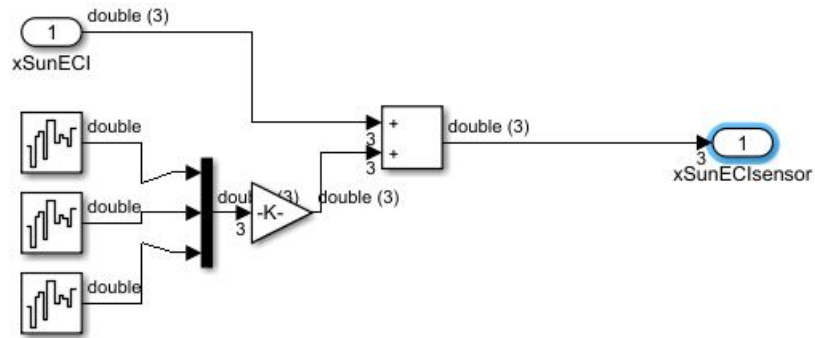


Figure 3.8: Sun Sensor representation of Simulink Block

Input	Unit	Output	Unit
xSunECI (3×1 Sun Position Vector in ECI Frame)	m (meters)	xSunECISensor (3×1 Sun Position Vector in ECI Frame affected by noises)	m (meters)

Table 3.8: Sun Sensor Simulink Block Inputs & Outputs

3.4.4 GPS

The main goal of Global Positioning System sensor is to generate position and velocity in ECEF frame of reference. The same is generated and taken as an input from the dynamics block. There is also an orbit propagator block, by simulink aerospace blockset, used in parallel in order to keep correcting the values of position and velocity generated by dynamics block or to make selection between *Orbit Propagator* or Dynamics Block position & velocity data. The orbit propagator block takes the epoch date of simulation and orbital parameters to propagate the position and velocity at every time stamp. Then, noise is added with the readings in order to represent the more realistic scenario.

The Simulink representation of *GPS* model is shown in figure. The *zero check* block makes sure that on the first instant of simulation, the zeros may not propagate to the next blocks.

Input	Unit	Output	Unit
xECEF (3×1 Position Vector in ECEF Frame)	m (meters)	xECEFGPS (3×1 Position Vector in ECEF Frame affected by noises)	m (meters)
vECEF (3×1 Velocity Vector in ECEF Frame)	m/s (meters per second)	vECEFGPS (3×1 Velocity Vector in ECEF Frame affected by noises)	m/s (meters per second)

Table 3.9: GPS Simulink Block Inputs & Outputs

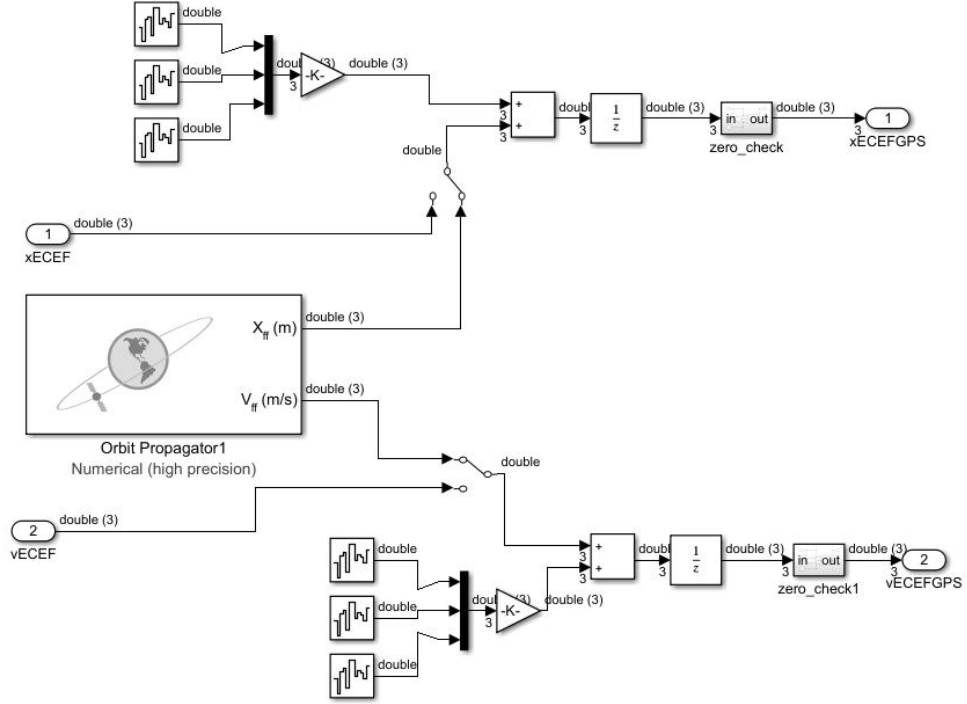


Figure 3.9: GPS representation of Simuink Block

3.5 Actuators

Actuators help in representing a mathematical approach to apply the required torque on body axes, which is calculated by the flight software in order to control the attitude of space craft or in order to damp the angular rates of satellite. In small satellites, unlikely the bigger satellite, the propulsion systems are not used in order to keep the mass budget lower. Thus, the other actuation methods although provide the necessary actuation, but are slower and less effective as compared to the propulsion system.

In real, actuators mathematical models should take the values of physical electrical stimuli like currents or voltages and calculate the respective torque commands. The conversion from torque to electrical stimuli should be taken into account by the *flight software*, but for the sake of simplicity and limited scope of the work, the actuators models take the torque or dipole vectors and performs calculations.

3.5.1 Magnetorquer Rods

The magnetorquer rods consist of a coil wound on ferromagnetic core or an air core. When the rod is energized (current is passed through the coil), it generates a magnetic dipole moment which depends on the number of turns of coil, the current passing through the coil and area of coil.

$$\vec{M} = NI\vec{A} \quad (3.7)$$

If that magnetic dipole is not aligned with earth's magnetic field, then a torque is produced. The resultant torque can be computed simply by taking the cross product of two elements.

$$\vec{\tau} = \vec{M} \times \vec{B} \quad (3.8)$$

In the study presented in this report, in order to have a control in all three axes of space craft, three magnetorquer rods are considered to be placed orthogonal to each other.

The flight software generates a set of *Dipole Commands* required to be generated by the torquer rods which are mapped on each of the magnetorquer rod. The magnetic field vector in Body frame B_{Body} is multiplied with the alignment matrix of each magnetorquer rod to compute the torque generated by each rod in body axes.

Input	Unit	Output	Unit
DipoleCMD (3×1 Dipole Commands for each of three magnetorquer rods)	$A - m^2$ (Ampere-square-meter)	Torque (3×1 Torque vector applied on each of body axis)	N-m (Newton-meter)
B_{BODY} (3×1 Magnetic Field Intensity vector in Body frame)	Tesla		

Table 3.10: Magnetorquer Rods Simulink Block Inputs & Outputs

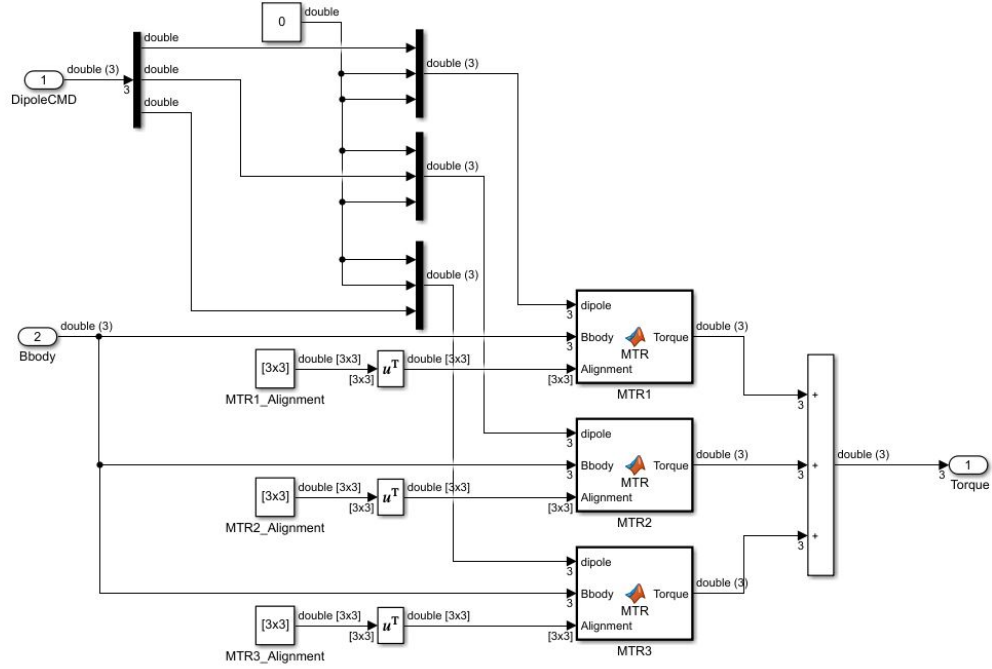


Figure 3.10: Magnetorquer Rods model Simulink representation

3.5.2 Reaction Wheels

Reaction wheels are the spinning wheels mounted on a motor placed on the satellite. Their speed can be increased and decreased as per requirement of resultant torque. A reaction wheel can provide torque on its axis of rotation. A tetrahedron configuration is assumed in this work that can provide the torque in any of three axes of satellite body. There are four reaction wheels assumed to be placed in tetrahedron configuration and they can withstand with one wheel failure. The advantage of this configuration is that the wheel assembly can provide twice as much of maximum torque on an axis that a single wheel can provide[2]. The tetrahedron configuration is shown in figure. The homogeneous distribution matrix of Reaction wheels in tetrahedron configuration is shown.

$$L = \begin{bmatrix} \sqrt{3}/3 & -\sqrt{3}/3 & -\sqrt{3}/3 & \sqrt{3}/3 \\ \sqrt{3}/3 & -\sqrt{3}/3 & \sqrt{3}/3 & -\sqrt{3}/3 \\ \sqrt{3}/3 & \sqrt{3}/3 & -\sqrt{3}/3 & -\sqrt{3}/3 \end{bmatrix} \quad (3.9)$$

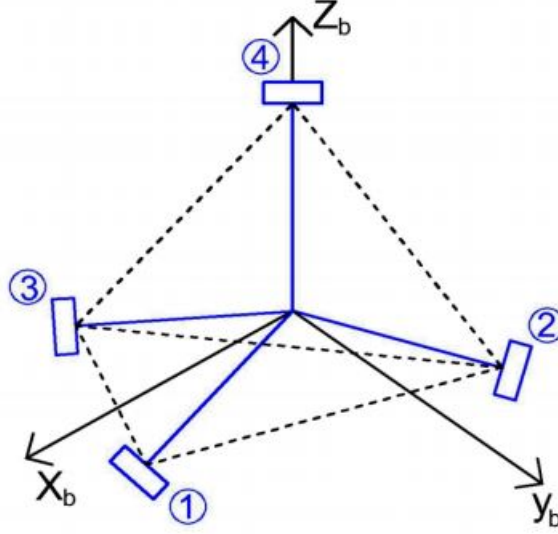


Figure 3.11: Tetrahedron configuration of Reaction Wheels[2]

The flight software generates a set of torque commands in body frame to be applied on satellite for the reaction wheels. These torque commands are mapped on the tetrahedron configuration of wheels by multiplying with distribution matrix. It is also assumed in this study, that the maximum torque that each wheel can produce is $5mN - m$, which is a common value for the commercially available reaction wheels for Cubesats. Therefore, the torque is limited to this value using saturation block and finally mapped back on to the body frame.

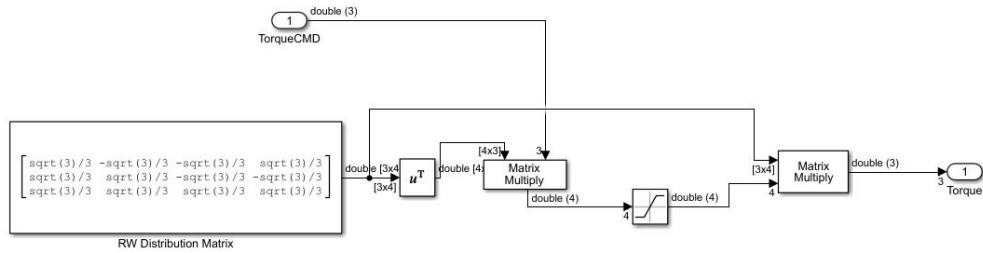


Figure 3.12: Reaction wheels representation of Simulink Model

Input	Unit	Output	Unit
TorqueCMD (3×1 Torque Commands required to be applied on body axes)	$N - m$ (Newton-meter)	Torque (3×1 Torque vector applied on each of body axis)	$N - m$ (Newton-meter)

Table 3.11: Reaction wheels Simulink Block Inputs & Outputs

De-Saturation of Reaction Wheels

The principle of reaction wheel is to transfer the total angular momentum of satellite between body and reaction wheels. The acceleration or deceleration can not be altered in an unlimited amount. The wheels, once accelerated and reached to their maximum speed, can not produce any more torque and are considered *saturated*. Therefore, a de-saturation of angular momentum of reaction wheels is needed. This action is also performed using magnetorquer rods. The torquer rods are required to produce torque in such axes so that they can off-load the reaction wheels and their speed can be decreased. This function is not modeled in this work and not presented in the mathematical modeling.

3.6 Flight Software

The flight software is the main Attitude Control software. It is the algorithm that takes the information of attitude and different required vectors from sensors and dynamics blocks and calculates the required torque to be applied on the actuators in order to maintain or control the attitude of satellite. This is the part of mathematical model which is then transformed into piece of software and is embedded in to the micro-controller or processor. The simplicity and execution efficiency of this block is of utmost concern. The output of this block is the required torque in body frame to control the orientation of space craft. The required commands are fed to the actuators and then the actuators block decides that how the torque will be acting on the body. There are three different mission modes considered in this work in order to better describe the functionality of attitude control software.

- Detumbling Mode

- Nadir Pointing Mode
- Sun Pointing Mode

Depending on the value of the *Mode* that can be set by user at run time, a *switch-case* diagram implements the selection of any of the modes. The only selected block is executed and the merge block transmits the only selected data. In case of *Sun Pointing* and *Nadir Pointing* mode, the output of Flight Software is torques to be applied on the body in order to correct the attitude error. In case of *Detumbling Mode*, the output of flight software is the Dipole Commands generated for each of the three magnetorquer rods placed at each of body axis.

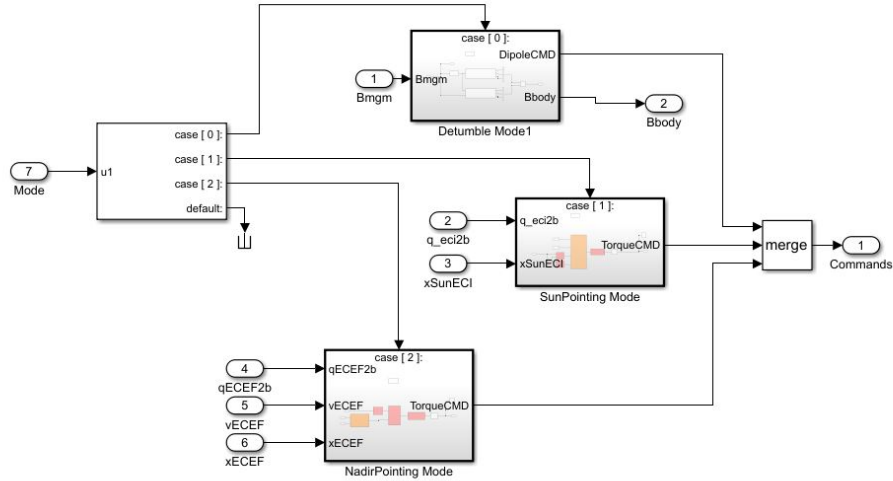


Figure 3.13: Flight Software Top Level hierarchy in Simulink

3.6.1 Detumbling Mode

When the satellite is separated from the launcher and is ejected into the orbit, the initial angular rates along the axes of body are unexpected and uncertain. In most cases, the magnitude of these angular rates is around 5 to 10 degrees per second. In order to start the normal mission life for the satellite, it is required to damp these rates under certain threshold. The

threshold is dependent on the mission analysts' requirements and is decided such that it can allow the satellite to start its mission modes.

In order not to drain the battery power and to perform the detumbling in a power efficient way, the magnetic control is used. Only magnetometer and magnetorquer rods are working in this mode. The basic magnetic control law is called *b-dot bang bang law*. The theory of law is that the derivative of magnetic field intensity is calculated at each time stamp and the maximum torque is applied in the opposite direction to that. This law is efficient mostly in detumbling mode. The law can be written as:

$$M_{b\dot{}} = -M_{max}.sign(\dot{B}_b) \quad (3.10)$$

The maximum saturation dipole, that can be provided on each body axis is $6Am^2$. This value is chosen as the normal range of dipole for commercially available torquer rods for cubesats and small satellites [3].

There is another method of implementing detumble control, which is the *proportional gain bdot* controller. In this law, a constant positive gain is multiplied with the normalized vector of rate of change of the magnetic field. The *B-dot proportional control law* could be written as[3] :

$$M_{b\dot{}} = -k_b \cdot \frac{\dot{B}_b}{||B_b||} \quad (3.11)$$

The value of the proportional gain k_b is chosen as 1024 [3]. The *B-dot Proportional Control law* is considered in this report for detumbling mode and the results of same are presented in the results chapter. For Simulink model development, both of these control laws are presented and one can be chosen using *switch*.

3.6.2 Sun Pointing Mode

In this mission mode, the satellite is made to point towards the sun so that solar panels may charge the batteries. This is a vital and basic mission mode which is necessary to achieve and maintain in every satellite.

This block relies on the quaternions error calculation which is part of the Cubesat Simulation library. There are two alignment vectors and two reference vectors are provided to the block. It tries to align the first reference vector with first alignment vector and then it tries to align the second

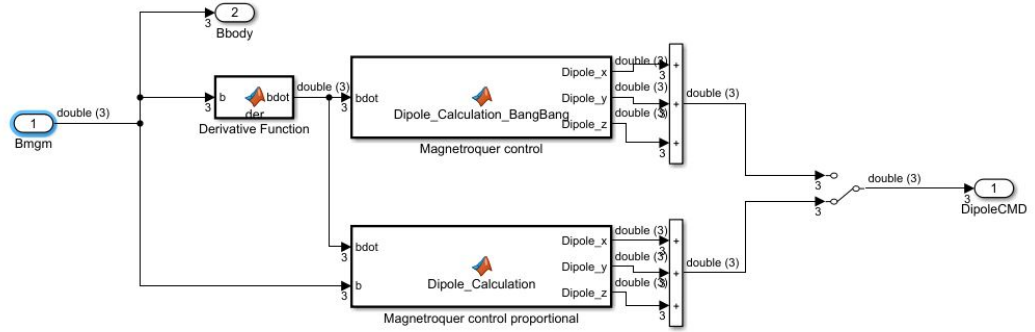


Figure 3.14: Detumbling Mode Matlab/Simulink Block

Input	Unit	Output	Unit
B_{MGM} $(3 \times 1 \text{ Magnetic Field Intensity Vector from Magnetometer})$	Tesla	DipoleCMD $(3 \times 1 \text{ Dipole Command vector in each of body axis})$	$A - m^2$ (Ampere-square-meter)
		B_{BODY} $(3 \times 1 \text{ Magnetic Field Intensity Vector from Magnetometer})$	Tesla

Table 3.12: Detumbling Mode Simulink Block Inputs & Outputs

reference vector with second alignment vector. The first align vector is $-z$ axis of satellite, which is required to face towards sun. Similarly, the first reference vector is the sun vector rotated by the attitude quaternions between ECI frame and Body frame.

$$FirstAlignmentVector = [0 \quad 0 \quad -1] \quad (3.12)$$

The second alignment vector is the x -axis of the satellite which is required to align with the z -axis of ECI frame, roughly called north pole of the earth.

$$SecondAlignmentVector = [1 \quad 0 \quad 0] \quad (3.13)$$

The error quaternions are calculated and then transformed into Euler angles. A PID controller takes the error Euler angles as input and tries to calculate the required torque commands. These commands are then directed to the actuators for further actions. The gains of PID controller are as follows:

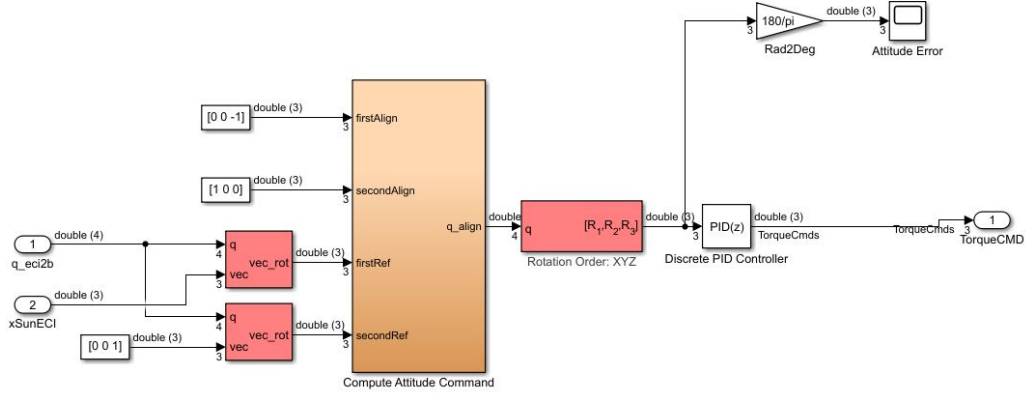


Figure 3.15: Sun Pointing Mode Simulink Model

$$K_P = 1.00 \times 10^{-4}$$

$$K_I = 2.00 \times 10^{-10}$$

$$K_D = 0.02$$

Input	Unit	Output	Unit
xSunECI (3×1 Sun position Vector in ECI frame)	m meters	TorqueCMD (3×1 Torque Command vector in each of body axis)	$N - m$ (Newton-meter)
qECI2b (4×1 Quaternions between Body frame to ECI frame)	-		

Table 3.13: Sun Pointing Mode Simulink Block Inputs & Outputs

3.6.3 Nadir Pointing Mode

This mission mode tries to point the satellite face containing scientific equipment towards the earth, considering it is a LEO satellite with imaging payload. This mode relies on the position and velocity information in ECEF frame generated by GPS and the quaternions generated by dynamics block between body frame and ECEF frame.

Firstly, the quaternions between position and velocity is calculated which can be represented as quaternions between ECEF frame and ORF (Orbital Reference Frame). Then, the quaternions multiplication is performed in order to calculate the quaternions angle between body frame and ORF frame, which is the actually quaternion error. This is then transformed into Euler angles and a similar PID controller tries to minimize this error by generating the required torque for the actuators. The gains of PID controller are as

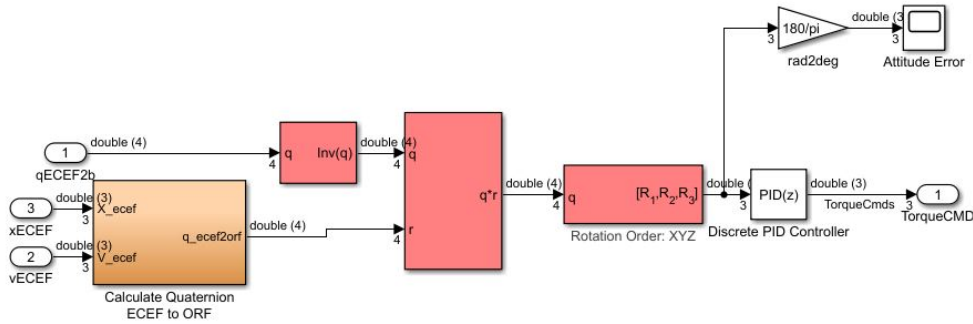


Figure 3.16: Nadir Pointing Mode Simulink Model

follows:

$$K_P = 1.00 \times 10^{-4}$$

$$K_I = 2.00 \times 10^{-10}$$

$$K_D = 0.02$$

3.6.4 Visualization

Matlab Simulink provides an interface to visualize three-dimensional animation of the satellite orbit and attitude maneuvers. There is an already developed Library model for animation provided with Simulink CubeSat

Input	Unit	Output	Unit
xECEF (3×1 Satellite Position Vector in ECEF frame)	m meters	TorqueCMD (3×1 Torque Command vector in each of body axis)	$N - m$ (Newton-meter)
vECEF (3×1 Satellite Velocity Vector in ECEF frame)	m/s meters per second		
qECEF2b (4×1 Quaternions between Body frame to ECEF frame)	-		

Table 3.14: Nadir Pointing Mode Simulink Block Inputs & Outputs

simulation that takes the information of sun position vector in ECI frame, attitude of space craft with reference to ECEF and ECI frames, earth rotation parameters and space craft position in ECEF frame of reference. This block creates a 3D model to visualize the motion and maneuver of satellite updated at each time stamp. The mission modes and angular rates of body can be analyzed in great convenience through it. In the following figure, a screenshot of a frame of animation is presented.

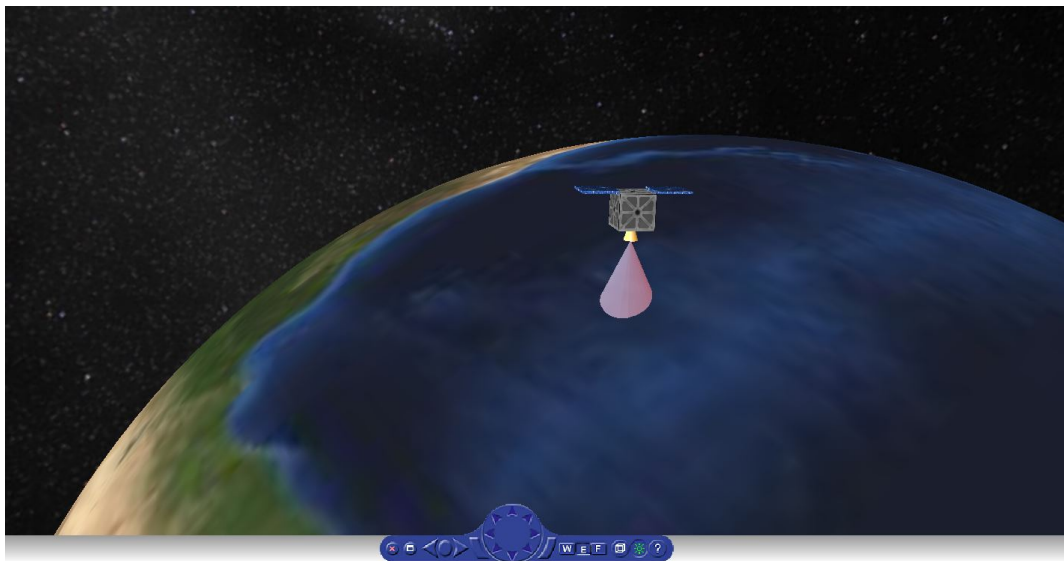


Figure 3.17: Three-Dimensional Visualization of satellite in Orbit

Chapter 4

Radiations Hardened Software

To develop a sophisticated mathematical algorithm for controlling the attitude of the satellite and embed it into the hardware is not enough for the space application. The space craft is constantly exposed to the huge amount of radiations and charged particles, which may not only damage the physical structure of the satellite but instead some of them may penetrate into the devices and cause transient and permanent damage. The hardware and software of the space craft both are vulnerable to the radiations' damage.

The developed mathematical model for attitude control, discussed in previous chapters, is made radiation hardened in order to tolerate the unexpected event of radiation hazards. Only the SEUs are considered and mitigated in this work and therefore the work methodology for same is discussed in this chapter.

4.1 Development of Flight Software

The part of the matlab/Simulink model that is supposed to be embed inside the hardware is the Flight Software for control algorithm. All the remaining parts of Simulink model are developed and simulated in order to provide the realistic scenario for the development and testing of Flight Software. The flight software is developed in the C++ language and tested in order to make sure that it provides the same required functionalities and then the *radiation hardening technique* is employed in order to make the system work properly

in the presence of radiations.

In this section, the development of flight software and its major components and hierarchy is discussed.

4.1.1 Data Structures

There are few global data structures defined to implement the flight software code. The purpose of global declaration of these data structures is convenient accessibility to data for every component of software by not passing the values or references between sub-routines.

The following data structures are defined in this study:

Data Structure	Description
ExtUVehicleT	External Inputs
ExtYVehicleT	External Outputs
ConstBVehicleT	Constants Data structure
DWVehicleT	States variables Data Structure
BVehicleT	Block Signals which are defined for intermediate operations on data

Table 4.1: List Of Data Structures in C++ Flight Software

4.1.2 Software Hierarchy

The software is divided into different subroutine files in order to better understand, debug and modify the code. The hierarchy can be adopted for any mission algorithm. List of important files is given:

- **FlightSoftware:** This function is called from the Simulink s-function as the entry point, which further calls the other functions.
- **Initialization:** It initializes all data structures with zeros or predefined constants.
- **Vehicle Step:** This function calls the relevant subroutine for mission modes depends on the mode selection using a *switch-case* scenario.

- **Detumbling Mode:** This function implements the *Detumbling mode* for the satellite mission.
- **Nadir Pointing:** This function implements the *Nadir Pointing mode* for the satellite mission.
- **Sun Pointing:** This function implements the *Sun Pointing mode* for the satellite mission.
- **Vehicle.h** stores the definitions of all data structures to be used in the flight software.

4.2 Development of Radiation Hardened Flight Software

In the work presented here, only the flight software block of closed loop model is converted into C++ language software code, as this is the only part that has to be embedded into the hardware and face the radiation noises. This code is then analyzed in detail in order to employ the effect of SEUs and its mitigation techniques.

The development of radiation hardened software has two major parts:

- Study the effect of Single Event Upset on embedded software and model it accordingly on the flight software.
- Integrate the radiation hardening technique to make the flight software tolerant against radiation events.

4.2.1 Radiation Effects on Software

Considering the software aspect, the radiations can corrupt the data stored in the memory devices by bit flipping. This effect is usually dealt in great precision by data scrubbers that constantly correct and update the corrupted data with the help of several error correction codes.

Radiations can also corrupt the running software by flipping some bits in the micro-controller when the software is running. This problem is unavoidable in some cases and may cause them serious errors. This can cause the code to jump to an unexpected memory locations or any unwanted routines may execute.

In the work presented in this report, the bit flipping event is simulated by calling a subroutine that randomly flips any bit of any of the running data structure. This method is a close approximation to what can happen during the SEU event during the flight.

The probability of bit flipping event can be controlled from outside by the user. User can choose any number zero or greater than zero, and in result of that, the subroutine that causes SEU will be called that many times at each time stamp.

4.2.2 Radiation Hardening Library

There is a Radiation Hardening Library provided, written in C++ programming language, which is discussed in the chapter 1, that provides the interface to the user to declare a special object of any data type which maintains the triple redundancy on the level of variable. The example of variable declaration is shown:

$$\textit{HardenedData} :: \textit{TripleData} < \textit{Datatype} > \textit{Variable}; \quad (4.1)$$

The three copies of data can be accessed as:

```
Variable.a();  
Variable.b();  
Variable.c();
```

Reading of the variable gives the majority voted value and writing the variable updates the three copies of data. All the basic operators are overridden to maintain the seamless operations of data in *TripleData.h* file.

There is also another file, *SingleData.h* provided which implements the same functions, but it only keeps one copy of data. The purpose of this file is to not change the software code when choosing *radiation hardened* or *unhardened* software. The example of variable declaration is shown:

$$\textit{HardenedData} :: \textit{SingleData} < \textit{Datatype} > \textit{Variable}; \quad (4.2)$$

The data can be accessed as:

```
Variable.a();
```

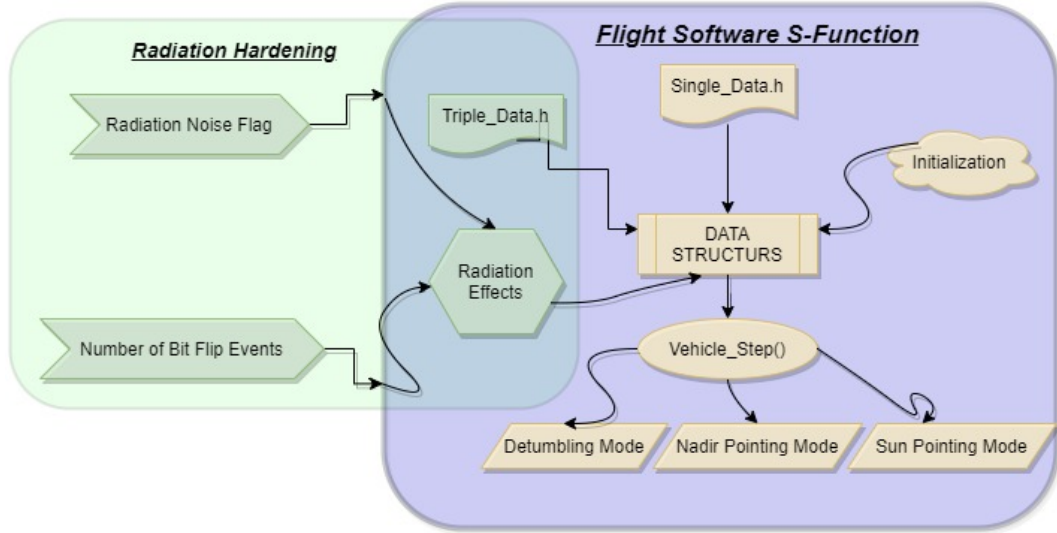


Figure 4.1: Radiation Hardened Software Hierarchy

4.2.3 Data Structures

The same data structures are kept for radiation hardened software. In order to model the radiation effects, every data component is accessible globally to the noise injection sub-routine. The only difference is that all variables in data structures are declared as *TripleData type* and *SingleData type*. A macro variable *RAD-HARD* can be defined or undefined which can select the definitions of *Triple Data* or *Single Data*.

4.2.4 Software Hierarchy

The radiation hardened flight software has few hierarchical changes as compared to the normal software. Following subroutines are added in order to model and support the hardening effects.

- **Radiation Effects:** This function implements the SEU subroutine for the modeling of bit flip event in the flight software.
- **TripleData.h** implements the overridden functions of all operators for the *Triple Data type* variables.
- **SingleData.h** implements the overridden functions of all operators for the *Single Data type* variables.

4.2.5 Radiation Effects Sub-Routine

The subroutine that causes the Bit Flipping is called every cycle if the flag *Radiation Noise Flag* is set to 1 by user. This sub routine is able to corrupt the value of any element of any data structure. It is called just before the *Vehicle Step* so that the corrupted data will affect the most part of software code. This subroutine generates a random number at each call and selects a random data from any data structure and changes its values to the random number.

It is also possible to control the number of calls to this subroutine in each cycle by user by setting the external input variable *Number of Bit Flip events*. This will define the probability of Bit Flipping event. In order to corrupt the data, a copy of data is corrupted by accessing its one copy. In order to employ the smooth transition of definition between *Triple Data* and *Single Data*, the copy of data is corrupted like the following:

$$Variable.a() = RandomNumber;$$

4.2.6 Radiation Tolerance

To fight against the effect of Radiations, the data structures are declared with each item in the way mentioned above as *TripleData*. In this way, the data corruption only occur to one of the copy of any variable while the other two copies remain valid. Although, the software code will take three times more space for data structures, but the whole flight software will be hardened against radiations effects. More number of bit flipping per cycle will cause the corruption of more variables in each cycle. But, the radiation hardened software will keep tolerating this effect.

4.2.7 S-functions Implementation

S-functions feature of the Matlab/Simulink is a gateway between C/C++ or VHDL code and Simulink model. In the work presented here, the flight software block of Simulink is converted into generic C++ code and the radiation hardening technique, presented above, is implemented and tested. Then an s-function is created using *s-function builder* in Simulink which calls the C++ function through Simulink. To the other blocks, it behaves like a black box and acts like a normal subsystem block that takes the same inputs as the Flight Software, but inside it runs the C++ code. User can select

whether to inject the radiation effects or not, and how much bit flip events per cycle can happen can also be controlled by sliding mode gain controller. When the *Radiation Noise Flag* is set to zero, the *Number of Bit Flip Events* do not have any effect.

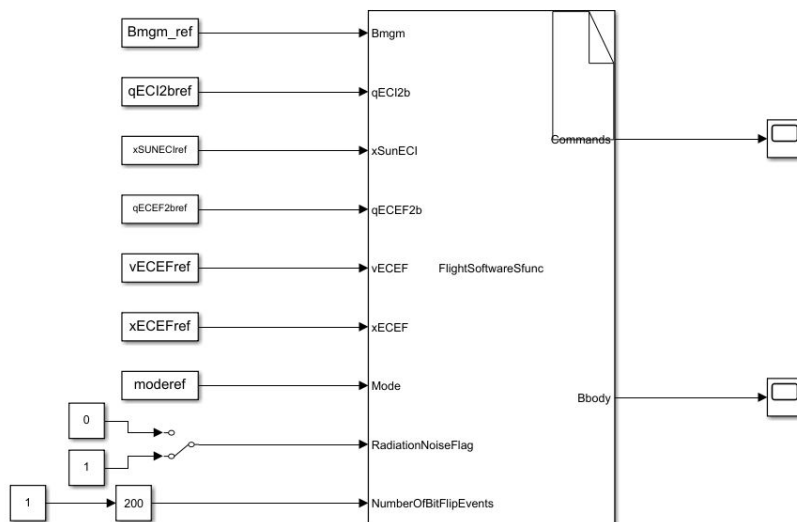


Figure 4.2: S-function Builder Representation of Flight Software

Chapter 5

Simulations and Results

This chapter summarizes the results of simulations and mathematical modeling performed during development and testing of radiation hardened attitude control algorithm.

5.1 Matlab & Simulink Results

In this section, the simulation results of Attitude Control Algorithm without the presence of radiations and without any radiation hardening technique are presented.

5.1.1 Development Platforms & Tools

Matlab/Simulink platform is used for development and testing of closed loop simulation of attitude control algorithm. For the following results, the *s-function* is not the part of loop for the simulations.

5.1.2 Mission Modes

In order to better visualize the results of the simulations, it is better to keep the mission modes separated and analyze their outcomes.

Detumbling Mode

Initially, when the satellite is separated from the launcher, it is assumed that it has certain angular rates along its body, which is usually 5 to 7 degrees

per second. The normal missions are assumed to be detumbled when their rates have dropped below 0.5 degrees per second in each axis, but it depends on the control as well as the mass of the satellite. Due to the very basic control of rate of change of magnetic field, the detumble *bdot* law is useful when the satellite has smaller angular rates. In the study presented in this report, the initial angular rates along the x-y-z body axes are following:

$$\omega_b = [5 \quad 4 \quad -0.0517] \quad \text{deg/sec} \quad (5.1)$$

The following is the simulation result achieved for *Detumble* mode. The angular rates estimation along the body axes is the observation of this mode. It can be seen in the figure that in the start when the angular rates are high, the control law tends to act stronger, but eventually it slows down the rates along the all body axes within 0.5deg/sec . The control law took almost 6 hours to damp the rates.

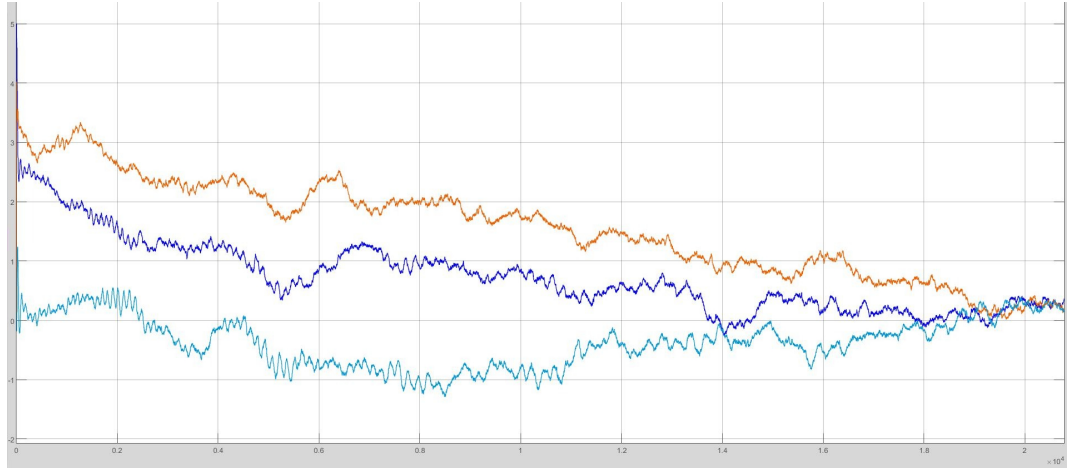


Figure 5.1: Detumbling Mode Simulation Results: Angular rates along xyz body axes

Nadir Pointing Mode

In this mode, the control algorithm is required to generate a set of torque commands in order to keep pointing the z-axis of satellite towards the center of earth. When the detumbling mode is succeeded in achieving the angular rates below a mission specific certain threshold, the nadir pointing mode can be switched to point the scientific equipment towards center of earth.

The angular rates at the start of this mode in the body are considered as $0.4deg/sec$ in x and y axes, and $-0.0517deg/sec$ in z axis. This mode relies on the reaction wheels thus the required attitude is achieved relatively fast. The attitude error in the form of Euler angles can be seen in the figure that the pointing error is start to reduce very fast initially and under 1000 seconds, the attitude error is damped within the limits of 0.05 degrees which is equals to 180 arc-seconds. The zoom-in of the curve is also shown for 8000 seconds which represents the steady state error and shows that the attitude error remains under 0.05 degrees.

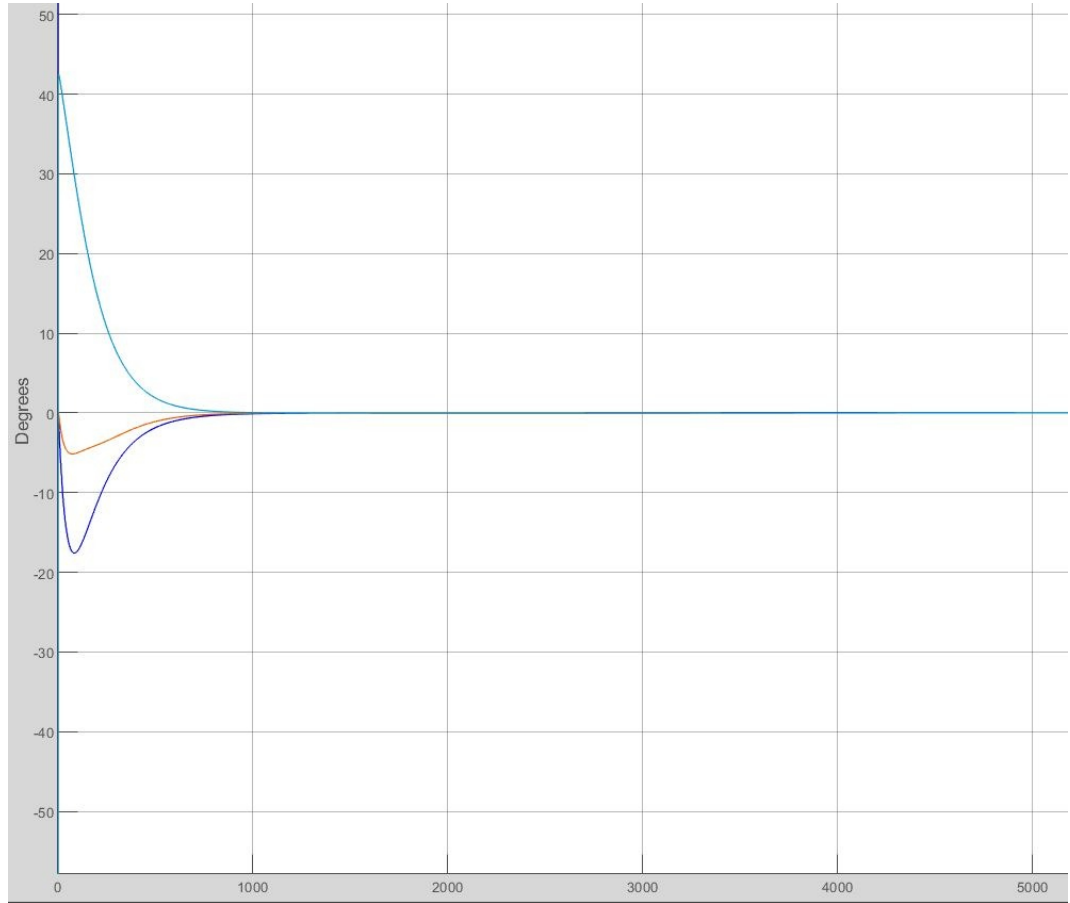


Figure 5.2: Nadir Pointing Mode Attitude Error: Euler angles

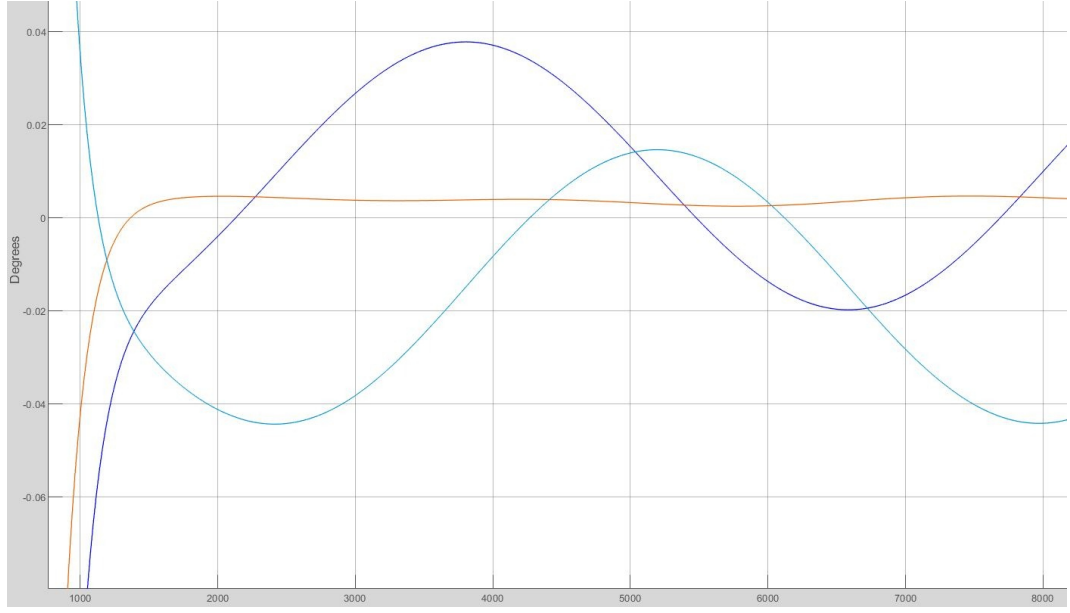


Figure 5.3: Nadir Pointing Mode Attitude Error steady state: Euler angles

Sun Pointing Mode

In this mode, the control algorithm is required to generate a set of torque commands in order to keep pointing the -z-axis of satellite towards the sun. This mode can be worked when the satellite is out of the solar eclipse. Therefore, it is considered that initially during the eclipse, the *Nadir Pointing Mode* is working, and when the satellite is out of the shadow, the *Sun Pointing Mode* is switched. As this mode also relies on the reaction wheels, so the attitude error is reduced faster than detumbling mode.

In the figure, it can be seen that around 500 seconds after the simulation started, the mode is switched to *Sun Pointing Mode*, and then after 2500 seconds, the attitude error reduced within 0.01 degrees which is equals to 36 arc seconds. Then, the steady state response of the controller can be seen in the other figure that the error remains same throughout the simulation.

5.2 Radiation Effects

When the radiation noise injection flag is set from the Simulink, the SEU subroutine is called at each timestamp and a randomly selected data value is corrupted each cycle in order to visualize and simulate the exaggerated

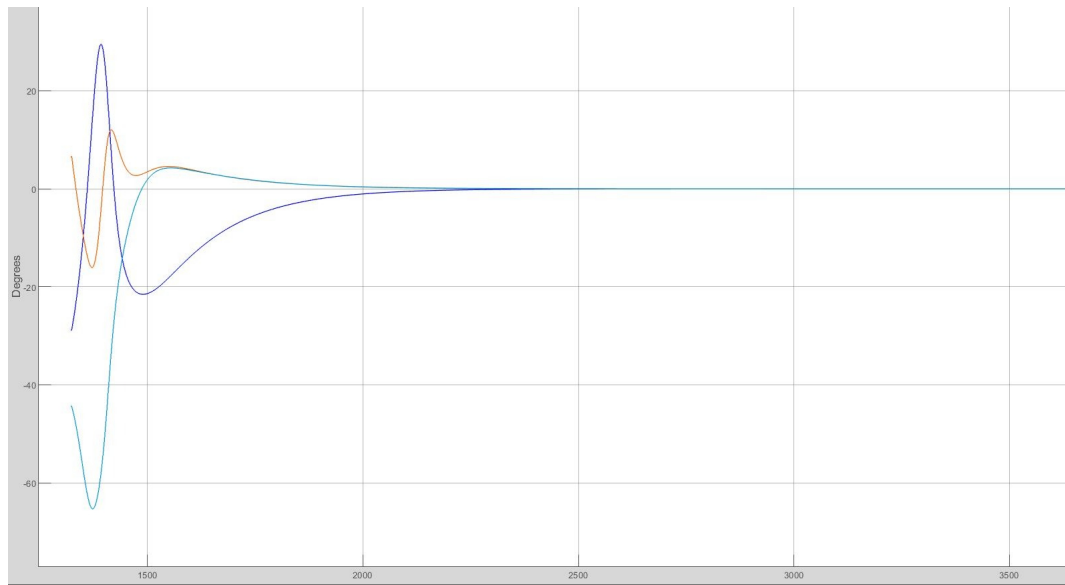


Figure 5.4: Sun Pointing Mode Attitude Error: Euler angles

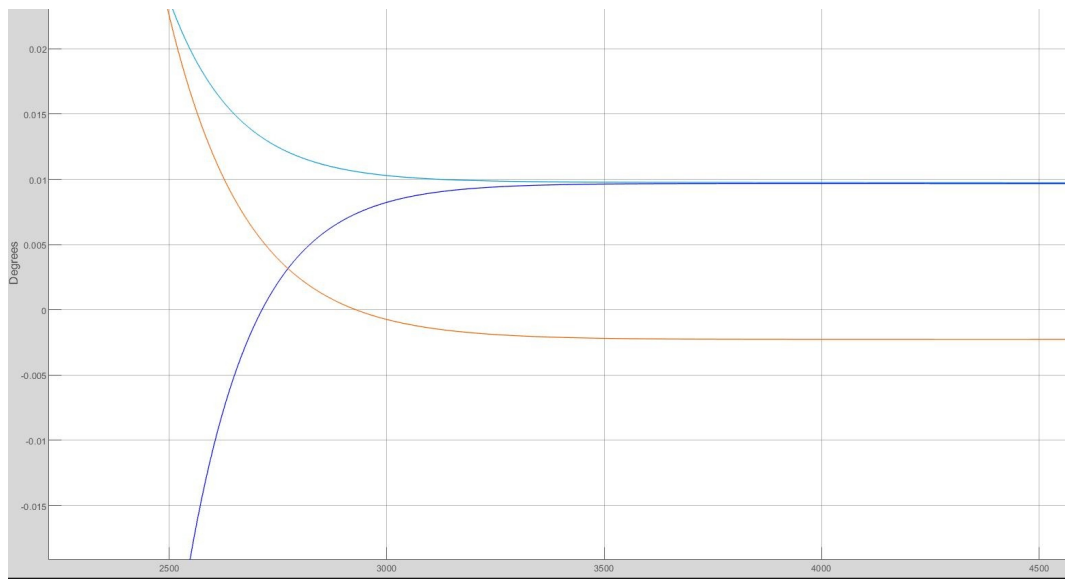


Figure 5.5: Sun Pointing Mode Attitude Error steady state: Euler angles

effect of radiations.

5.2.1 Mission Modes Simulation Results

The un-hardened software code will not be able to withstand the bit flipping of SEU. The results can be seen in the simulations.

Detumbling Mode

Detumbling mode will not be able to detumble the satellite towards threshold rates due to the radiation noises. The data structures employ the *Single Data* header file for data elements declaration and the *Radiation Noise Flag* is set to 1 in order to inject noises. *Number of Bit Flip Events* is also set to 1 which is enough to corrupt the software. As the bit flipping event simulation causes the random number assignment to the victim data, therefore, the closed loop simulation will not be able to proceed with the huge values of B_{Body} and *Commands*. Therefore, it is necessary to put a saturation block on the output of s-function block. The general trend shows that the maximum values of B_{Body} are in the range of -5×10^{-5} to 5×10^{-5} , and the maximum values of *Commands* can vary from -10 to 10. The saturation block limits the output of s-function block under these values.

In the following figure, the detumbling mode simulation is shown which represents that the flight software is unable to detumble the body angular rates due to the constant bit flipping events.

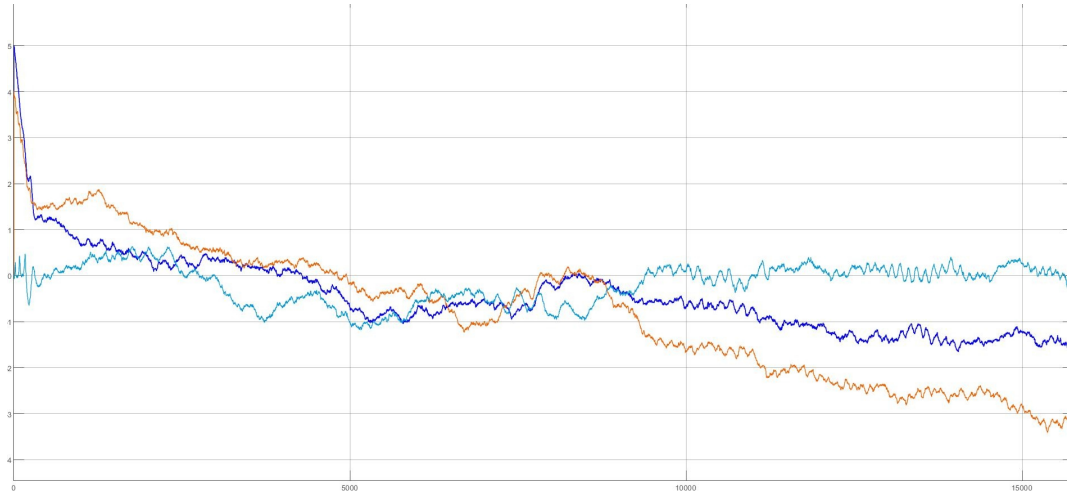


Figure 5.6: Radiations affected Detumbling Mode Simulation Results: Angular rates along xyz body axes

Nadir Pointing Mode

Nadir Pointing mode accuracy and efficiency can be seen as the attitude error after the noise effects as in the following figure. The attitude error should ideally reduce to minimum values after few seconds but due to the bit flipping events every cycle, the attitude error changes every second. Therefore, unwanted torques will be produced every second and the required attitude will not be achieved. In the following figure, when the *Radiation Noise Flag* is set to 1, the attitude error constantly changes to different values. Therefore, it is impossible to achieve the required attitude.

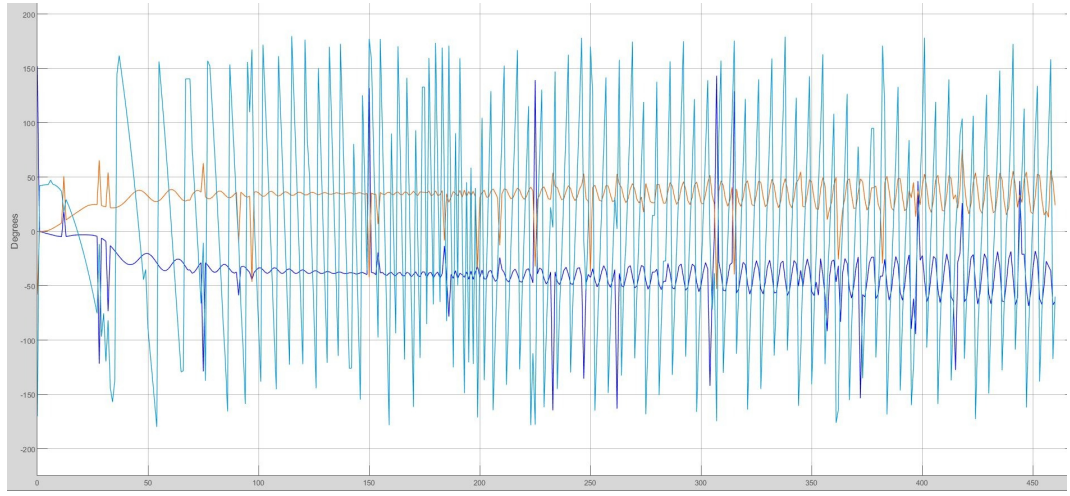


Figure 5.7: Radiations affected Nadir Pointing Mode Simulation Results: Euler angles

Sun Pointing Mode

Sun Pointing mode accuracy and efficiency can be seen as the attitude error after the noise effects as in the following figure. When the satellite is out of the eclipse, the mode is switched to Sun Pointing Mode and then *Radiation Noise Flag* is set to 1 for the bit flipping simulation. Due to the random bit flipping, the attitude error constantly changes and the unwanted torques are applied on the body. The satellite may experience unwanted abrupt huge torque commands to actuators and the desired attitude will be impossible to achieve.

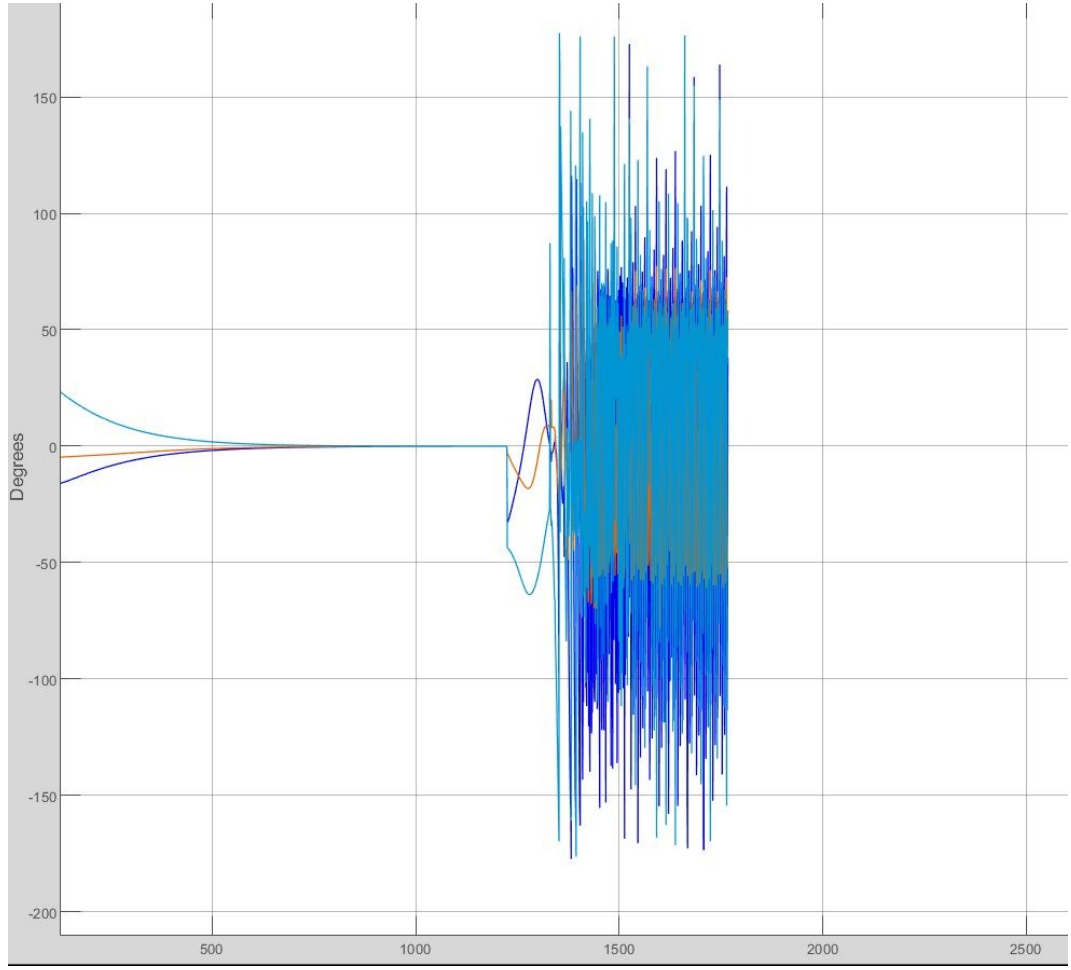


Figure 5.8: Radiations affected Sun Pointing Mode Simulation Results: Euler angles

5.3 Radiation Hardened Software

After implementing the radiation hardening technique in software, this version of control algorithm should be able to withstand the radiation noises. The SEU effects should be mitigated by the triple data redundancy technique.

5.3.1 Mission Modes Simulation Results

The hardened control algorithm for each mode can be analyzed by simulation results.

Detumbling Mode

The detumbling mode should be able to work normally and the rates along the body axes should tend to detumble even in the presence of radiation noises, as seen in figure. The initial angular rates along body were chosen same as previous detumbling mode. The *Radiation Noise Flag* is set to 1 externally and *Number of Bit Flip Events* is set to 2, which could be any value. The system withstand the radiation effects and performed the similar way as normal detumbling mode performed in the absence of radiation effects.

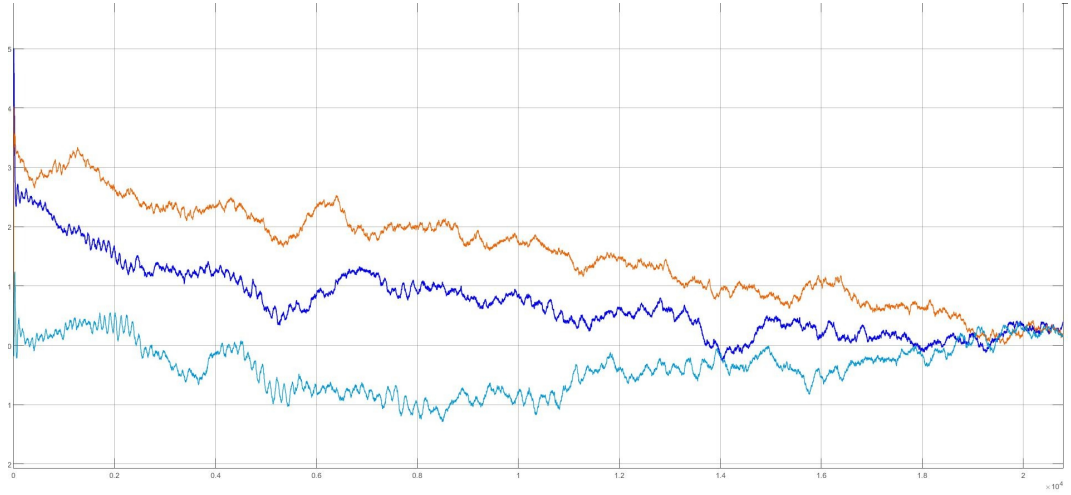


Figure 5.9: Radiation Hardened Detumbling Mode Simulation Results: Angular rates along xyz body axes

Nadir Pointing Mode

Nadir Pointing mode should work seamlessly as it was working without radiation noises. The hardening technique should tolerate the SEU effects and keep minimizing the attitude error effectively, as seen in figure. The initial conditions for this simulation is chosen as the previous. In the initial 1000 seconds, the attitude error reduced under 0.07 degrees which is equals to 252 arc seconds and for the rest of the simulation, the steady state error remains under 0.07 degrees.

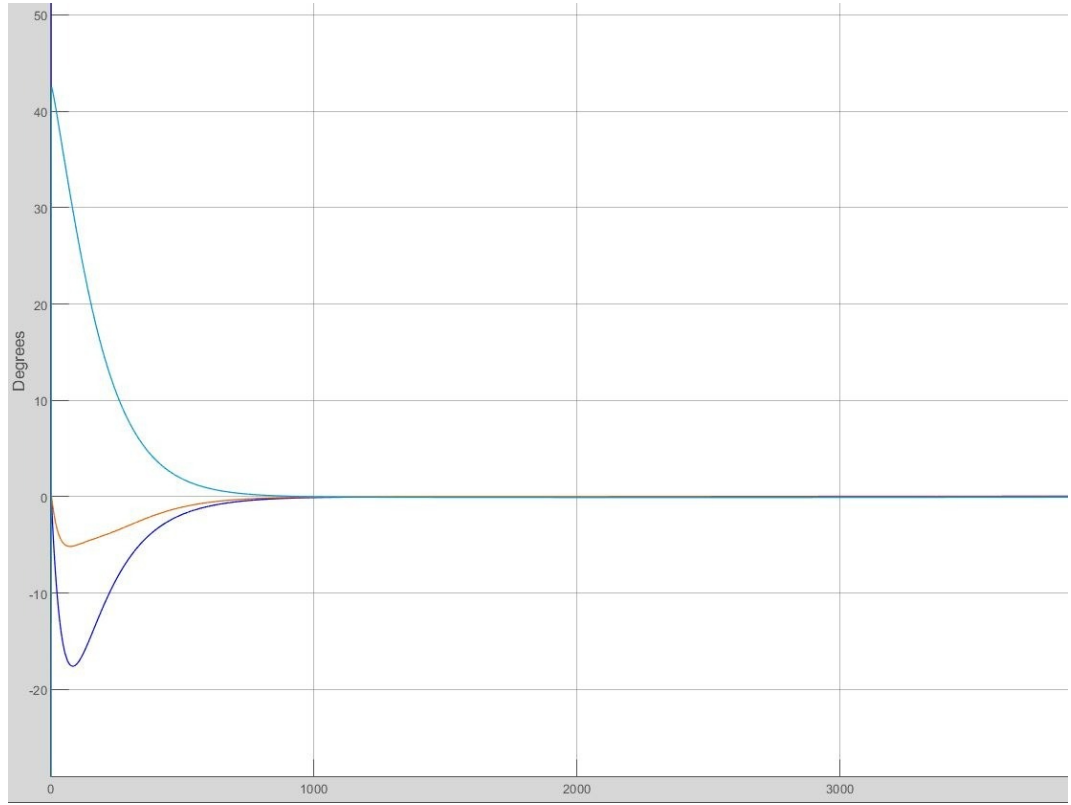


Figure 5.10: Radiation Hardened Nadir Pointing Mode Simulation Results: Euler angles

Sun Pointing Mode

Sun Pointing mode should work seamlessly as it was working without radiation noises. The hardening technique should tolerate the SEU effects and keep minimizing the attitude error effectively, as seen in figure. The Sun Pointing mode is switched as soon as the satellite came out of the solar eclipse.

After 1000 seconds, the mode is switched to *Sun Pointing*, and the attitude error is reduced under 0.035 degrees, which is 126 arc-seconds, after 3000 seconds. The attitude error remains same for the rest of simulation.

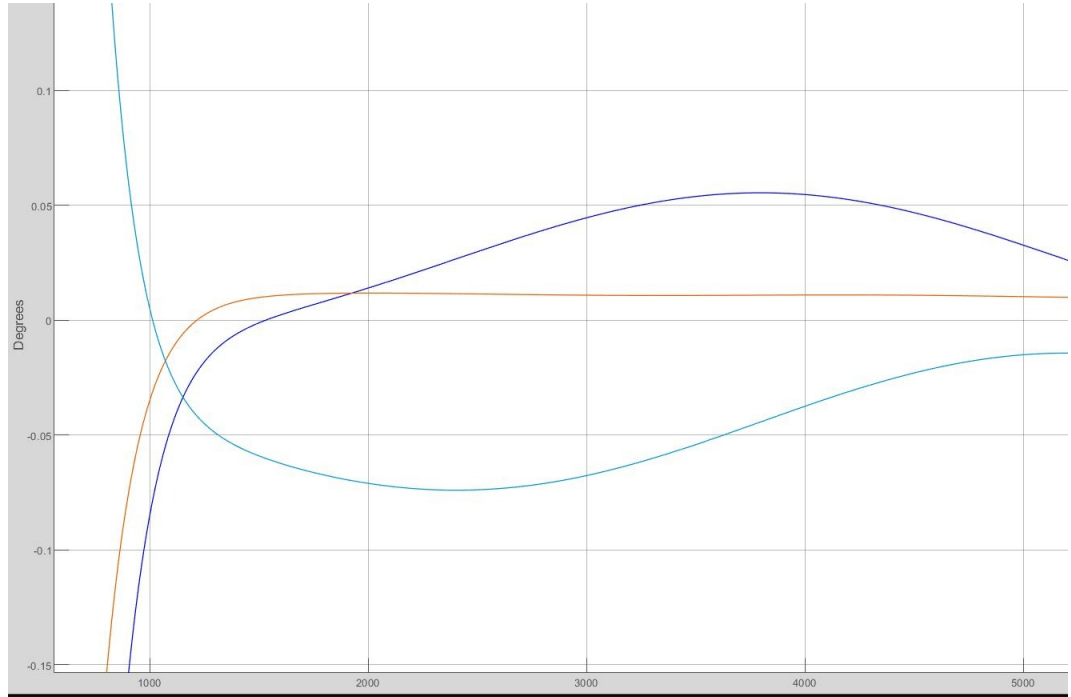


Figure 5.11: Radiation Hardened Nadir Pointing Mode steady state Simulation Results: Euler angles

5.4 Comparative Analysis

It can be seen from the results that the designed attitude control system, when working in the absence of radiations, achieved the required results. But, due to the SEU effects, the software affected badly and it could not proceed with the mission objectives in any of the mode. Then, the radiation hardening technique again achieved the required performance even in the presence of radiations.

The performance parameters comparison for each of the mission mode are summarized in the following tables.

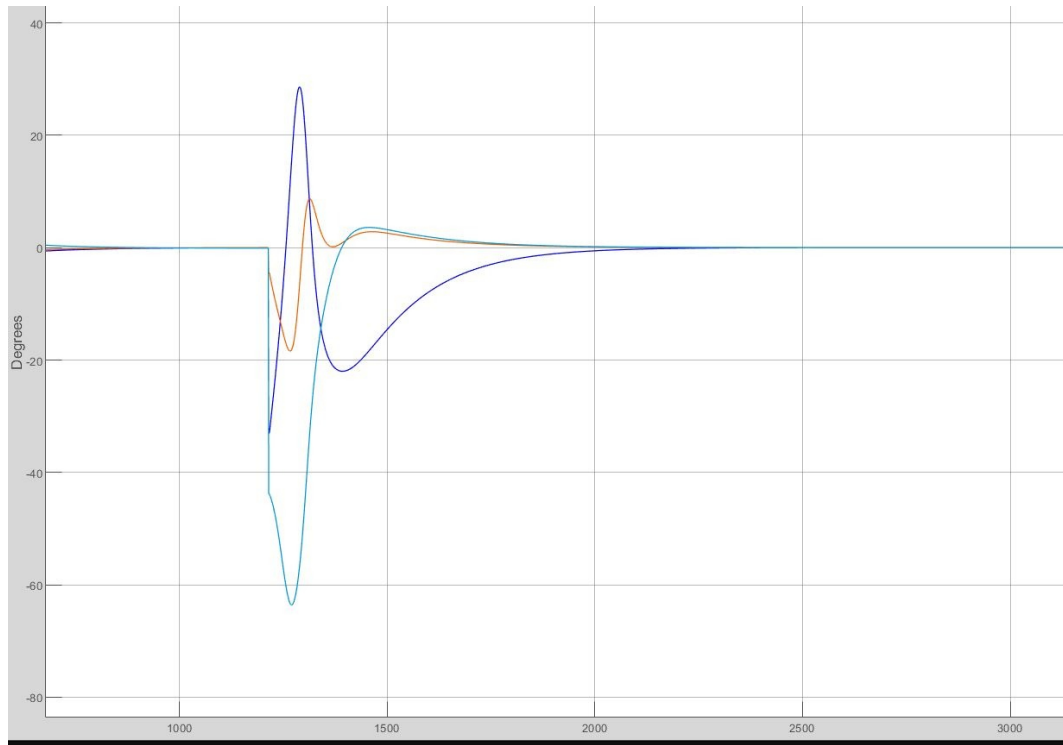


Figure 5.12: Radiation Hardened Sun Pointing Mode Simulation Results: Euler angles

Detumbling Mode	Damped Rates	Comments
Simulink Model	0.5 deg/sec	It took 6 hours to damp the rates withing threshold
Radiations Affected s-function	<i>Un-damped</i>	The angular rates of satellite remains constant around 3 deg/sec even after 6 hours
Radiations Hardened s-function	0.5 deg/sec	After 6 hours, the rates are damped within threshold

Table 5.1: Detumbling mode performance comparison

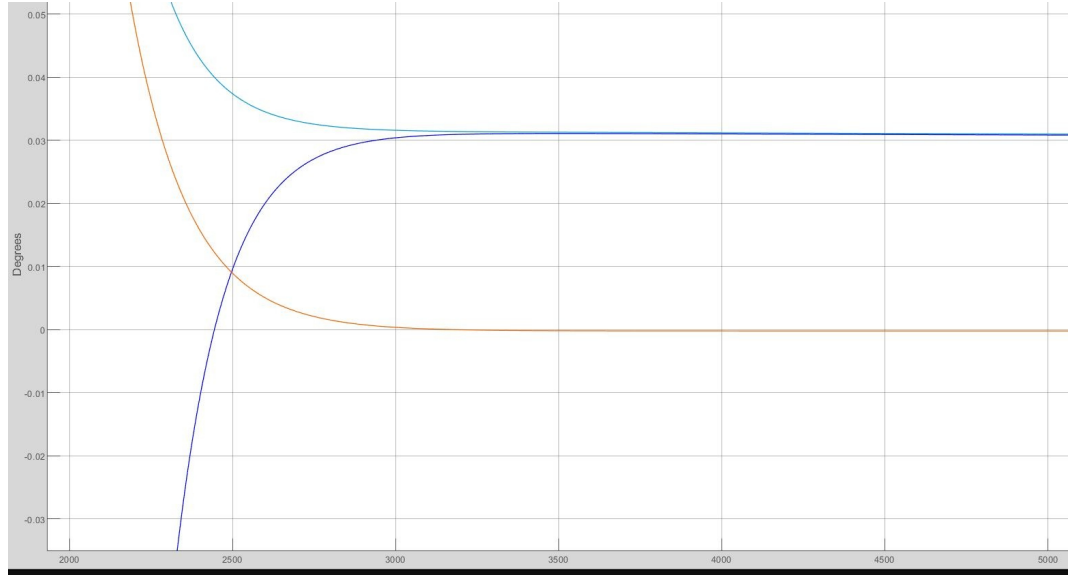


Figure 5.13: Radiation Hardened Sun Pointing Mode steady state Simulation Results: Euler angles

Nadir Pointing Mode	Pointing Accuracy	Comments
Simulink Model	180 arc-sec	After 1000 seconds, the attitude error remains under threshold.
Radiations Affected s-function	<i>Failed to point</i>	The attitude error constantly changes due to radiations and unwanted torques on the body make it impossible to achieve the desired attitude.
Radiations Hardened s-function	252 arc-sec	After 1000 seconds, the attitude error remains under threshold.

Table 5.2: Nadir Pointing mode performance comparison

Sun Pointing Mode	Pointing Accuracy	Comments
Simulink Model	36 arc-sec	After 2500 seconds, the attitude error remains under threshold.
Radiations Affected s-function	<i>Failed to point</i>	The attitude error constantly changes due to radiations and unwanted torques on the body make it impossible to achieve the desired attitude.
Radiations Hardened s-function	126 arc-sec	After 3000 seconds, the attitude error remains under threshold.

Table 5.3: Sun Pointing mode performance comparison

Chapter 6

Conclusion

In the work presented in this report, a basic attitude control system of a small satellite for low earth orbit mission is designed and simulated. *Detumbling Mode*, *Nadir Pointing Mode* and *Sun Pointing Mode* are developed and modeled using Matlab/Simulink. The closed loop mathematical model developed on Simulink achieves these basic functional modes of satellite. The *detumbling mode* damps the satellite angular body rates along x, y and z axes under 0.5 deg/sec. The magnetic control of this mode much longer time, but eventually settles down the body rates. The *Sun Pointing mode* and *Nadir Pointing mode* implements 3-axis stabilization and provides the pointing accuracy of 36 arc-sec and 180 arc-sec respectively.

The effects of radiations are considered that can affect the running software of the satellite, more precisely, the effects of Single Event Upset that can cause the bit-flipping in the software is considered and modeled. In order to analyze the effects of SEU, the control algorithm which is also called the flight software, is converted into C++ language code and the SEUs are modeled on that software. This software is plugged back into the Simulink model using *s-function* technique so that the closed loop simulation can analyze the performance of software in the presence of radiations. Upon the simulation of radiation effects, it is concluded that the random bit flipping of the software may lead the code flow into an unexpected state or can damage the data so badly that the several calculations of corrupted data can unstable the whole system with the failure of achieving mission objectives.

The software is then made hardened against the radiations using a software technique which triplicates the data. The radiation-hardened software runs in the closed loop Simulink model to prove that the software achieves the

required functionality and performance. The hardened software makes the system tolerable with the SEU events. The *detumbling mode* settles down the body angular rates along x, y and z axes of satellite under 0.5 deg/sec while the *Sun Pointing mode* and *Nadir Pointing mode* provides the pointing accuracy of 126 arc-sec and 252 arc-sec respectively, which are acceptable.

6.1 Future Work

The scope of flight software can be extended. The sensors' raw data may be taken as an input in the form of currents and voltages and then processed to determine the required vectors and matrices. Similarly, the flight software outputs should be converted into electrical stimuli which are voltage or current signals and given to the actuators.

The reaction wheel block should have a desaturation block in order to lower down the speeds of reaction wheels without affecting the momentum of satellite. The proper commands for magnetorquer rods can be calculated in order to implement desaturation function.

The disturbances acting on the satellite are not considered in this work. They can be modeled to make the system behave even more realistic.

The radiation effects modeling can be extended. The SEU event should randomly happen during the execution of the software. The radiation effects subroutine can be called by a software interrupt randomly during the execution of any function in order to more randomize the event probability. The development of code of flight software can be optimized by reusing many variables. But this might lead to increase in the probability of using the already corrupted data. On the other hand, defining long data structures and breaking down the operations into smaller operations increase the data size. The trade-off can be carefully chosen by analyzing the behaviour of system deeply.

Appendix A

Flight Software Source Code

FlightSoftware.cpp

```
1 #include "Vehicle.h"
2
3
4 /*The following is the declarations of date structures which are used in the flight software*/
5
6 /* Block signals: The local variables of the software to store temporary values and intermediate calculations*/
7 B_Vehicle_T Vehicle_B;
8
9 /* Block states: The state variables, that are required to store the values between different cycles*/
10 DW_Vehicle_T Vehicle_DW;
11
12 /* External inputs: The inputs of the flight software */
13 ExtU_Vehicle_T Vehicle_U;
14
15 /* External outputs: The outputs of the flight software */
16 ExtY_Vehicle_T Vehicle_Y;
17
18 /*Constants: These values are initialized in the start to store the constants*/
19 ConstB_Vehicle_T Vehicle_ConstB;
20
21
22
23
24 /*The entry point of the flight software. Simulink s-function calls this subroutine to execute the flight software at each
   time stamp
25 * Inputs: const double B_mgm[3],const double q_eci2b[4],const double xSunECI[3], const double qECEF2b[4],const double
   vECEF[3],
26 * const double xECEF[3],const double mode,unsigned int RadiationNoiseFlag, unsigned int NumberOfBitFlipEvents
27 * Outputs: double Commands[3], double Bbody[3], double AttitudeError[3]
28 * */
29 void FlightSoftware(const double B_mgm[3],const double q_eci2b[4],const double xSunECI[3],
30 const double qECEF2b[4],const double vECEF[3],const double xECEF[3],
31 const double mode,double Commands[3], double Bbody[3], unsigned int RadiationNoiseFlag, unsigned int
   NumberOfBitFlipEvents, double AttitudeError[3])
32 {
33
34 static int IdxState = 0;
35 unsigned int index = 0;
36
37 /* Initialization of structures need to be done only once*/
38 if (IdxState == 0){
39 Vehicle_initialize();
40 IdxState = 1;}
41
42 /*Reading external inputs */
43 Vehicle_U.Bmgm[0] = B_mgm[0];
44 Vehicle_U.Bmgm[1] = B_mgm[1];
45 Vehicle_U.Bmgm[2] = B_mgm[2];
46 Vehicle_U.q_eci2b[0] = q_eci2b[0];
47 Vehicle_U.q_eci2b[1] = q_eci2b[1];
48 Vehicle_U.q_eci2b[2] = q_eci2b[2];
49 Vehicle_U.q_eci2b[3] = q_eci2b[3];
50 Vehicle_U.xSunECI[0] = xSunECI[0];
51 Vehicle_U.xSunECI[1] = xSunECI[1];
52 Vehicle_U.xSunECI[2] = xSunECI[2];
53 Vehicle_U.qECEF2b[0] = qECEF2b[0];
54 Vehicle_U.qECEF2b[1] = qECEF2b[1];
55 Vehicle_U.qECEF2b[2] = qECEF2b[2];
56 Vehicle_U.qECEF2b[3] = qECEF2b[3];
57 Vehicle_U.vECEF[0] = vECEF[0];
58 Vehicle_U.vECEF[1] = vECEF[1];
59 Vehicle_U.vECEF[2] = vECEF[2];
60 Vehicle_U.xECEF[0] = xECEF[0];
61 Vehicle_U.xECEF[1] = xECEF[1];
62 Vehicle_U.xECEF[2] = xECEF[2];
63 Vehicle_U.Mode = mode;
64
65 /* Single Event Upset Function */
66 if (RadiationNoiseFlag == 1){
67 for(index = 0; index <NumberOfBitFlipEvents; index++)
68 SEU(); // SEU() will be called the number of times defined by NumberOfBitFlipEvents
69 }
70
71 /*Calling main function*/
72 Vehicle_step();
73
```

```

74  /* external outputs Mapping*/
75  Commands[0]= (real_T)Vehicle_Y.Commands[0] ;
76  Commands[1]= (real_T) Vehicle_Y.Commands[1] ;
77  Commands[2]= (real_T) Vehicle_Y.Commands[2] ;
78  Bbody[0]= (real_T) Vehicle_Y.Bbody[0] ;
79  Bbody[1]= (real_T)Vehicle_Y.Bbody[1] ;
80  Bbody[2]= (real_T)Vehicle_Y.Bbody[2] ;
81  AttitudeError[0] = (real_T)Vehicle_B.VectorConcatenate_a[0];
82  AttitudeError[1] = (real_T)Vehicle_B.VectorConcatenate_a[1];
83  AttitudeError[2] = (real_T)Vehicle_B.VectorConcatenate_a[2];
84  }

```

DetumblingMode.cpp

```

1  #include "Vehicle.h"
2
3  void Detumble()
4  {
5      real_T tmpForInput_idx_0;
6
7      /* Outputs for IfAction SubSystem: '<S1>/Detumble Mode' incorporates:
8       *   ActionPort: '<S2>/Action Port'
9       */
10     /* Outport: '<Root>/Bbody' incorporates:
11      *   Inport: '<Root>/Bmgn'
12      *   Inport: '<S2>/Bmgn'
13      */
14     /* MATLAB Function 'Vehicle Flight Software/Detumble Mode/Derivative Function': '<S5>:1' */
15     /* '<S5>:1:4' if isempty (state) */
16     /* '<S5>:1:7' bdot = b-state; */
17     /* '<S5>:1:8' state = b; */
18     Vehicle_Y.Bbody[0] = Vehicle_U.Bmgn[0];
19
20     /* MATLAB Function: '<S2>/Derivative Function' incorporates:
21      *   Outport: '<Root>/Bbody'
22      */
23     Vehicle_B.bdot[0] = Vehicle_Y.Bbody[0] - Vehicle_DW.state[0];
24     Vehicle_DW.state[0] = Vehicle_Y.Bbody[0];
25
26     /* Outport: '<Root>/Bbody' incorporates:
27      *   Inport: '<Root>/Bmgn'
28      *   Inport: '<S2>/Bmgn'
29      */
30     Vehicle_Y.Bbody[1] = Vehicle_U.Bmgn[1];
31
32     /* MATLAB Function: '<S2>/Derivative Function' incorporates:
33      *   Outport: '<Root>/Bbody'
34      */
35     Vehicle_B.bdot[1] = Vehicle_Y.Bbody[1] - Vehicle_DW.state[1];
36     Vehicle_DW.state[1] = Vehicle_Y.Bbody[1];
37
38     /* Outport: '<Root>/Bbody' incorporates:
39      *   Inport: '<Root>/Bmgn'
40      *   Inport: '<S2>/Bmgn'
41      */
42     Vehicle_Y.Bbody[2] = Vehicle_U.Bmgn[2];
43
44     /* MATLAB Function: '<S2>/Derivative Function' incorporates:
45      *   Outport: '<Root>/Bbody'
46      */
47     Vehicle_B.bdot[2] = Vehicle_Y.Bbody[2] - Vehicle_DW.state[2];
48     Vehicle_DW.state[2] = Vehicle_Y.Bbody[2];
49
50     /* MATLAB Function: '<S2>/Magnetorquer control' */
51     /* MATLAB Function 'Vehicle Flight Software/Detumble Mode/Magnetorquer control': '<S6>:1' */
52     /* '<S6>:1:3' mmax=6; */
53     /* '<S6>:1:6' Dipole_x= [-mmax*sign (bdot(1));0;0]; */
54     tmpForInput_idx_0 = (real_T)Vehicle_B.bdot[0];
55     /* if (tmpForInput_idx_0 < 0.0) {
56      *   tmpForInput_idx_0 = -1.0;
57      * } else if (tmpForInput_idx_0 > 0.0) {
58      *   tmpForInput_idx_0 = 1.0;
59      * } else if (tmpForInput_idx_0 == 0.0) {
60      *   tmpForInput_idx_0 = 0.0;
61      * } */
62
63     Vehicle_B.norm_b = sqrt (( Vehicle_Y.Bbody[0]*Vehicle_Y.Bbody[0])+(Vehicle_Y.Bbody[1]*Vehicle_Y.Bbody[1])+(Vehicle_Y.Bbody[2]*Vehicle_Y.Bbody[2]));
64
65     if (Vehicle_B.norm_b == 0.0)
66     {
67         Vehicle_B.norm_b = 1.732050807568877e-09;
68         //Vehicle_B.Dipole_x[0] = -6.0 * tmpForInput_idx_0;
69         Vehicle_B.Dipole_x[0] = -1024.0 * Vehicle_B.bdot[0]/Vehicle_B.norm_b;
70         Vehicle_B.Dipole_x[1] = 0.0;
71         Vehicle_B.Dipole_x[2] = 0.0;
72
73         /* '<S6>:1:7' Dipole_y= [0;-mmax*sign (bdot(2));0]; */
74         Vehicle_B.Dipole_y[0] = 0.0;
75         /* tmpForInput_idx_0 = (real_T)Vehicle_B.bdot[1];
76          * if (tmpForInput_idx_0 < 0.0) {
77          *   tmpForInput_idx_0 = -1.0;
78          * } else if (tmpForInput_idx_0 > 0.0) {
79          *   tmpForInput_idx_0 = 1.0;
80          * } else if (tmpForInput_idx_0 == 0.0) {
81          *   tmpForInput_idx_0 = 0.0;
82          * } */
83         //Vehicle_B.Dipole_y[1] = -6.0 * tmpForInput_idx_0;
84         Vehicle_B.Dipole_y[1] = -1024.0 * Vehicle_B.bdot[1]/Vehicle_B.norm_b;
85         Vehicle_B.Dipole_y[2] = 0.0;
86
87         /* '<S6>:1:8' Dipole_z= [0;0;-mmax*sign (bdot(3))]; */

```



```

88     Vehicle_B.Dipole_z[0] = 0.0;
89     Vehicle_B.Dipole_z[1] = 0.0;
90     /* tmpForInput_idx_0 = (real_T)Vehicle_B.bdot[2];
91     if (tmpForInput_idx_0 < 0.0) {
92         tmpForInput_idx_0 = -1.0;
93     } else if (tmpForInput_idx_0 > 0.0) {
94         tmpForInput_idx_0 = 1.0;
95     } else if (tmpForInput_idx_0 == 0.0) {
96         tmpForInput_idx_0 = 0.0;
97     }*/
98
99     //Vehicle_B.Dipole_z[2] = -6.0 * tmpForInput_idx_0;
100    Vehicle_B.Dipole_z[2] = -1024.0 * Vehicle_B.bdot[2]/ Vehicle_B.norm_b;
101
102    /* End of MATLAB Function: '<S2>/Magnetorquer control' */
103
104    /* Merge: '<S1>/Merge' incorporates:
105     * Sum: '<S2>/Add'
106     */
107    Vehicle_B.Merge[0] = (Vehicle_B.Dipole_x[0] + Vehicle_B.Dipole_y[0]) +
108        Vehicle_B.Dipole_z[0];
109    Vehicle_B.Merge[1] = (Vehicle_B.Dipole_x[1] + Vehicle_B.Dipole_y[1]) +
110        Vehicle_B.Dipole_z[1];
111    Vehicle_B.Merge[2] = (Vehicle_B.Dipole_x[2] + Vehicle_B.Dipole_y[2]) +
112        Vehicle_B.Dipole_z[2];
113
114    /* End of Outputs for SubSystem: '<S1>/Detumble Mode' */
115
116 }

```

NadirPointing.cpp

```

1  #include "Vehicle.h"
2
3  void NadirPointing()
4  {
5      real_T tmpForInput_idx_0;
6      real_T tmpForInput_idx_1;
7      real_T tmpForInput_idx_2;
8      int32_T i;
9      /* Outputs for IfAction SubSystem: '<S1>/NadirPointing Mode' incorporates:
10       * ActionPort: '<S3>/Action Port'
11       */
12      /* Product: '<S20>/j x k' incorporates:
13       * Inport: '<Root>/vECEF'
14       * Inport: '<Root>/xECEF'
15       */
16      Vehicle_B.jxk = Vehicle_U.vECEF[1] * Vehicle_U.xECEF[2];
17
18      /* Product: '<S20>/k x i' incorporates:
19       * Inport: '<Root>/vECEF'
20       * Inport: '<Root>/xECEF'
21       */
22      Vehicle_B.kxi = Vehicle_U.vECEF[2] * Vehicle_U.xECEF[0];
23
24      /* Product: '<S20>/i x j' incorporates:
25       * Inport: '<Root>/vECEF'
26       * Inport: '<Root>/xECEF'
27       */
28      Vehicle_B.ixj = Vehicle_U.vECEF[0] * Vehicle_U.xECEF[1];
29
30      /* Product: '<S21>/k x j' incorporates:
31       * Inport: '<Root>/vECEF'
32       * Inport: '<Root>/xECEF'
33       */
34      Vehicle_B.kxj = Vehicle_U.vECEF[2] * Vehicle_U.xECEF[1];
35
36      /* Product: '<S21>/i x k' incorporates:
37       * Inport: '<Root>/vECEF'
38       * Inport: '<Root>/xECEF'
39       */
40      Vehicle_B.ixk = Vehicle_U.vECEF[0] * Vehicle_U.xECEF[2];
41
42      /* Product: '<S21>/j x i' incorporates:
43       * Inport: '<Root>/vECEF'
44       * Inport: '<Root>/xECEF'
45       */
46      Vehicle_B.jxi = Vehicle_U.vECEF[1] * Vehicle_U.xECEF[0];
47
48      /* Sum: '<S13>/Sum' */
49      Vehicle_B.Sum[0] = Vehicle_B.jxk - Vehicle_B.kxj;
50      Vehicle_B.Sum[1] = Vehicle_B.kxi - Vehicle_B.ixk;
51      Vehicle_B.Sum[2] = Vehicle_B.ixj - Vehicle_B.jxi;
52
53      /* Gain: '<S7>/z = -r' incorporates:
54       * Inport: '<Root>/xECEF'
55       */
56      Vehicle_B.zr[0] = -Vehicle_U.xECEF[0];
57      Vehicle_B.zr[1] = -Vehicle_U.xECEF[1];
58      Vehicle_B.zr[2] = -Vehicle_U.xECEF[2];
59
60      /* Product: '<S18>/i x j' */
61      Vehicle_B.ixj_f = Vehicle_B.Sum[0] * Vehicle_B.zr[1];
62
63      /* Product: '<S18>/j x k' */
64      Vehicle_B.jxk_o = Vehicle_B.Sum[1] * Vehicle_B.zr[2];
65
66      /* Product: '<S18>/k x i' */
67      Vehicle_B.kxi_e = Vehicle_B.Sum[2] * Vehicle_B.zr[0];
68
69      /* Product: '<S19>/i x k' */
70      Vehicle_B.ixk_g = Vehicle_B.Sum[0] * Vehicle_B.zr[2];

```

```

71
72 /* Product: '<S19>/j x i' */
73 Vehicle_B.jxi_1 = Vehicle_B.Sum[1] * Vehicle_B.zr[0];
74
75 /* Product: '<S19>/k x j' */
76 Vehicle_B.kxj_a = Vehicle_B.Sum[2] * Vehicle_B.zr[1];
77
78 /* Sum: '<S12>/Sum' */
79 Vehicle_B.Sum_1[0] = Vehicle_B.jxk_o - Vehicle_B.kxj_a;
80 Vehicle_B.Sum_1[1] = Vehicle_B.kxi_e - Vehicle_B.ixk_g;
81 Vehicle_B.Sum_1[2] = Vehicle_B.ixj_f - Vehicle_B.jxi_1;
82
83 /* DotProduct: '<S15>/Dot Product' */
84 tmpForInput_idx_1 = Vehicle_B.Sum_1[0];
85 tmpForInput_idx_0 = Vehicle_B.Sum_1[0];
86 tmpForInput_idx_2 = tmpForInput_idx_1 * tmpForInput_idx_0;
87 tmpForInput_idx_1 = Vehicle_B.Sum_1[1];
88 tmpForInput_idx_0 = Vehicle_B.Sum_1[1];
89 tmpForInput_idx_2 += tmpForInput_idx_1 * tmpForInput_idx_0;
90 tmpForInput_idx_1 = Vehicle_B.Sum_1[2];
91 tmpForInput_idx_0 = Vehicle_B.Sum_1[2];
92 tmpForInput_idx_2 += tmpForInput_idx_1 * tmpForInput_idx_0;
93
94 /* DotProduct: '<S15>/Dot Product' */
95 Vehicle_B.DotProduct = tmpForInput_idx_2;
96
97 /* Sum: '<S15>/Sum of Elements' */
98 tmpForInput_idx_0 = Vehicle_B.DotProduct;
99
100 /* Sum: '<S15>/Sum of Elements' */
101 Vehicle_B.SumofElements = tmpForInput_idx_0;
102
103 /* Math: '<S15>/Math Function1'
104  *
105  * About '<S15>/Math Function1':
106  * Operator: sqrt
107  */
108 tmpForInput_idx_0 = Vehicle_B.SumofElements;
109 if (tmpForInput_idx_0 < 0.0) {
110     /* Math: '<S15>/Math Function1'
111     *
112     * About '<S15>/Math Function1':
113     * Operator: sqrt
114     */
115     Vehicle_B.MathFunction1 = -std::sqrt(std::abs(tmpForInput_idx_0));
116 } else {
117     /* Math: '<S15>/Math Function1'
118     *
119     * About '<S15>/Math Function1':
120     * Operator: sqrt
121     */
122     Vehicle_B.MathFunction1 = std::sqrt(tmpForInput_idx_0);
123 }
124
125 /* End of Math: '<S15>/Math Function1' */
126
127 /* Product: '<S15>/Divide' */
128 Vehicle_B.ORFtoECEf[0] = Vehicle_B.Sum_1[0] / Vehicle_B.MathFunction1;
129
130 /* DotProduct: '<S16>/Dot Product' */
131 tmpForInput_idx_1 = Vehicle_B.Sum[0];
132 tmpForInput_idx_0 = Vehicle_B.Sum[0];
133 tmpForInput_idx_2 = tmpForInput_idx_1 * tmpForInput_idx_0;
134
135 /* Product: '<S15>/Divide' */
136 Vehicle_B.ORFtoECEf[1] = Vehicle_B.Sum_1[1] / Vehicle_B.MathFunction1;
137
138 /* DotProduct: '<S16>/Dot Product' */
139 tmpForInput_idx_1 = Vehicle_B.Sum[1];
140 tmpForInput_idx_0 = Vehicle_B.Sum[1];
141 tmpForInput_idx_2 += tmpForInput_idx_1 * tmpForInput_idx_0;
142
143 /* Product: '<S15>/Divide' */
144 Vehicle_B.ORFtoECEf[2] = Vehicle_B.Sum_1[2] / Vehicle_B.MathFunction1;
145
146 /* DotProduct: '<S16>/Dot Product' */
147 tmpForInput_idx_1 = Vehicle_B.Sum[2];
148 tmpForInput_idx_0 = Vehicle_B.Sum[2];
149 tmpForInput_idx_2 += tmpForInput_idx_1 * tmpForInput_idx_0;
150
151 /* DotProduct: '<S16>/Dot Product' */
152 Vehicle_B.DotProduct_p = tmpForInput_idx_2;
153
154 /* Sum: '<S16>/Sum of Elements' */
155 tmpForInput_idx_0 = Vehicle_B.DotProduct_p;
156
157 /* Sum: '<S16>/Sum of Elements' */
158 Vehicle_B.SumofElements_c = tmpForInput_idx_0;
159
160 /* Math: '<S16>/Math Function1'
161  *
162  * About '<S16>/Math Function1':
163  * Operator: sqrt
164  */
165 tmpForInput_idx_0 = Vehicle_B.SumofElements_c;
166 if (tmpForInput_idx_0 < 0.0) {
167     /* Math: '<S16>/Math Function1'
168     *
169     * About '<S16>/Math Function1':
170     * Operator: sqrt
171     */
172     Vehicle_B.MathFunction1_b = -std::sqrt(std::abs(tmpForInput_idx_0));
173 } else {
174     /* Math: '<S16>/Math Function1'

```

```

175     *
176     * About '<S16>/Math Function1':
177     * Operator: sqrt
178     */
179     Vehicle_B.MathFunction1_b = std::sqrt(tmpForInput_idx_0);
180 }
181
182 /* End of Math: '<S16>/Math Function1' */
183
184 /* Product: '<S16>/Divide' */
185 Vehicle_B.ORFtoECEf[3] = Vehicle_B.Sum[0] / Vehicle_B.MathFunction1_b;
186
187 /* DotProduct: '<S17>/Dot Product' */
188 tmpForInput_idx_1 = Vehicle_B.zr[0];
189 tmpForInput_idx_0 = Vehicle_B.zr[0];
190 tmpForInput_idx_2 = tmpForInput_idx_1 * tmpForInput_idx_0;
191
192 /* Product: '<S16>/Divide' */
193 Vehicle_B.ORFtoECEf[4] = Vehicle_B.Sum[1] / Vehicle_B.MathFunction1_b;
194
195 /* DotProduct: '<S17>/Dot Product' */
196 tmpForInput_idx_1 = Vehicle_B.zr[1];
197 tmpForInput_idx_0 = Vehicle_B.zr[1];
198 tmpForInput_idx_2 += tmpForInput_idx_1 * tmpForInput_idx_0;
199
200 /* Product: '<S16>/Divide' */
201 Vehicle_B.ORFtoECEf[5] = Vehicle_B.Sum[2] / Vehicle_B.MathFunction1_b;
202
203 /* DotProduct: '<S17>/Dot Product' */
204 tmpForInput_idx_1 = Vehicle_B.zr[2];
205 tmpForInput_idx_0 = Vehicle_B.zr[2];
206 tmpForInput_idx_2 += tmpForInput_idx_1 * tmpForInput_idx_0;
207
208 /* DotProduct: '<S17>/Dot Product' */
209 Vehicle_B.DotProduct_b = tmpForInput_idx_2;
210
211 /* Sum: '<S17>/Sum of Elements' */
212 tmpForInput_idx_0 = Vehicle_B.DotProduct_b;
213
214 /* Sum: '<S17>/Sum of Elements' */
215 Vehicle_B.SumofElements_l = tmpForInput_idx_0;
216
217 /* Math: '<S17>/Math Function1'
218  *
219  * About '<S17>/Math Function1':
220  * Operator: sqrt
221  */
222 tmpForInput_idx_0 = Vehicle_B.SumofElements_l;
223 if (tmpForInput_idx_0 < 0.0) {
224     /* Math: '<S17>/Math Function1'
225     *
226     * About '<S17>/Math Function1':
227     * Operator: sqrt
228     */
229     Vehicle_B.MathFunction1_m = -std::sqrt(std::abs(tmpForInput_idx_0));
230 } else {
231     /* Math: '<S17>/Math Function1'
232     *
233     * About '<S17>/Math Function1':
234     * Operator: sqrt
235     */
236     Vehicle_B.MathFunction1_m = std::sqrt(tmpForInput_idx_0);
237 }
238
239 /* End of Math: '<S17>/Math Function1' */
240
241 /* Product: '<S17>/Divide' */
242 Vehicle_B.ORFtoECEf[6] = Vehicle_B.zr[0] / Vehicle_B.MathFunction1_m;
243 Vehicle_B.ORFtoECEf[7] = Vehicle_B.zr[1] / Vehicle_B.MathFunction1_m;
244 Vehicle_B.ORFtoECEf[8] = Vehicle_B.zr[2] / Vehicle_B.MathFunction1_m;
245 for (i = 0; i < 3; i++) {
246     /* Math: '<S7>/ECEf to ORF' incorporates:
247     * Concatenate: '<S7>/ORF to ECEf'
248     */
249     Vehicle_B.ECEfToORF[3 * i] = Vehicle_B.ORFtoECEf[i];
250     Vehicle_B.ECEfToORF[3 * i + 1] = Vehicle_B.ORFtoECEf[i + 3];
251     Vehicle_B.ECEfToORF[3 * i + 2] = Vehicle_B.ORFtoECEf[i + 6];
252 }
253
254 /* Sum: '<S25>/Add' */
255 tmpForInput_idx_0 = Vehicle_B.ECEfToORF[0];
256 tmpForInput_idx_1 = Vehicle_B.ECEfToORF[4];
257 tmpForInput_idx_2 = Vehicle_B.ECEfToORF[8];
258 tmpForInput_idx_0 += tmpForInput_idx_1;
259 tmpForInput_idx_0 += tmpForInput_idx_2;
260
261 /* Sum: '<S25>/Add' */
262 Vehicle_B.Add = tmpForInput_idx_0;
263
264 /* If: '<S14>/If' incorporates:
265  * If: '<S22>/Find Maximum Diagonal Value'
266  */
267 if (Vehicle_B.Add > 0.0) {
268     /* Outputs for IfAction SubSystem: '<S14>/Positive Trace' incorporates:
269     * ActionPort: '<S23>/Action Port'
270     */
271     /* Sum: '<S23>/Sum' incorporates:
272     * Constant: '<S23>/Constant'
273     */
274     Vehicle_B.Sum_gh = Vehicle_B.Add + 1.0;
275
276     /* Sqrt: '<S23>/sqrt' */
277     Vehicle_B.sqrt_a = std::sqrt(Vehicle_B.Sum_gh);
278

```

```

279 /* Gain: '<S23>/Gain' */
280 Vehicle_B.Merge_i[0] = 0.5 * Vehicle_B.sqrt_a;
281
282 /* Gain: '<S23>/Gain1' */
283 Vehicle_B.Gain1 = 2.0 * Vehicle_B.sqrt_a;
284
285 /* Sum: '<S46>/Add' */
286 Vehicle_B.Add_k = Vehicle_B.ECEFtoORF[7] - Vehicle_B.ECEFtoORF[5];
287
288 /* Sum: '<S45>/Add' */
289 Vehicle_B.Add_el = Vehicle_B.ECEFtoORF[2] - Vehicle_B.ECEFtoORF[6];
290
291 /* Sum: '<S47>/Add' */
292 Vehicle_B.Add_a = Vehicle_B.ECEFtoORF[3] - Vehicle_B.ECEFtoORF[1];
293
294 /* Product: '<S23>/Product' */
295 Vehicle_B.Merge_i[1] = Vehicle_B.Add_k / Vehicle_B.Gain1;
296 Vehicle_B.Merge_i[2] = Vehicle_B.Add_el / Vehicle_B.Gain1;
297 Vehicle_B.Merge_i[3] = Vehicle_B.Add_a / Vehicle_B.Gain1;
298
299 /* End of Outputs for SubSystem: '<S14>/Positive Trace' */
300 } else {
301 /* Outputs for IfAction SubSystem: '<S14>/Negative Trace' incorporates:
302 * ActionPort: '<S22>/Action Port'
303 */
304 if ((Vehicle_B.ECEFtoORF[4] > Vehicle_B.ECEFtoORF[0]) &&
305     (Vehicle_B.ECEFtoORF[4] > Vehicle_B.ECEFtoORF[8])) {
306 /* Outputs for IfAction SubSystem: '<S22>/Maximum Value at DCM(2,2)' incorporates:
307 * ActionPort: '<S27>/Action Port'
308 */
309 /* If: '<S22>/Find Maximum Diagonal Value' incorporates:
310 * Constant: '<S39>/Constant'
311 * Gain: '<S27>/Gain'
312 * Gain: '<S27>/Gain1'
313 * Gain: '<S27>/Gain3'
314 * Gain: '<S27>/Gain4'
315 * Product: '<S27>/Product'
316 * Product: '<S38>/Product'
317 * Sqrt: '<S27>/sqrt'
318 * Sum: '<S35>/Add'
319 * Sum: '<S36>/Add'
320 * Sum: '<S37>/Add'
321 * Sum: '<S39>/Add'
322 * Switch: '<S38>/Switch'
323 */
324 Vehicle_B.Add_g = ((Vehicle_B.ECEFtoORF[4] - Vehicle_B.ECEFtoORF[0]) -
325                    Vehicle_B.ECEFtoORF[8]) + 1.0;
326 Vehicle_B.sqrt_mw = std::sqrt(Vehicle_B.Add_g);
327 Vehicle_B.Merge_i[2] = 0.5 * Vehicle_B.sqrt_mw;
328 Vehicle_B.Add_gw = Vehicle_B.ECEFtoORF[1] + Vehicle_B.ECEFtoORF[3];
329 Vehicle_B.Add_fc = Vehicle_B.ECEFtoORF[5] + Vehicle_B.ECEFtoORF[7];
330 Vehicle_B.Add_d = Vehicle_B.ECEFtoORF[2] - Vehicle_B.ECEFtoORF[6];
331 if (Vehicle_B.sqrt_mw != 0.0) {
332 /* Switch: '<S38>/Switch' incorporates:
333 * Constant: '<S38>/Constant1'
334 */
335 Vehicle_B.Switch_f[0] = 0.5;
336 Vehicle_B.Switch_f[1] = Vehicle_B.sqrt_mw;
337 } else {
338 /* Switch: '<S38>/Switch' */
339 Vehicle_B.Switch_f[0] = 0.0;
340 Vehicle_B.Switch_f[1] = 1.0;
341 }
342
343 Vehicle_B.Product_e = Vehicle_B.Switch_f[0] / Vehicle_B.Switch_f[1];
344 Vehicle_B.Product_o[0] = Vehicle_B.Add_gw * Vehicle_B.Product_e;
345 Vehicle_B.Product_o[1] = Vehicle_B.Add_fc * Vehicle_B.Product_e;
346 Vehicle_B.Product_o[2] = Vehicle_B.Add_d * Vehicle_B.Product_e;
347 Vehicle_B.Merge_i[1] = Vehicle_B.Product_o[0];
348 Vehicle_B.Merge_i[3] = Vehicle_B.Product_o[1];
349 Vehicle_B.Merge_i[0] = Vehicle_B.Product_o[2];
350
351 /* End of Outputs for SubSystem: '<S22>/Maximum Value at DCM(2,2)' */
352 } else if (Vehicle_B.ECEFtoORF[8] > Vehicle_B.ECEFtoORF[0]) {
353 /* Outputs for IfAction SubSystem: '<S22>/Maximum Value at DCM(3,3)' incorporates:
354 * ActionPort: '<S28>/Action Port'
355 */
356 /* If: '<S22>/Find Maximum Diagonal Value' incorporates:
357 * Constant: '<S44>/Constant'
358 * Gain: '<S28>/Gain'
359 * Gain: '<S28>/Gain1'
360 * Gain: '<S28>/Gain2'
361 * Gain: '<S28>/Gain3'
362 * Product: '<S28>/Product'
363 * Product: '<S43>/Product'
364 * Sqrt: '<S28>/sqrt'
365 * Sum: '<S40>/Add'
366 * Sum: '<S41>/Add'
367 * Sum: '<S42>/Add'
368 * Sum: '<S44>/Add'
369 * Switch: '<S43>/Switch'
370 */
371 Vehicle_B.Add_f = ((Vehicle_B.ECEFtoORF[8] - Vehicle_B.ECEFtoORF[0]) -
372                    Vehicle_B.ECEFtoORF[4]) + 1.0;
373 Vehicle_B.sqrt_l = std::sqrt(Vehicle_B.Add_f);
374 Vehicle_B.Merge_i[3] = 0.5 * Vehicle_B.sqrt_l;
375 Vehicle_B.Add_e = Vehicle_B.ECEFtoORF[2] + Vehicle_B.ECEFtoORF[6];
376 Vehicle_B.Add_l2 = Vehicle_B.ECEFtoORF[5] + Vehicle_B.ECEFtoORF[7];
377 Vehicle_B.Add_ek = Vehicle_B.ECEFtoORF[3] - Vehicle_B.ECEFtoORF[1];
378 if (Vehicle_B.sqrt_l != 0.0) {
379 /* Switch: '<S43>/Switch' incorporates:
380 * Constant: '<S43>/Constant1'
381 */
382 Vehicle_B.Switch_b[0] = 0.5;

```

```

383     Vehicle_B.Switch_b[1] = Vehicle_B.sqrt_1;
384 } else {
385     /* Switch: '<S43>/Switch' */
386     Vehicle_B.Switch_b[0] = 0.0;
387     Vehicle_B.Switch_b[1] = 1.0;
388 }
389
390 Vehicle_B.Product_d = Vehicle_B.Switch_b[0] / Vehicle_B.Switch_b[1];
391 Vehicle_B.Product_i[0] = Vehicle_B.Add_e * Vehicle_B.Product_d;
392 Vehicle_B.Product_i[1] = Vehicle_B.Add_l2 * Vehicle_B.Product_d;
393 Vehicle_B.Product_i[2] = Vehicle_B.Add_ek * Vehicle_B.Product_d;
394 Vehicle_B.Merge_i[1] = Vehicle_B.Product_i[0];
395 Vehicle_B.Merge_i[2] = Vehicle_B.Product_i[1];
396 Vehicle_B.Merge_i[0] = Vehicle_B.Product_i[2];
397
398 /* End of Outputs for SubSystem: '<S22>/Maximum Value at DCM(3,3)' */
399 } else {
400     /* Outputs for IfAction SubSystem: '<S22>/Maximum Value at DCM(1,1)' incorporates:
401      * ActionPort: '<S26>/Action Port'
402      */
403     /* If: '<S22>/Find Maximum Diagonal Value' incorporates:
404      * Constant: '<S34>/Constant'
405      * Gain: '<S26>/Gain'
406      * Gain: '<S26>/Gain1'
407      * Gain: '<S26>/Gain2'
408      * Gain: '<S26>/Gain3'
409      * Product: '<S26>/Product'
410      * Product: '<S33>/Product'
411      * Sqrt: '<S26>/sqrt'
412      * Sum: '<S30>/Add'
413      * Sum: '<S31>/Add'
414      * Sum: '<S32>/Add'
415      * Sum: '<S34>/Add'
416      * Switch: '<S33>/Switch'
417      */
418     Vehicle_B.Add_m = ((Vehicle_B.ECEFtoORF[0] - Vehicle_B.ECEFtoORF[4]) -
419                       Vehicle_B.ECEFtoORF[8]) + 1.0;
420     Vehicle_B.sqrt_m = std::sqrt(Vehicle_B.Add_m);
421     Vehicle_B.Merge_i[1] = 0.5 * Vehicle_B.sqrt_m;
422     if (Vehicle_B.sqrt_m != 0.0) {
423         /* Switch: '<S33>/Switch' incorporates:
424          * Constant: '<S33>/Constant1'
425          */
426         Vehicle_B.Switch[0] = 0.5;
427         Vehicle_B.Switch[1] = Vehicle_B.sqrt_m;
428     } else {
429         /* Switch: '<S33>/Switch' */
430         Vehicle_B.Switch[0] = 0.0;
431         Vehicle_B.Switch[1] = 1.0;
432     }
433
434     Vehicle_B.Product_k = Vehicle_B.Switch[0] / Vehicle_B.Switch[1];
435     Vehicle_B.Add_o = Vehicle_B.ECEFtoORF[1] + Vehicle_B.ECEFtoORF[3];
436     Vehicle_B.Add_ov = Vehicle_B.ECEFtoORF[2] + Vehicle_B.ECEFtoORF[6];
437     Vehicle_B.Add_l = Vehicle_B.ECEFtoORF[7] - Vehicle_B.ECEFtoORF[5];
438     Vehicle_B.Product_cy[0] = Vehicle_B.Product_k * Vehicle_B.Add_o;
439     Vehicle_B.Product_cy[1] = Vehicle_B.Product_k * Vehicle_B.Add_ov;
440     Vehicle_B.Product_cy[2] = Vehicle_B.Product_k * Vehicle_B.Add_l;
441     Vehicle_B.Merge_i[2] = Vehicle_B.Product_cy[0];
442     Vehicle_B.Merge_i[3] = Vehicle_B.Product_cy[1];
443     Vehicle_B.Merge_i[0] = Vehicle_B.Product_cy[2];
444
445     /* End of Outputs for SubSystem: '<S22>/Maximum Value at DCM(1,1)' */
446 }
447
448 /* End of Outputs for SubSystem: '<S14>/Negative Trace' */
449 }
450
451 /* End of If: '<S14>/If' */
452
453 /* Product: '<S111>/Product' incorporates:
454  * Inport: '<Root>/qECEF2b'
455  */
456 Vehicle_B.Product = Vehicle_U.qECEF2b[0] * Vehicle_U.qECEF2b[0];
457
458 /* Product: '<S111>/Product1' incorporates:
459  * Inport: '<Root>/qECEF2b'
460  */
461 Vehicle_B.Product1 = Vehicle_U.qECEF2b[1] * Vehicle_U.qECEF2b[1];
462
463 /* Product: '<S111>/Product2' incorporates:
464  * Inport: '<Root>/qECEF2b'
465  */
466 Vehicle_B.Product2 = Vehicle_U.qECEF2b[2] * Vehicle_U.qECEF2b[2];
467
468 /* Product: '<S111>/Product3' incorporates:
469  * Inport: '<Root>/qECEF2b'
470  */
471 Vehicle_B.Product3 = Vehicle_U.qECEF2b[3] * Vehicle_U.qECEF2b[3];
472
473 /* Sum: '<S111>/Sum' */
474 Vehicle_B.Sum_i = ((Vehicle_B.Product + Vehicle_B.Product1) +
475                   Vehicle_B.Product2) + Vehicle_B.Product3;
476
477 /* Product: '<S9>/Divide' incorporates:
478  * Inport: '<Root>/qECEF2b'
479  */
480 Vehicle_B.Divide = Vehicle_U.qECEF2b[0] / Vehicle_B.Sum_i;
481
482 /* Product: '<S112>/Product' */
483 Vehicle_B.Product_j = Vehicle_B.Divide * Vehicle_B.Merge_i[0];
484
485 /* UnaryMinus: '<S110>/Unary Minus' incorporates:
486  * Inport: '<Root>/qECEF2b'

```

```

487  */
488  Vehicle_B.UnaryMinus = -Vehicle_U.qECEF2b[1];
489
490  /* Product: '<S9>/Divide1' */
491  Vehicle_B.Divide1 = Vehicle_B.UnaryMinus / Vehicle_B.Sum_i;
492
493  /* Product: '<S112>/Product1' */
494  Vehicle_B.Product1_h = Vehicle_B.Divide1 * Vehicle_B.Merge_i[1];
495
496  /* UnaryMinus: '<S110>/Unary Minus1' incorporates:
497   * Inport: '<Root>/qECEF2b'
498   */
499  Vehicle_B.UnaryMinus1 = -Vehicle_U.qECEF2b[2];
500
501  /* Product: '<S9>/Divide2' */
502  Vehicle_B.Divide2 = Vehicle_B.UnaryMinus1 / Vehicle_B.Sum_i;
503
504  /* Product: '<S112>/Product2' */
505  Vehicle_B.Product2_d = Vehicle_B.Divide2 * Vehicle_B.Merge_i[2];
506
507  /* UnaryMinus: '<S110>/Unary Minus2' incorporates:
508   * Inport: '<Root>/qECEF2b'
509   */
510  Vehicle_B.UnaryMinus2 = -Vehicle_U.qECEF2b[3];
511
512  /* Product: '<S9>/Divide3' */
513  Vehicle_B.Divide3 = Vehicle_B.UnaryMinus2 / Vehicle_B.Sum_i;
514
515  /* Product: '<S112>/Product3' */
516  Vehicle_B.Product3_c = Vehicle_B.Divide3 * Vehicle_B.Merge_i[3];
517
518  /* Sum: '<S112>/Sum' */
519  Vehicle_B.Sum_p = ((Vehicle_B.Product_j - Vehicle_B.Product1_h) -
520                    Vehicle_B.Product2_d) - Vehicle_B.Product3_c;
521
522  /* Product: '<S123>/Product' */
523  Vehicle_B.Product_b = Vehicle_B.Sum_p * Vehicle_B.Sum_p;
524
525  /* Product: '<S113>/Product' */
526  Vehicle_B.Product_p = Vehicle_B.Divide * Vehicle_B.Merge_i[1];
527
528  /* Product: '<S113>/Product1' */
529  Vehicle_B.Product1_i = Vehicle_B.Divide1 * Vehicle_B.Merge_i[0];
530
531  /* Product: '<S113>/Product2' */
532  Vehicle_B.Product2_o = Vehicle_B.Divide2 * Vehicle_B.Merge_i[3];
533
534  /* Product: '<S113>/Product3' */
535  Vehicle_B.Product3_p = Vehicle_B.Divide3 * Vehicle_B.Merge_i[2];
536
537  /* Sum: '<S113>/Sum' */
538  Vehicle_B.Sum_iw = ((Vehicle_B.Product_p + Vehicle_B.Product1_i) +
539                    Vehicle_B.Product2_o) - Vehicle_B.Product3_p;
540
541  /* Product: '<S123>/Product1' */
542  Vehicle_B.Product1_in = Vehicle_B.Sum_iw * Vehicle_B.Sum_iw;
543
544  /* Product: '<S114>/Product' */
545  Vehicle_B.Product_by = Vehicle_B.Divide * Vehicle_B.Merge_i[2];
546
547  /* Product: '<S114>/Product1' */
548  Vehicle_B.Product1_f = Vehicle_B.Divide1 * Vehicle_B.Merge_i[3];
549
550  /* Product: '<S114>/Product2' */
551  Vehicle_B.Product2_m = Vehicle_B.Divide2 * Vehicle_B.Merge_i[0];
552
553  /* Product: '<S114>/Product3' */
554  Vehicle_B.Product3_pa = Vehicle_B.Divide3 * Vehicle_B.Merge_i[1];
555
556  /* Sum: '<S114>/Sum' */
557  Vehicle_B.Sum_d = ((Vehicle_B.Product_by - Vehicle_B.Product1_f) +
558                    Vehicle_B.Product2_m) + Vehicle_B.Product3_pa;
559
560  /* Product: '<S123>/Product2' */
561  Vehicle_B.Product2_n = Vehicle_B.Sum_d * Vehicle_B.Sum_d;
562
563  /* Product: '<S115>/Product' */
564  Vehicle_B.Product_pd = Vehicle_B.Divide * Vehicle_B.Merge_i[3];
565
566  /* Product: '<S115>/Product1' */
567  Vehicle_B.Product1_d = Vehicle_B.Divide1 * Vehicle_B.Merge_i[2];
568
569  /* Product: '<S115>/Product2' */
570  Vehicle_B.Product2_nn = Vehicle_B.Divide2 * Vehicle_B.Merge_i[1];
571
572  /* Product: '<S115>/Product3' */
573  Vehicle_B.Product3_l = Vehicle_B.Divide3 * Vehicle_B.Merge_i[0];
574
575  /* Sum: '<S115>/Sum' */
576  Vehicle_B.Sum_o = ((Vehicle_B.Product_pd + Vehicle_B.Product1_d) -
577                    Vehicle_B.Product2_nn) + Vehicle_B.Product3_l;
578
579  /* Product: '<S123>/Product3' */
580  Vehicle_B.Product3_ph = Vehicle_B.Sum_o * Vehicle_B.Sum_o;
581
582  /* Sum: '<S123>/Sum' */
583  Vehicle_B.Sum_g = ((Vehicle_B.Product_b + Vehicle_B.Product1_in) +
584                    Vehicle_B.Product2_n) + Vehicle_B.Product3_ph;
585
586  /* Sqrt: '<S122>/sqrt' */
587  Vehicle_B.sqrt_g = std::sqrt(Vehicle_B.Sum_g);
588
589  /* Product: '<S117>/Product' */
590  Vehicle_B.Product_h = Vehicle_B.Sum_p / Vehicle_B.sqrt_g;

```

```

591 /* Product: '<S117>/Product1' */
592 Vehicle_B.Product1_j = Vehicle_B.Sum_iw / Vehicle_B.sqrt_g;
593
594 /* Product: '<S117>/Product2' */
595 Vehicle_B.Product2_dj = Vehicle_B.Sum_d / Vehicle_B.sqrt_g;
596
597 /* Product: '<S117>/Product3' */
598 Vehicle_B.Product3_e = Vehicle_B.Sum_o / Vehicle_B.sqrt_g;
599
600
601 /* Fcn: '<S11>/fcn1' */
602 Vehicle_B.fcn1 = (Vehicle_B.Product2_dj * Vehicle_B.Product3_e -
603     Vehicle_B.Product_h * Vehicle_B.Product1_j) * -2.0;
604
605 /* Fcn: '<S11>/fcn2' */
606 Vehicle_B.fcn2 = ((Vehicle_B.Product_h * Vehicle_B.Product_h -
607     Vehicle_B.Product1_j * Vehicle_B.Product1_j) -
608     Vehicle_B.Product2_dj * Vehicle_B.Product2_dj) +
609     Vehicle_B.Product3_e * Vehicle_B.Product3_e;
610
611 /* Trigonometry: '<S116>/Trigonometric Function1' */
612 Vehicle_B.VectorConcatenate_a[0] = rt_atan2d_snf(Vehicle_B.fcn1 ,
613     Vehicle_B.fcn2);
614
615 /* Fcn: '<S11>/fcn3' */
616 Vehicle_B.fcn3 = (Vehicle_B.Product1_j * Vehicle_B.Product3_e +
617     Vehicle_B.Product_h * Vehicle_B.Product2_dj) * 2.0;
618
619 /* If: '<S118>/If' */
620 if (Vehicle_B.fcn3 > 1.0) {
621     /* Outputs for IfAction SubSystem: '<S118>/If Action Subsystem' incorporates:
622     * ActionPort: '<S119>/Action Port'
623     */
624     /* Merge: '<S118>/Merge' incorporates:
625     * Constant: '<S119>/Constant'
626     */
627     Vehicle_B.Merge_il = 1.0;
628
629     /* End of Outputs for SubSystem: '<S118>/If Action Subsystem' */
630 } else if (Vehicle_B.fcn3 < -1.0) {
631     /* Outputs for IfAction SubSystem: '<S118>/If Action Subsystem1' incorporates:
632     * ActionPort: '<S120>/Action Port'
633     */
634     /* Merge: '<S118>/Merge' incorporates:
635     * Constant: '<S120>/Constant'
636     */
637     Vehicle_B.Merge_il = 1.0;
638
639     /* End of Outputs for SubSystem: '<S118>/If Action Subsystem1' */
640 } else {
641     /* Outputs for IfAction SubSystem: '<S118>/If Action Subsystem2' incorporates:
642     * ActionPort: '<S121>/Action Port'
643     */
644     //Vehicle_IfActionSubsystem2_f(Vehicle_B.fcn3, &Vehicle_B.Merge_il);
645     Vehicle_B.Merge_il = Vehicle_B.fcn3;
646     /* End of Outputs for SubSystem: '<S118>/If Action Subsystem2' */
647 }
648
649 /* End of If: '<S118>/If' */
650
651 /* Trigonometry: '<S116>/trigFcn' */
652 tmpForInput_idx_0 = Vehicle_B.Merge_il;
653 if (tmpForInput_idx_0 > 1.0) {
654     tmpForInput_idx_0 = 1.0;
655 } else {
656     if (tmpForInput_idx_0 < -1.0) {
657         tmpForInput_idx_0 = -1.0;
658     }
659 }
660
661 Vehicle_B.VectorConcatenate_a[1] = std::asin(tmpForInput_idx_0);
662
663 /* End of Trigonometry: '<S116>/trigFcn' */
664
665 /* Fcn: '<S11>/fcn4' */
666 Vehicle_B.fcn4 = (Vehicle_B.Product1_j * Vehicle_B.Product2_dj -
667     Vehicle_B.Product_h * Vehicle_B.Product3_e) * -2.0;
668
669 /* Fcn: '<S11>/fcn5' */
670 Vehicle_B.fcn5 = ((Vehicle_B.Product_h * Vehicle_B.Product_h +
671     Vehicle_B.Product1_j * Vehicle_B.Product1_j) -
672     Vehicle_B.Product2_dj * Vehicle_B.Product2_dj) -
673     Vehicle_B.Product3_e * Vehicle_B.Product3_e;
674
675 /* Trigonometry: '<S116>/Trigonometric Function3' */
676 Vehicle_B.VectorConcatenate_a[2] = rt_atan2d_snf(Vehicle_B.fcn4 ,
677     Vehicle_B.fcn5);
678
679 /* Gain: '<S85>/Derivative Gain' */
680 Vehicle_B.DerivativeGain[0] = 0.02 * Vehicle_B.VectorConcatenate_a[0];
681
682 /* SampleTimeMath: '<S88>/Tsamp'
683 *
684 * About '<S88>/Tsamp':
685 * y = u * K where K = 1 / ( w * Ts )
686 * Multiplication by K = weightedTsampQuantized is being
687 * done implicitly by changing the scaling of the input signal.
688 * No work needs to be done here. Downstream blocks may need
689 * to do work to handle the scaling of the output; this happens
690 * automatically.
691 */
692 Vehicle_B.Tsamp[0] = Vehicle_B.DerivativeGain[0];
693
694 /* Delay: '<S86>/UD' */

```



```
695 Vehicle_B.UD[0] = Vehicle_DW.UD_DSTATE[0];
696
697 /* Sum: '<S86>/Diff' */
698 Vehicle_B.Diff[0] = Vehicle_B.Tsamp[0] - Vehicle_B.UD[0];
699
700 /* Gain: '<S90>/Integral Gain' */
701 Vehicle_B.IntegralGain[0] = 2.0E-10 * Vehicle_B.VectorConcatenate_a[0];
702
703 /* DiscreteIntegrator: '<S93>/Integrator' */
704 Vehicle_B.Integrator[0] = Vehicle_DW.Integrator_DSTATE[0];
705
706 /* Gain: '<S98>/Proportional Gain' */
707 Vehicle_B.ProportionalGain[0] = 10.0E-5 * Vehicle_B.VectorConcatenate_a[0];
708
709 /* Merge: '<S1>/Merge' incorporates:
710  * Sum: '<S102>/Sum'
711  */
712 Vehicle_B.Merge[0] = (Vehicle_B.ProportionalGain[0] + Vehicle_B.Integrator[0])
713 + Vehicle_B.Diff[0];
714
715 /* Update for Delay: '<S86>/UD' */
716 Vehicle_DW.UD_DSTATE[0] = Vehicle_B.Tsamp[0];
717
718 /* Update for DiscreteIntegrator: '<S93>/Integrator' */
719 Vehicle_DW.Integrator_DSTATE[0] += Vehicle_B.IntegralGain[0];
720
721 /* Gain: '<S85>/Derivative Gain' */
722 Vehicle_B.DerivativeGain[1] = 0.02 * Vehicle_B.VectorConcatenate_a[1];
723
724 /* SampleTimeMath: '<S88>/Tsamp'
725  *
726  * About '<S88>/Tsamp':
727  * y = u * K where K = 1 / ( w * Ts )
728  * Multiplication by K = weightedTsampQuantized is being
729  * done implicitly by changing the scaling of the input signal.
730  * No work needs to be done here. Downstream blocks may need
731  * to do work to handle the scaling of the output; this happens
732  * automatically.
733  */
734 Vehicle_B.Tsamp[1] = Vehicle_B.DerivativeGain[1];
735
736 /* Delay: '<S86>/UD' */
737 Vehicle_B.UD[1] = Vehicle_DW.UD_DSTATE[1];
738
739 /* Sum: '<S86>/Diff' */
740 Vehicle_B.Diff[1] = Vehicle_B.Tsamp[1] - Vehicle_B.UD[1];
741
742 /* Gain: '<S90>/Integral Gain' */
743 Vehicle_B.IntegralGain[1] = 2.0E-10 * Vehicle_B.VectorConcatenate_a[1];
744
745 /* DiscreteIntegrator: '<S93>/Integrator' */
746 Vehicle_B.Integrator[1] = Vehicle_DW.Integrator_DSTATE[1];
747
748 /* Gain: '<S98>/Proportional Gain' */
749 Vehicle_B.ProportionalGain[1] = 10.0E-5 * Vehicle_B.VectorConcatenate_a[1];
750
751 /* Merge: '<S1>/Merge' incorporates:
752  * Sum: '<S102>/Sum'
753  */
754 Vehicle_B.Merge[1] = (Vehicle_B.ProportionalGain[1] + Vehicle_B.Integrator[1])
755 + Vehicle_B.Diff[1];
756
757 /* Update for Delay: '<S86>/UD' */
758 Vehicle_DW.UD_DSTATE[1] = Vehicle_B.Tsamp[1];
759
760 /* Update for DiscreteIntegrator: '<S93>/Integrator' */
761 Vehicle_DW.Integrator_DSTATE[1] += Vehicle_B.IntegralGain[1];
762
763 /* Gain: '<S85>/Derivative Gain' */
764 Vehicle_B.DerivativeGain[2] = 0.02 * Vehicle_B.VectorConcatenate_a[2];
765
766 /* SampleTimeMath: '<S88>/Tsamp'
767  *
768  * About '<S88>/Tsamp':
769  * y = u * K where K = 1 / ( w * Ts )
770  * Multiplication by K = weightedTsampQuantized is being
771  * done implicitly by changing the scaling of the input signal.
772  * No work needs to be done here. Downstream blocks may need
773  * to do work to handle the scaling of the output; this happens
774  * automatically.
775  */
776 Vehicle_B.Tsamp[2] = Vehicle_B.DerivativeGain[2];
777
778 /* Delay: '<S86>/UD' */
779 Vehicle_B.UD[2] = Vehicle_DW.UD_DSTATE[2];
780
781 /* Sum: '<S86>/Diff' */
782 Vehicle_B.Diff[2] = Vehicle_B.Tsamp[2] - Vehicle_B.UD[2];
783
784 /* Gain: '<S90>/Integral Gain' */
785 Vehicle_B.IntegralGain[2] = 2.0E-10 * Vehicle_B.VectorConcatenate_a[2];
786
787 /* DiscreteIntegrator: '<S93>/Integrator' */
788 Vehicle_B.Integrator[2] = Vehicle_DW.Integrator_DSTATE[2];
789
790 /* Gain: '<S98>/Proportional Gain' */
791 Vehicle_B.ProportionalGain[2] = 10.0E-5 * Vehicle_B.VectorConcatenate_a[2];
792
793 /* Merge: '<S1>/Merge' incorporates:
794  * Sum: '<S102>/Sum'
795  */
796 Vehicle_B.Merge[2] = (Vehicle_B.ProportionalGain[2] + Vehicle_B.Integrator[2])
797 + Vehicle_B.Diff[2];
798
```



```

799  /* Update for Delay: '<S86>/UD' */
800  Vehicle_DW.UD_DSTATE[2] = Vehicle_B.Tsamp[2];
801
802  /* Update for DiscreteIntegrator: '<S93>/Integrator' */
803  Vehicle_DW.Integrator_DSTATE[2] += Vehicle_B.IntegralGain[2];
804
805  /* End of Outputs for SubSystem: '<S1>/NadirPointing Mode' */
806 }

```

SunPointing.cpp

```

1  #include "Vehicle.h"
2
3  void SunPointing()
4  {
5      real_T tmpForInput;
6      real_T tmpForInput_0;
7      real_T tmpForInput_1;
8      real_T tmpForInput_idx_0;
9      real_T tmpForInput_idx_1;
10     real_T tmpForInput_idx_2;
11     /* Outputs for IfAction SubSystem: '<S1>/SunPointing Mode' incorporates:
12      *   ActionPort: '<S4>/Action Port'
13      */
14     /* If: '<S129>/If' */
15     if (Vehicle_ConstB.Abs < 1.0E-6) {
16         /* Outputs for IfAction SubSystem: '<S129>/If Action Subsystem' incorporates:
17          *   ActionPort: '<S140>/Action Port'
18          */
19         // Vehicle_IfActionSubsystem(&Vehicle_B.Merge_m);
20         Vehicle_B.Merge_m = 1.0;
21         /* End of Outputs for SubSystem: '<S129>/If Action Subsystem' */
22     } else if (Vehicle_ConstB.Abs1 < 1.0E-6) {
23         /* Outputs for IfAction SubSystem: '<S129>/If Action Subsystem1' incorporates:
24          *   ActionPort: '<S141>/Action Port'
25          */
26         // Vehicle_IfActionSubsystem1(&Vehicle_B.Merge_m);
27         Vehicle_B.Merge_m = -1.0;
28         /* End of Outputs for SubSystem: '<S129>/If Action Subsystem1' */
29     } else {
30         /* Outputs for IfAction SubSystem: '<S129>/If Action Subsystem2' incorporates:
31          *   ActionPort: '<S142>/Action Port'
32          */
33         // Vehicle_IfActionSubsystem2(&Vehicle_B.Merge_m);
34         Vehicle_B.Merge_m = 0.0;
35         /* End of Outputs for SubSystem: '<S129>/If Action Subsystem2' */
36     }
37
38     /* End of If: '<S129>/If' */
39
40     /* Product: '<S259>/Product' incorporates:
41      *   Inport: '<Root>/q_eci2b'
42      */
43     Vehicle_B.Product_n = Vehicle_U.q_eci2b[0] * Vehicle_U.q_eci2b[0];
44
45     /* Product: '<S259>/Product1' incorporates:
46      *   Inport: '<Root>/q_eci2b'
47      */
48     Vehicle_B.Product1_ii = Vehicle_U.q_eci2b[1] * Vehicle_U.q_eci2b[1];
49
50     /* Product: '<S259>/Product2' incorporates:
51      *   Inport: '<Root>/q_eci2b'
52      */
53     Vehicle_B.Product2_c = Vehicle_U.q_eci2b[2] * Vehicle_U.q_eci2b[2];
54
55     /* Product: '<S259>/Product3' incorporates:
56      *   Inport: '<Root>/q_eci2b'
57      */
58     Vehicle_B.Product3_d = Vehicle_U.q_eci2b[3] * Vehicle_U.q_eci2b[3];
59
60     /* Sum: '<S259>/Sum' */
61     Vehicle_B.Sum_j = ((Vehicle_B.Product_n + Vehicle_B.Product1_ii) +
62                       Vehicle_B.Product2_c) + Vehicle_B.Product3_d;
63
64     /* Sqrt: '<S258>/sqrt' */
65     Vehicle_B.sqrt_n = std::sqrt(Vehicle_B.Sum_j);
66
67     /* Product: '<S254>/Product2' incorporates:
68      *   Inport: '<Root>/q_eci2b'
69      */
70     Vehicle_B.Product2_oi = Vehicle_U.q_eci2b[2] / Vehicle_B.sqrt_n;
71
72     /* Product: '<S255>/Product6' */
73     Vehicle_B.Product6 = Vehicle_B.Product2_oi * Vehicle_B.Product2_oi;
74
75     /* Product: '<S254>/Product3' incorporates:
76      *   Inport: '<Root>/q_eci2b'
77      */
78     Vehicle_B.Product3_a = Vehicle_U.q_eci2b[3] / Vehicle_B.sqrt_n;
79
80     /* Product: '<S255>/Product7' */
81     Vehicle_B.Product7 = Vehicle_B.Product3_a * Vehicle_B.Product3_a;
82
83     /* Sum: '<S255>/Sum3' incorporates:
84      *   Constant: '<S255>/Constant'
85      */
86     Vehicle_B.Sum3 = (0.5 - Vehicle_B.Product6) - Vehicle_B.Product7;
87
88     /* Gain: '<S255>/Gain2' */
89     Vehicle_B.Gain2 = 2.0 * Vehicle_B.Sum3;
90
91     /* Product: '<S255>/Product8' incorporates:

```

```

92     *   Inport: '<Root>/xSunECI'
93   */
94   Vehicle_B.Product8 = Vehicle_U.xSunECI[0] * Vehicle_B.Gain2;
95
96   /* Product: '<S254>/Product1' incorporates:
97     *   Inport: '<Root>/q_eci2b'
98   */
99   Vehicle_B.Product1_fv = Vehicle_U.q_eci2b[1] / Vehicle_B.sqrt_n;
100
101   /* Product: '<S255>/Product' */
102   Vehicle_B.Product_jb = Vehicle_B.Product1_fv * Vehicle_B.Product2_oi;
103
104   /* Product: '<S254>/Product' incorporates:
105     *   Inport: '<Root>/q_eci2b'
106   */
107   Vehicle_B.Product_cw = Vehicle_U.q_eci2b[0] / Vehicle_B.sqrt_n;
108
109   /* Product: '<S255>/Product1' */
110   Vehicle_B.Product1_b = Vehicle_B.Product_cw * Vehicle_B.Product3_a;
111
112   /* Sum: '<S255>/Sum1' */
113   Vehicle_B.Sum1 = Vehicle_B.Product_jb + Vehicle_B.Product1_b;
114
115   /* Gain: '<S255>/Gain' */
116   Vehicle_B.Gain = 2.0 * Vehicle_B.Sum1;
117
118   /* Product: '<S255>/Product4' incorporates:
119     *   Inport: '<Root>/xSunECI'
120   */
121   Vehicle_B.Product4_p = Vehicle_B.Gain * Vehicle_U.xSunECI[1];
122
123   /* Product: '<S255>/Product2' */
124   Vehicle_B.Product2_ox = Vehicle_B.Product_cw * Vehicle_B.Product2_oi;
125
126   /* Product: '<S255>/Product3' */
127   Vehicle_B.Product3_pv = Vehicle_B.Product1_fv * Vehicle_B.Product3_a;
128
129   /* Sum: '<S255>/Sum2' */
130   Vehicle_B.Sum2 = Vehicle_B.Product3_pv - Vehicle_B.Product2_ox;
131
132   /* Gain: '<S255>/Gain1' */
133   Vehicle_B.Gain1_l = 2.0 * Vehicle_B.Sum2;
134
135   /* Product: '<S255>/Product5' incorporates:
136     *   Inport: '<Root>/xSunECI'
137   */
138   Vehicle_B.Product5_g = Vehicle_B.Gain1_l * Vehicle_U.xSunECI[2];
139
140   /* Sum: '<S255>/Sum' */
141   Vehicle_B.Sum_k = (Vehicle_B.Product8 + Vehicle_B.Product4_p) +
142     Vehicle_B.Product5_g;
143
144   /* Product: '<S256>/Product' */
145   Vehicle_B.Product_if = Vehicle_B.Product1_fv * Vehicle_B.Product2_oi;
146
147   /* Product: '<S256>/Product1' */
148   Vehicle_B.Product1_e = Vehicle_B.Product_cw * Vehicle_B.Product3_a;
149
150   /* Sum: '<S256>/Sum1' */
151   Vehicle_B.Sum1_b = Vehicle_B.Product_if - Vehicle_B.Product1_e;
152
153   /* Gain: '<S256>/Gain' */
154   Vehicle_B.Gain_a = 2.0 * Vehicle_B.Sum1_b;
155
156   /* Product: '<S256>/Product4' incorporates:
157     *   Inport: '<Root>/xSunECI'
158   */
159   Vehicle_B.Product4_k = Vehicle_U.xSunECI[0] * Vehicle_B.Gain_a;
160
161   /* Product: '<S256>/Product6' */
162   Vehicle_B.Product6_d = Vehicle_B.Product1_fv * Vehicle_B.Product1_fv;
163
164   /* Product: '<S256>/Product7' */
165   Vehicle_B.Product7_e = Vehicle_B.Product3_a * Vehicle_B.Product3_a;
166
167   /* Sum: '<S256>/Sum3' incorporates:
168     *   Constant: '<S256>/Constant'
169   */
170   Vehicle_B.Sum3_p = (0.5 - Vehicle_B.Product6_d) - Vehicle_B.Product7_e;
171
172   /* Gain: '<S256>/Gain2' */
173   Vehicle_B.Gain2_a = 2.0 * Vehicle_B.Sum3_p;
174
175   /* Product: '<S256>/Product8' incorporates:
176     *   Inport: '<Root>/xSunECI'
177   */
178   Vehicle_B.Product8_c = Vehicle_B.Gain2_a * Vehicle_U.xSunECI[1];
179
180   /* Product: '<S256>/Product2' */
181   Vehicle_B.Product2_l = Vehicle_B.Product_cw * Vehicle_B.Product1_fv;
182
183   /* Product: '<S256>/Product3' */
184   Vehicle_B.Product3_c5 = Vehicle_B.Product2_oi * Vehicle_B.Product3_a;
185
186   /* Sum: '<S256>/Sum2' */
187   Vehicle_B.Sum2_k = Vehicle_B.Product2_l + Vehicle_B.Product3_c5;
188
189   /* Gain: '<S256>/Gain1' */
190   Vehicle_B.Gain1_d = 2.0 * Vehicle_B.Sum2_k;
191
192   /* Product: '<S256>/Product5' incorporates:
193     *   Inport: '<Root>/xSunECI'
194   */
195   Vehicle_B.Product5_j = Vehicle_B.Gain1_d * Vehicle_U.xSunECI[2];

```

```

196 /* Sum: '<S256>/Sum' */
197 Vehicle_B.Sum_lm = (Vehicle_B.Product4_k + Vehicle_B.Product8_c) +
198     Vehicle_B.Product5_j;
199
200
201 /* Product: '<S257>/Product' */
202 Vehicle_B.Product_c3 = Vehicle_B.Product1_fv * Vehicle_B.Product3_a;
203
204 /* Product: '<S257>/Product1' */
205 Vehicle_B.Product1_c = Vehicle_B.Product_cw * Vehicle_B.Product2_oi;
206
207 /* Sum: '<S257>/Sum1' */
208 Vehicle_B.Sum1_bu = Vehicle_B.Product_c3 + Vehicle_B.Product1_c;
209
210 /* Gain: '<S257>/Gain' */
211 Vehicle_B.Gain_k = 2.0 * Vehicle_B.Sum1_bu;
212
213 /* Product: '<S257>/Product4' incorporates:
214  * Inport: '<Root>/xSunECI'
215  */
216 Vehicle_B.Product4_m = Vehicle_U.xSunECI[0] * Vehicle_B.Gain_k;
217
218 /* Product: '<S257>/Product2' */
219 Vehicle_B.Product2_k = Vehicle_B.Product_cw * Vehicle_B.Product1_fv;
220
221 /* Product: '<S257>/Product3' */
222 Vehicle_B.Product3_f = Vehicle_B.Product2_oi * Vehicle_B.Product3_a;
223
224 /* Sum: '<S257>/Sum2' */
225 Vehicle_B.Sum2_p = Vehicle_B.Product3_f - Vehicle_B.Product2_k;
226
227 /* Gain: '<S257>/Gain1' */
228 Vehicle_B.Gain1_i = 2.0 * Vehicle_B.Sum2_p;
229
230 /* Product: '<S257>/Product5' incorporates:
231  * Inport: '<Root>/xSunECI'
232  */
233 Vehicle_B.Product5_d = Vehicle_B.Gain1_i * Vehicle_U.xSunECI[1];
234
235 /* Product: '<S257>/Product6' */
236 Vehicle_B.Product6_b = Vehicle_B.Product1_fv * Vehicle_B.Product1_fv;
237
238 /* Product: '<S257>/Product7' */
239 Vehicle_B.Product7_h = Vehicle_B.Product2_oi * Vehicle_B.Product2_oi;
240
241 /* Sum: '<S257>/Sum3' incorporates:
242  * Constant: '<S257>/Constant'
243  */
244 Vehicle_B.Sum3_e = (0.5 - Vehicle_B.Product6_b) - Vehicle_B.Product7_h;
245
246 /* Gain: '<S257>/Gain2' */
247 Vehicle_B.Gain2_b = 2.0 * Vehicle_B.Sum3_e;
248
249 /* Product: '<S257>/Product8' incorporates:
250  * Inport: '<Root>/xSunECI'
251  */
252 Vehicle_B.Product8_g = Vehicle_B.Gain2_b * Vehicle_U.xSunECI[2];
253
254 /* Sum: '<S257>/Sum' */
255 Vehicle_B.Sum_n = (Vehicle_B.Product4_m + Vehicle_B.Product5_d) +
256     Vehicle_B.Product8_g;
257
258 /* SignalConversion generated from: '<S130>/Dot Product1' */
259 Vehicle_B.TmpSignalConversionAtDotProduct[0] = Vehicle_B.Sum_k;
260 Vehicle_B.TmpSignalConversionAtDotProduct[1] = Vehicle_B.Sum_lm;
261 Vehicle_B.TmpSignalConversionAtDotProduct[2] = Vehicle_B.Sum_n;
262
263 /* Product: '<S265>/Product' incorporates:
264  * Inport: '<Root>/q_eci2b'
265  */
266 Vehicle_B.Product_jz = Vehicle_U.q_eci2b[0] * Vehicle_U.q_eci2b[0];
267
268 /* Product: '<S265>/Product1' incorporates:
269  * Inport: '<Root>/q_eci2b'
270  */
271 Vehicle_B.Product1_bp = Vehicle_U.q_eci2b[1] * Vehicle_U.q_eci2b[1];
272
273 /* Product: '<S265>/Product2' incorporates:
274  * Inport: '<Root>/q_eci2b'
275  */
276 Vehicle_B.Product2_b = Vehicle_U.q_eci2b[2] * Vehicle_U.q_eci2b[2];
277
278 /* Product: '<S265>/Product3' incorporates:
279  * Inport: '<Root>/q_eci2b'
280  */
281 Vehicle_B.Product3_dh = Vehicle_U.q_eci2b[3] * Vehicle_U.q_eci2b[3];
282
283 /* Sum: '<S265>/Sum' */
284 Vehicle_B.Sum_d5 = ((Vehicle_B.Product_jz + Vehicle_B.Product1_bp) +
285     Vehicle_B.Product2_b) + Vehicle_B.Product3_dh;
286
287 /* Sqrt: '<S264>/sqrt' */
288 Vehicle_B.sqrt_k = std::sqrt(Vehicle_B.Sum_d5);
289
290 /* Product: '<S260>/Product2' incorporates:
291  * Inport: '<Root>/q_eci2b'
292  */
293 Vehicle_B.Product2_lq = Vehicle_U.q_eci2b[2] / Vehicle_B.sqrt_k;
294
295 /* Product: '<S261>/Product6' */
296 Vehicle_B.Product6_l = Vehicle_B.Product2_lq * Vehicle_B.Product2_lq;
297
298 /* Product: '<S260>/Product3' incorporates:
299  * Inport: '<Root>/q_eci2b'

```

```

300 */
301 Vehicle_B.Product3_h = Vehicle_U.q_eci2b[3] / Vehicle_B.sqrt_k;
302
303 /* Product: '<S261>/Product7' */
304 Vehicle_B.Product7_d = Vehicle_B.Product3_h * Vehicle_B.Product3_h;
305
306 /* Sum: '<S261>/Sum3' incorporates:
307 * Constant: '<S261>/Constant'
308 */
309 Vehicle_B.Sum3_j = (0.5 - Vehicle_B.Product6_l) - Vehicle_B.Product7_d;
310
311 /* Gain: '<S261>/Gain2' */
312 Vehicle_B.Gain2_f = 2.0 * Vehicle_B.Sum3_j;
313
314 /* Product: '<S261>/Product8' */
315 Vehicle_B.Product8_m = 0.0 * Vehicle_B.Gain2_f;
316
317 /* Product: '<S260>/Product1' incorporates:
318 * Inport: '<Root>/q_eci2b'
319 */
320 Vehicle_B.Product1_b1 = Vehicle_U.q_eci2b[1] / Vehicle_B.sqrt_k;
321
322 /* Product: '<S261>/Product' */
323 Vehicle_B.Product_h1 = Vehicle_B.Product1_b1 * Vehicle_B.Product2_lq;
324
325 /* Product: '<S260>/Product' incorporates:
326 * Inport: '<Root>/q_eci2b'
327 */
328 Vehicle_B.Product_f = Vehicle_U.q_eci2b[0] / Vehicle_B.sqrt_k;
329
330 /* Product: '<S261>/Product1' */
331 Vehicle_B.Product1_l = Vehicle_B.Product_f * Vehicle_B.Product3_h;
332
333 /* Sum: '<S261>/Sum1' */
334 Vehicle_B.Sum1_d = Vehicle_B.Product_h1 + Vehicle_B.Product1_l;
335
336 /* Gain: '<S261>/Gain' */
337 Vehicle_B.Gain_j = 2.0 * Vehicle_B.Sum1_d;
338
339 /* Product: '<S261>/Product4' */
340 Vehicle_B.Product4_o = Vehicle_B.Gain_j * 0.0;
341
342 /* Product: '<S261>/Product2' */
343 Vehicle_B.Product2_p = Vehicle_B.Product_f * Vehicle_B.Product2_lq;
344
345 /* Product: '<S261>/Product3' */
346 Vehicle_B.Product3_lx = Vehicle_B.Product1_b1 * Vehicle_B.Product3_h;
347
348 /* Sum: '<S261>/Sum2' */
349 Vehicle_B.Sum2_p3 = Vehicle_B.Product3_lx - Vehicle_B.Product2_p;
350
351 /* Gain: '<S261>/Gain1' */
352 Vehicle_B.Gain1_o = 2.0 * Vehicle_B.Sum2_p3;
353
354 /* Product: '<S261>/Product5' */
355 Vehicle_B.Product5_i = Vehicle_B.Gain1_o;
356
357 /* Sum: '<S261>/Sum' */
358 Vehicle_B.Sum_kk = (Vehicle_B.Product8_m + Vehicle_B.Product4_o) +
359 Vehicle_B.Product5_i;
360
361 /* Product: '<S262>/Product' */
362 Vehicle_B.Product_bg = Vehicle_B.Product1_b1 * Vehicle_B.Product2_lq;
363
364 /* Product: '<S262>/Product1' */
365 Vehicle_B.Product1_l3 = Vehicle_B.Product_f * Vehicle_B.Product3_h;
366
367 /* Sum: '<S262>/Sum1' */
368 Vehicle_B.Sum1_l = Vehicle_B.Product_bg - Vehicle_B.Product1_l3;
369
370 /* Gain: '<S262>/Gain' */
371 Vehicle_B.Gain_i = 2.0 * Vehicle_B.Sum1_l;
372
373 /* Product: '<S262>/Product4' */
374 Vehicle_B.Product4_pt = 0.0 * Vehicle_B.Gain_i;
375
376 /* Product: '<S262>/Product6' */
377 Vehicle_B.Product6_p = Vehicle_B.Product1_b1 * Vehicle_B.Product1_b1;
378
379 /* Product: '<S262>/Product7' */
380 Vehicle_B.Product7_dj = Vehicle_B.Product3_h * Vehicle_B.Product3_h;
381
382 /* Sum: '<S262>/Sum3' incorporates:
383 * Constant: '<S262>/Constant'
384 */
385 Vehicle_B.Sum3_o = (0.5 - Vehicle_B.Product6_p) - Vehicle_B.Product7_dj;
386
387 /* Gain: '<S262>/Gain2' */
388 Vehicle_B.Gain2_j = 2.0 * Vehicle_B.Sum3_o;
389
390 /* Product: '<S262>/Product8' */
391 Vehicle_B.Product8_e = Vehicle_B.Gain2_j * 0.0;
392
393 /* Product: '<S262>/Product2' */
394 Vehicle_B.Product2_mf = Vehicle_B.Product_f * Vehicle_B.Product1_b1;
395
396 /* Product: '<S262>/Product3' */
397 Vehicle_B.Product3_k = Vehicle_B.Product2_lq * Vehicle_B.Product3_h;
398
399 /* Sum: '<S262>/Sum2' */
400 Vehicle_B.Sum2_a = Vehicle_B.Product2_mf + Vehicle_B.Product3_k;
401
402 /* Gain: '<S262>/Gain1' */
403 Vehicle_B.Gain1_oi = 2.0 * Vehicle_B.Sum2_a;

```

```
404 /* Product: '<S262>/Product5' */
405 Vehicle_B.Product5_b = Vehicle_B.Gain1_oi;
406
407 /* Sum: '<S262>/Sum' */
408 Vehicle_B.Sum_ln = (Vehicle_B.Product4_pt + Vehicle_B.Product8_e) +
409     Vehicle_B.Product5_b;
410
411 /* Product: '<S263>/Product' */
412 Vehicle_B.Product_l = Vehicle_B.Product1_b1 * Vehicle_B.Product3_h;
413
414 /* Product: '<S263>/Product1' */
415 Vehicle_B.Product1_id = Vehicle_B.Product_f * Vehicle_B.Product2_lq;
416
417 /* Sum: '<S263>/Sum1' */
418 Vehicle_B.Sum1_d1 = Vehicle_B.Product_l + Vehicle_B.Product1_id;
419
420 /* Gain: '<S263>/Gain' */
421 Vehicle_B.Gain_o = 2.0 * Vehicle_B.Sum1_d1;
422
423 /* Product: '<S263>/Product4' */
424 Vehicle_B.Product4_d = 0.0 * Vehicle_B.Gain_o;
425
426 /* Product: '<S263>/Product2' */
427 Vehicle_B.Product2_f = Vehicle_B.Product_f * Vehicle_B.Product1_b1;
428
429 /* Product: '<S263>/Product3' */
430 Vehicle_B.Product3_gx = Vehicle_B.Product2_lq * Vehicle_B.Product3_h;
431
432 /* Sum: '<S263>/Sum2' */
433 Vehicle_B.Sum2_l = Vehicle_B.Product3_gx - Vehicle_B.Product2_f;
434
435 /* Gain: '<S263>/Gain1' */
436 Vehicle_B.Gain1_g = 2.0 * Vehicle_B.Sum2_l;
437
438 /* Product: '<S263>/Product5' */
439 Vehicle_B.Product5_n = Vehicle_B.Gain1_g * 0.0;
440
441 /* Product: '<S263>/Product6' */
442 Vehicle_B.Product6_j = Vehicle_B.Product1_b1 * Vehicle_B.Product1_b1;
443
444 /* Product: '<S263>/Product7' */
445 Vehicle_B.Product7_g = Vehicle_B.Product2_lq * Vehicle_B.Product2_lq;
446
447 /* Sum: '<S263>/Sum3' incorporates:
448  * Constant: '<S263>/Constant'
449  */
450 Vehicle_B.Sum3_b = (0.5 - Vehicle_B.Product6_j) - Vehicle_B.Product7_g;
451
452 /* Gain: '<S263>/Gain2' */
453 Vehicle_B.Gain2_o = 2.0 * Vehicle_B.Sum3_b;
454
455 /* Product: '<S263>/Product8' */
456 Vehicle_B.Product8_a = Vehicle_B.Gain2_o;
457
458 /* Sum: '<S263>/Sum' */
459 Vehicle_B.Sum_kb = (Vehicle_B.Product4_d + Vehicle_B.Product5_n) +
460     Vehicle_B.Product8_a;
461
462 /* SignalConversion generated from: '<S130>/Dot Product2' */
463 Vehicle_B.TmpSignalConversionAtDotProdu_d[0] = Vehicle_B.Sum_kk;
464 Vehicle_B.TmpSignalConversionAtDotProdu_d[1] = Vehicle_B.Sum_ln;
465 Vehicle_B.TmpSignalConversionAtDotProdu_d[2] = Vehicle_B.Sum_kb;
466
467 /* DotProduct: '<S130>/Dot Product3' */
468 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProduct[0];
469 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_d[0];
470 tmpForInput_idx_2 = tmpForInput_idx_1 * tmpForInput_idx_0;
471
472 /* DotProduct: '<S130>/Dot Product1' */
473 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProduct[0];
474 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProduct[0];
475 tmpForInput = tmpForInput_idx_1 * tmpForInput_idx_0;
476
477 /* DotProduct: '<S130>/Dot Product2' */
478 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_d[0];
479 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_d[0];
480 tmpForInput_0 = tmpForInput_idx_1 * tmpForInput_idx_0;
481
482 /* DotProduct: '<S130>/Dot Product3' */
483 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProduct[1];
484 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_d[1];
485 tmpForInput_idx_2 += tmpForInput_idx_1 * tmpForInput_idx_0;
486
487 /* DotProduct: '<S130>/Dot Product1' */
488 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProduct[1];
489 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProduct[1];
490 tmpForInput += tmpForInput_idx_1 * tmpForInput_idx_0;
491
492 /* DotProduct: '<S130>/Dot Product2' */
493 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_d[1];
494 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_d[1];
495 tmpForInput_0 += tmpForInput_idx_1 * tmpForInput_idx_0;
496
497 /* DotProduct: '<S130>/Dot Product3' */
498 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProduct[2];
499 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_d[2];
500 tmpForInput_idx_2 += tmpForInput_idx_1 * tmpForInput_idx_0;
501
502 /* DotProduct: '<S130>/Dot Product1' */
503 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProduct[2];
504 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProduct[2];
505 tmpForInput += tmpForInput_idx_1 * tmpForInput_idx_0;
506
507
```

```

508 /* DotProduct: '<S130>/Dot Product2' */
509 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_d[2];
510 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_d[2];
511 tmpForInput_0 += tmpForInput_idx_1 * tmpForInput_idx_0;
512
513 /* DotProduct: '<S130>/Dot Product3' */
514 Vehicle_B.DotProduct3 = tmpForInput_idx_2;
515
516 /* DotProduct: '<S130>/Dot Product1' */
517 Vehicle_B.DotProduct1 = tmpForInput;
518
519 /* DotProduct: '<S130>/Dot Product2' */
520 Vehicle_B.DotProduct2 = tmpForInput_0;
521
522 /* Product: '<S130>/Divide1' */
523 Vehicle_B.Divide1_k = Vehicle_B.DotProduct1 * Vehicle_B.DotProduct2;
524
525 /* Sqrt: '<S130>/Sqrt3' */
526 Vehicle_B.Sqrt3 = std::sqrt(Vehicle_B.Divide1_k);
527
528 /* Product: '<S130>/Divide' */
529 Vehicle_B.Divide_o = Vehicle_B.DotProduct3 / Vehicle_B.Sqrt3;
530
531 /* Bias: '<S130>/Bias' */
532 Vehicle_B.Bias_k = Vehicle_B.Divide_o - 1.0;
533
534 /* Abs: '<S130>/Abs' */
535 Vehicle_B.Abs = std::abs(Vehicle_B.Bias_k);
536
537 /* Bias: '<S130>/Bias1' */
538 Vehicle_B.Bias1_j = Vehicle_B.Divide_o + 1.0;
539
540 /* Abs: '<S130>/Abs1' */
541 Vehicle_B.Abs1_d = std::abs(Vehicle_B.Bias1_j);
542
543 /* If: '<S130>/If' */
544 if (Vehicle_B.Abs < 1.0E-6) {
545     /* Outputs for IfAction SubSystem: '<S130>/If Action Subsystem' incorporates:
546      * ActionPort: '<S143>/Action Port'
547      */
548     // Vehicle_IfActionSubsystem(&Vehicle_B.Merge_ip);
549     Vehicle_B.Merge_ip = 1.0;
550     /* End of Outputs for SubSystem: '<S130>/If Action Subsystem' */
551 } else if (Vehicle_B.Abs1_d < 1.0E-6) {
552     /* Outputs for IfAction SubSystem: '<S130>/If Action Subsystem1' incorporates:
553      * ActionPort: '<S144>/Action Port'
554      */
555     //Vehicle_IfActionSubsystem1(&Vehicle_B.Merge_ip);
556     Vehicle_B.Merge_ip = -1.0;
557     /* End of Outputs for SubSystem: '<S130>/If Action Subsystem1' */
558 } else {
559     /* Outputs for IfAction SubSystem: '<S130>/If Action Subsystem2' incorporates:
560      * ActionPort: '<S145>/Action Port'
561      */
562     // Vehicle_IfActionSubsystem2(&Vehicle_B.Merge_ip);
563     Vehicle_B.Merge_ip = 0.0;
564     /* End of Outputs for SubSystem: '<S130>/If Action Subsystem2' */
565 }
566
567 /* End of If: '<S130>/If' */
568
569 /* RelationalOperator: '<S131>/Compare' incorporates:
570  * Constant: '<S131>/Constant'
571  */
572 Vehicle_B.Compare_k = (Vehicle_B.Merge_m != 0.0);
573
574 /* RelationalOperator: '<S132>/Compare' incorporates:
575  * Constant: '<S132>/Constant'
576  */
577 Vehicle_B.Compare_a = (Vehicle_B.Merge_ip != 0.0);
578
579 /* Logic: '<S124>/OR' */
580 Vehicle_B.OR = (Vehicle_B.Compare_k || Vehicle_B.Compare_a);
581
582 /* DotProduct: '<S136>/Dot Product3' */
583 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProduct[0];
584 tmpForInput_idx_1 = 0.0 * tmpForInput_idx_0;
585
586 /* DotProduct: '<S136>/Dot Product2' */
587 tmpForInput_idx_2 = Vehicle_B.TmpSignalConversionAtDotProduct[0];
588 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProduct[0];
589 tmpForInput = tmpForInput_idx_2 * tmpForInput_idx_0;
590
591 /* DotProduct: '<S163>/Dot Product3' */
592 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProduct[0];
593 tmpForInput_0 = 0.0 * tmpForInput_idx_0;
594
595 /* DotProduct: '<S163>/Dot Product2' */
596 tmpForInput_idx_2 = Vehicle_B.TmpSignalConversionAtDotProduct[0];
597 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProduct[0];
598 tmpForInput_1 = tmpForInput_idx_2 * tmpForInput_idx_0;
599
600 /* DotProduct: '<S136>/Dot Product3' */
601 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProduct[1];
602 tmpForInput_idx_1 += 0.0 * tmpForInput_idx_0;
603
604 /* DotProduct: '<S136>/Dot Product2' */
605 tmpForInput_idx_2 = Vehicle_B.TmpSignalConversionAtDotProduct[1];
606 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProduct[1];
607 tmpForInput += tmpForInput_idx_2 * tmpForInput_idx_0;
608
609 /* DotProduct: '<S163>/Dot Product3' */
610 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProduct[1];
611 tmpForInput_0 += 0.0 * tmpForInput_idx_0;

```

```

612
613 /* DotProduct: '<S163>/Dot Product2' */
614 tmpForInput_idx_2 = Vehicle_B.TmpSignalConversionAtDotProduct[1];
615 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProduct[1];
616 tmpForInput_1 += tmpForInput_idx_2 * tmpForInput_idx_0;
617
618 /* DotProduct: '<S136>/Dot Product3' */
619 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProduct[2];
620 tmpForInput_idx_1 += -tmpForInput_idx_0;
621
622 /* DotProduct: '<S136>/Dot Product2' */
623 tmpForInput_idx_2 = Vehicle_B.TmpSignalConversionAtDotProduct[2];
624 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProduct[2];
625 tmpForInput += tmpForInput_idx_2 * tmpForInput_idx_0;
626
627 /* DotProduct: '<S163>/Dot Product3' */
628 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProduct[2];
629 tmpForInput_0 += -tmpForInput_idx_0;
630
631 /* DotProduct: '<S163>/Dot Product2' */
632 tmpForInput_idx_2 = Vehicle_B.TmpSignalConversionAtDotProduct[2];
633 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProduct[2];
634 tmpForInput_1 += tmpForInput_idx_2 * tmpForInput_idx_0;
635
636 /* DotProduct: '<S136>/Dot Product3' */
637 Vehicle_B.DotProduct3_o = tmpForInput_idx_1;
638
639 /* DotProduct: '<S136>/Dot Product2' */
640 Vehicle_B.DotProduct2_c = tmpForInput;
641
642 /* Product: '<S136>/Divide1' */
643 Vehicle_B.Divide1_d = Vehicle_ConstB.DotProduct1_c * Vehicle_B.DotProduct2_c;
644
645 /* Sqrt: '<S136>/Sqrt3' */
646 Vehicle_B.Sqrt3_j = std::sqrt(Vehicle_B.Divide1_d);
647
648 /* Sum: '<S136>/Add2' */
649 Vehicle_B.Add2 = Vehicle_B.DotProduct3_o + Vehicle_B.Sqrt3_j;
650
651 /* Product: '<S176>/Product' */
652 Vehicle_B.Product_i2 = Vehicle_B.Add2 * Vehicle_B.Add2;
653
654 /* DotProduct: '<S163>/Dot Product3' */
655 Vehicle_B.DotProduct3_p = tmpForInput_0;
656
657 /* DotProduct: '<S163>/Dot Product2' */
658 Vehicle_B.DotProduct2_f = tmpForInput_1;
659
660 /* Product: '<S163>/Divide1' */
661 Vehicle_B.Divide1_b = Vehicle_ConstB.DotProduct1_g * Vehicle_B.DotProduct2_f;
662
663 /* Sqrt: '<S163>/Sqrt3' */
664 Vehicle_B.Sqrt3_l = std::sqrt(Vehicle_B.Divide1_b);
665
666 /* Product: '<S163>/Divide' */
667 Vehicle_B.Divide_k = Vehicle_B.DotProduct3_p / Vehicle_B.Sqrt3_l;
668
669 /* Bias: '<S163>/Bias' */
670 Vehicle_B.Bias_b = Vehicle_B.Divide_k + -1.0;
671
672 /* Abs: '<S163>/Abs' */
673 Vehicle_B.Abs_o = std::abs(Vehicle_B.Bias_b);
674
675 /* Bias: '<S163>/Bias1' */
676 Vehicle_B.Bias1_m = Vehicle_B.Divide_k + 1.0;
677
678 /* Abs: '<S163>/Abs1' */
679 Vehicle_B.Abs1_l = std::abs(Vehicle_B.Bias1_m);
680
681 /* If: '<S163>/If' */
682 if (Vehicle_B.Abs_o < 1.0E-6) {
683     /* Outputs for IfAction SubSystem: '<S163>/If Action Subsystem' incorporates:
684      * ActionPort: '<S169>/Action Port'
685      */
686     //Vehicle_IfActionSubsystem(&Vehicle_B.Merge_d);
687     Vehicle_B.Merge_d = 1.0;
688     /* End of Outputs for SubSystem: '<S163>/If Action Subsystem' */
689 } else if (Vehicle_B.Abs1_l < 1.0E-6) {
690     /* Outputs for IfAction SubSystem: '<S163>/If Action Subsystem1' incorporates:
691      * ActionPort: '<S170>/Action Port'
692      */
693     //Vehicle_IfActionSubsystem1(&Vehicle_B.Merge_d);
694     Vehicle_B.Merge_d = -1.0;
695     /* End of Outputs for SubSystem: '<S163>/If Action Subsystem1' */
696 } else {
697     /* Outputs for IfAction SubSystem: '<S163>/If Action Subsystem2' incorporates:
698      * ActionPort: '<S171>/Action Port'
699      */
700     //Vehicle_IfActionSubsystem2(&Vehicle_B.Merge_d);
701     Vehicle_B.Merge_d = 0.0;
702     /* End of Outputs for SubSystem: '<S163>/If Action Subsystem2' */
703 }
704
705 /* End of If: '<S163>/If' */
706
707 /* RelationalOperator: '<S164>/Compare' incorporates:
708  * Constant: '<S164>/Constant'
709  */
710 Vehicle_B.Compare_m = (Vehicle_B.Merge_d != -1.0);
711
712 /* Switch: '<S136>/is 180deg Rot' */
713 if (Vehicle_B.Compare_m) {
714     /* Product: '<S168>/j x i' */
715     Vehicle_B.jxi_k = 0.0 * Vehicle_B.Sum_k;

```



```

716 /* Product: '<S168>/i x k' */
717 Vehicle_B.ixk_a = 0.0 * Vehicle_B.Sum_n;
718
719 /* Product: '<S168>/k x j' */
720 Vehicle_B.kxj_c = -Vehicle_B.Sum_lm;
721
722 /* Product: '<S167>/i x j' */
723 Vehicle_B.ixj_o = 0.0 * Vehicle_B.Sum_lm;
724
725 /* Product: '<S167>/k x i' */
726 Vehicle_B.kxi_p = -Vehicle_B.Sum_k;
727
728 /* Product: '<S167>/j x k' */
729 Vehicle_B.jxk_er = 0.0 * Vehicle_B.Sum_n;
730
731 /* Sum: '<S162>/Sum' */
732 Vehicle_B.Sum_ff[0] = Vehicle_B.jxk_er - Vehicle_B.kxj_c;
733 Vehicle_B.Sum_ff[1] = Vehicle_B.kxi_p - Vehicle_B.ixk_a;
734 Vehicle_B.Sum_ff[2] = Vehicle_B.ixj_o - Vehicle_B.jxi_k;
735
736 /* Switch: '<S136>/is 180deg Rot' */
737 Vehicle_B.is180degRot[0] = Vehicle_B.Sum_ff[0];
738 Vehicle_B.is180degRot[1] = Vehicle_B.Sum_ff[1];
739 Vehicle_B.is180degRot[2] = Vehicle_B.Sum_ff[2];
740 } else {
741 /* Switch: '<S136>/is 180deg Rot' */
742 Vehicle_B.is180degRot[0] = Vehicle_ConstB.Sum[0];
743 Vehicle_B.is180degRot[1] = Vehicle_ConstB.Sum[1];
744 Vehicle_B.is180degRot[2] = Vehicle_ConstB.Sum[2];
745 }
746
747 /* End of Switch: '<S136>/is 180deg Rot' */
748
749 /* Product: '<S176>/Product1' */
750 Vehicle_B.Product1_g = Vehicle_B.is180degRot[0] * Vehicle_B.is180degRot[0];
751
752 /* Product: '<S176>/Product2' */
753 Vehicle_B.Product2_g = Vehicle_B.is180degRot[1] * Vehicle_B.is180degRot[1];
754
755 /* Product: '<S176>/Product3' */
756 Vehicle_B.Product3_aj = Vehicle_B.is180degRot[2] * Vehicle_B.is180degRot[2];
757
758 /* Sum: '<S176>/Sum' */
759 Vehicle_B.Sum_c = ((Vehicle_B.Product_i2 + Vehicle_B.Product1_g) +
760 Vehicle_B.Product2_g) + Vehicle_B.Product3_aj;
761
762 /* Sqrt: '<S175>/sqrt' */
763 Vehicle_B.sqrt_h = std::sqrt(Vehicle_B.Sum_c);
764
765 /* Product: '<S166>/Product' */
766 Vehicle_B.Product_fv = Vehicle_B.Add2 / Vehicle_B.sqrt_h;
767
768 /* Product: '<S166>/Product2' */
769 Vehicle_B.Product2_a = Vehicle_B.is180degRot[1] / Vehicle_B.sqrt_h;
770
771 /* Product: '<S155>/Product' */
772 Vehicle_B.Product_l5 = Vehicle_B.Product_fv * Vehicle_B.Product_fv;
773
774 /* Product: '<S166>/Product1' */
775 Vehicle_B.Product1_n = Vehicle_B.is180degRot[0] / Vehicle_B.sqrt_h;
776
777 /* Product: '<S155>/Product1' */
778 Vehicle_B.Product1_lg = Vehicle_B.Product1_n * Vehicle_B.Product1_n;
779
780 /* Product: '<S155>/Product2' */
781 Vehicle_B.Product2_fh = Vehicle_B.Product2_a * Vehicle_B.Product2_a;
782
783 /* Product: '<S166>/Product3' */
784 Vehicle_B.Product3_pe = Vehicle_B.is180degRot[2] / Vehicle_B.sqrt_h;
785
786 /* Product: '<S155>/Product3' */
787 Vehicle_B.Product3_l0 = Vehicle_B.Product3_pe * Vehicle_B.Product3_pe;
788
789 /* Sum: '<S155>/Sum' */
790 Vehicle_B.Sum_n3 = ((Vehicle_B.Product_l5 + Vehicle_B.Product1_lg) +
791 Vehicle_B.Product2_fh) + Vehicle_B.Product3_l0;
792
793 /* Sqrt: '<S154>/sqrt' */
794 Vehicle_B.sqrt_mv = std::sqrt(Vehicle_B.Sum_n3);
795
796 /* Product: '<S150>/Product2' */
797 Vehicle_B.Product2_ke = Vehicle_B.Product2_a / Vehicle_B.sqrt_mv;
798
799 /* Product: '<S151>/Product6' */
800 Vehicle_B.Product6_e = Vehicle_B.Product2_ke * Vehicle_B.Product2_ke;
801
802 /* Product: '<S150>/Product3' */
803 Vehicle_B.Product3_cc = Vehicle_B.Product3_pe / Vehicle_B.sqrt_mv;
804
805 /* Product: '<S151>/Product7' */
806 Vehicle_B.Product7_a = Vehicle_B.Product3_cc * Vehicle_B.Product3_cc;
807
808 /* Sum: '<S151>/Sum3' incorporates:
809 * Constant: '<S151>/Constant'
810 */
811 Vehicle_B.Sum3_b5 = (0.5 - Vehicle_B.Product6_e) - Vehicle_B.Product7_a;
812
813 /* Gain: '<S151>/Gain2' */
814 Vehicle_B.Gain2_p = 2.0 * Vehicle_B.Sum3_b5;
815
816 /* Product: '<S151>/Product8' */
817 Vehicle_B.Product8_k = Vehicle_ConstB.Sum_d0[0] * Vehicle_B.Gain2_p;
818
819

```



```

820 /* Product: '<S150>/Product1' */
821 Vehicle_B.Product1_10 = Vehicle_B.Product1_n / Vehicle_B.sqrt_mv;
822
823 /* Product: '<S151>/Product' */
824 Vehicle_B.Product_da = Vehicle_B.Product1_10 * Vehicle_B.Product2_ke;
825
826 /* Product: '<S150>/Product' */
827 Vehicle_B.Product_m = Vehicle_B.Product_fv / Vehicle_B.sqrt_mv;
828
829 /* Product: '<S151>/Product1' */
830 Vehicle_B.Product1_dj = Vehicle_B.Product_m * Vehicle_B.Product3_cc;
831
832 /* Sum: '<S151>/Sum1' */
833 Vehicle_B.Sum1_n = Vehicle_B.Product_da + Vehicle_B.Product1_dj;
834
835 /* Gain: '<S151>/Gain' */
836 Vehicle_B.Gain_l = 2.0 * Vehicle_B.Sum1_n;
837
838 /* Product: '<S151>/Product4' */
839 Vehicle_B.Product4_c = Vehicle_B.Gain_l * Vehicle_ConstB.Sum_d0[1];
840
841 /* Product: '<S151>/Product2' */
842 Vehicle_B.Product2_gk = Vehicle_B.Product_m * Vehicle_B.Product2_ke;
843
844 /* Product: '<S151>/Product3' */
845 Vehicle_B.Product3_i = Vehicle_B.Product1_10 * Vehicle_B.Product3_cc;
846
847 /* Sum: '<S151>/Sum2' */
848 Vehicle_B.Sum2_g = Vehicle_B.Product3_i - Vehicle_B.Product2_gk;
849
850 /* Gain: '<S151>/Gain1' */
851 Vehicle_B.Gain1_e = 2.0 * Vehicle_B.Sum2_g;
852
853 /* Product: '<S151>/Product5' */
854 Vehicle_B.Product5_o = Vehicle_B.Gain1_e * Vehicle_ConstB.Sum_d0[2];
855
856 /* Sum: '<S151>/Sum' */
857 Vehicle_B.Sum_oz = (Vehicle_B.Product8_k + Vehicle_B.Product4_c) +
858     Vehicle_B.Product5_o;
859
860 /* Product: '<S152>/Product' */
861 Vehicle_B.Product_g = Vehicle_B.Product1_10 * Vehicle_B.Product2_ke;
862
863 /* Product: '<S152>/Product1' */
864 Vehicle_B.Product1_ew = Vehicle_B.Product_m * Vehicle_B.Product3_cc;
865
866 /* Sum: '<S152>/Sum1' */
867 Vehicle_B.Sum1_m = Vehicle_B.Product_g - Vehicle_B.Product1_ew;
868
869 /* Gain: '<S152>/Gain' */
870 Vehicle_B.Gain_b = 2.0 * Vehicle_B.Sum1_m;
871
872 /* Product: '<S152>/Product4' */
873 Vehicle_B.Product4_a = Vehicle_ConstB.Sum_d0[0] * Vehicle_B.Gain_b;
874
875 /* Product: '<S152>/Product6' */
876 Vehicle_B.Product6_o = Vehicle_B.Product1_10 * Vehicle_B.Product1_10;
877
878 /* Product: '<S152>/Product7' */
879 Vehicle_B.Product7_k = Vehicle_B.Product3_cc * Vehicle_B.Product3_cc;
880
881 /* Sum: '<S152>/Sum3' incorporates:
882  * Constant: '<S152>/Constant'
883  */
884 Vehicle_B.Sum3_bi = (0.5 - Vehicle_B.Product6_o) - Vehicle_B.Product7_k;
885
886 /* Gain: '<S152>/Gain2' */
887 Vehicle_B.Gain2_l = 2.0 * Vehicle_B.Sum3_bi;
888
889 /* Product: '<S152>/Product8' */
890 Vehicle_B.Product8_me = Vehicle_B.Gain2_l * Vehicle_ConstB.Sum_d0[1];
891
892 /* Product: '<S152>/Product2' */
893 Vehicle_B.Product2_fhq = Vehicle_B.Product_m * Vehicle_B.Product1_10;
894
895 /* Product: '<S152>/Product3' */
896 Vehicle_B.Product3_n = Vehicle_B.Product2_ke * Vehicle_B.Product3_cc;
897
898 /* Sum: '<S152>/Sum2' */
899 Vehicle_B.Sum2_la = Vehicle_B.Product2_fhq + Vehicle_B.Product3_n;
900
901 /* Gain: '<S152>/Gain1' */
902 Vehicle_B.Gain1_ox = 2.0 * Vehicle_B.Sum2_la;
903
904 /* Product: '<S152>/Product5' */
905 Vehicle_B.Product5_l = Vehicle_B.Gain1_ox * Vehicle_ConstB.Sum_d0[2];
906
907 /* Sum: '<S152>/Sum' */
908 Vehicle_B.Sum_oa = (Vehicle_B.Product4_a + Vehicle_B.Product8_me) +
909     Vehicle_B.Product5_l;
910
911 /* Product: '<S153>/Product' */
912 Vehicle_B.Product_ew = Vehicle_B.Product1_10 * Vehicle_B.Product3_cc;
913
914 /* Product: '<S153>/Product1' */
915 Vehicle_B.Product1_m = Vehicle_B.Product_m * Vehicle_B.Product2_ke;
916
917 /* Sum: '<S153>/Sum1' */
918 Vehicle_B.Sum1_h = Vehicle_B.Product_ew + Vehicle_B.Product1_m;
919
920 /* Gain: '<S153>/Gain' */
921 Vehicle_B.Gain_kp = 2.0 * Vehicle_B.Sum1_h;
922
923 /* Product: '<S153>/Product4' */

```

```

924 Vehicle_B.Product4_g = Vehicle_ConstB.Sum_d0[0] * Vehicle_B.Gain_kp;
925
926 /* Product: '<S153>/Product2' */
927 Vehicle_B.Product2_gh = Vehicle_B.Product_m * Vehicle_B.Product1_l0;
928
929 /* Product: '<S153>/Product3' */
930 Vehicle_B.Product3_k4 = Vehicle_B.Product2_ke * Vehicle_B.Product3_cc;
931
932 /* Sum: '<S153>/Sum2' */
933 Vehicle_B.Sum2_i = Vehicle_B.Product3_k4 - Vehicle_B.Product2_gh;
934
935 /* Gain: '<S153>/Gain1' */
936 Vehicle_B.Gain1_f = 2.0 * Vehicle_B.Sum2_i;
937
938 /* Product: '<S153>/Product5' */
939 Vehicle_B.Product5_ij = Vehicle_B.Gain1_f * Vehicle_ConstB.Sum_d0[1];
940
941 /* Product: '<S153>/Product6' */
942 Vehicle_B.Product6_c = Vehicle_B.Product1_l0 * Vehicle_B.Product1_l0;
943
944 /* Product: '<S153>/Product7' */
945 Vehicle_B.Product7_a4 = Vehicle_B.Product2_ke * Vehicle_B.Product2_ke;
946
947 /* Sum: '<S153>/Sum3' incorporates:
948    * Constant: '<S153>/Constant'
949    */
950 Vehicle_B.Sum3_d = (0.5 - Vehicle_B.Product6_c) - Vehicle_B.Product7_a4;
951
952 /* Gain: '<S153>/Gain2' */
953 Vehicle_B.Gain2_n = 2.0 * Vehicle_B.Sum3_d;
954
955 /* Product: '<S153>/Product8' */
956 Vehicle_B.Product8_f = Vehicle_B.Gain2_n * Vehicle_ConstB.Sum_d0[2];
957
958 /* Sum: '<S153>/Sum' */
959 Vehicle_B.Sum_nx = (Vehicle_B.Product4_g + Vehicle_B.Product5_ij) +
960     Vehicle_B.Product8_f;
961
962 /* SignalConversion generated from: '<S178>/Dot Product1' */
963 Vehicle_B.TmpSignalConversionAtDotProdu_k[0] = Vehicle_B.Sum_oz;
964 Vehicle_B.TmpSignalConversionAtDotProdu_k[1] = Vehicle_B.Sum_oa;
965 Vehicle_B.TmpSignalConversionAtDotProdu_k[2] = Vehicle_B.Sum_nx;
966
967 /* Product: '<S200>/j x k' */
968 Vehicle_B.jxk_b = Vehicle_B.Sum_lm * Vehicle_B.Sum_kb;
969
970 /* Product: '<S200>/k x i' */
971 Vehicle_B.kxi_b = Vehicle_B.Sum_n * Vehicle_B.Sum_kk;
972
973 /* Product: '<S200>/i x j' */
974 Vehicle_B.ixj_h = Vehicle_B.Sum_k * Vehicle_B.Sum_ln;
975
976 /* Product: '<S201>/k x j' */
977 Vehicle_B.kxj_l = Vehicle_B.Sum_n * Vehicle_B.Sum_ln;
978
979 /* Product: '<S201>/i x k' */
980 Vehicle_B.ixk_e = Vehicle_B.Sum_k * Vehicle_B.Sum_kb;
981
982 /* Product: '<S201>/j x i' */
983 Vehicle_B.jxi_n = Vehicle_B.Sum_lm * Vehicle_B.Sum_kk;
984
985 /* Sum: '<S198>/Sum' */
986 Vehicle_B.Sum_m[0] = Vehicle_B.jxk_b - Vehicle_B.kxj_l;
987 Vehicle_B.Sum_m[1] = Vehicle_B.kxi_b - Vehicle_B.ixk_e;
988 Vehicle_B.Sum_m[2] = Vehicle_B.ixj_h - Vehicle_B.jxi_n;
989
990 /* Product: '<S202>/j x k' */
991 Vehicle_B.jxk_e = Vehicle_B.Sum_m[1] * Vehicle_B.Sum_n;
992
993 /* Product: '<S202>/k x i' */
994 Vehicle_B.kxi_es = Vehicle_B.Sum_m[2] * Vehicle_B.Sum_k;
995
996 /* Product: '<S202>/i x j' */
997 Vehicle_B.ixj_n = Vehicle_B.Sum_m[0] * Vehicle_B.Sum_lm;
998
999 /* Product: '<S203>/k x j' */
1000 Vehicle_B.kxj_j = Vehicle_B.Sum_m[2] * Vehicle_B.Sum_lm;
1001
1002 /* Product: '<S203>/i x k' */
1003 Vehicle_B.ixk_j = Vehicle_B.Sum_m[0] * Vehicle_B.Sum_n;
1004
1005 /* Product: '<S203>/j x i' */
1006 Vehicle_B.jxi_f = Vehicle_B.Sum_m[1] * Vehicle_B.Sum_k;
1007
1008 /* Sum: '<S199>/Sum' */
1009 Vehicle_B.Sum_pb[0] = Vehicle_B.jxk_e - Vehicle_B.kxj_j;
1010 Vehicle_B.Sum_pb[1] = Vehicle_B.kxi_es - Vehicle_B.ixk_j;
1011 Vehicle_B.Sum_pb[2] = Vehicle_B.ixj_n - Vehicle_B.jxi_f;
1012
1013 /* Product: '<S161>/Product' */
1014 Vehicle_B.Product_dj = Vehicle_B.Product_fv * Vehicle_B.Product_fv;
1015
1016 /* Product: '<S161>/Product1' */
1017 Vehicle_B.Product1_ig = Vehicle_B.Product1_n * Vehicle_B.Product1_n;
1018
1019 /* Product: '<S161>/Product2' */
1020 Vehicle_B.Product2_ad = Vehicle_B.Product2_a * Vehicle_B.Product2_a;
1021
1022 /* Product: '<S161>/Product3' */
1023 Vehicle_B.Product3_hf = Vehicle_B.Product3_pe * Vehicle_B.Product3_pe;
1024
1025 /* Sum: '<S161>/Sum' */
1026 Vehicle_B.Sum_gt = ((Vehicle_B.Product_dj + Vehicle_B.Product1_ig) +
1027     Vehicle_B.Product2_ad) + Vehicle_B.Product3_hf;

```

```
1028 /* Sqrt: '<S160>/sqrt' */
1029 Vehicle_B.sqrt_i = std::sqrt(Vehicle_B.Sum_gt);
1030
1031
1032 /* Product: '<S156>/Product2' */
1033 Vehicle_B.Product2_nx = Vehicle_B.Product2_a / Vehicle_B.sqrt_i;
1034
1035 /* Product: '<S157>/Product6' */
1036 Vehicle_B.Product6_a = Vehicle_B.Product2_nx * Vehicle_B.Product2_nx;
1037
1038 /* Product: '<S156>/Product3' */
1039 Vehicle_B.Product3_ce = Vehicle_B.Product3_pe / Vehicle_B.sqrt_i;
1040
1041 /* Product: '<S157>/Product7' */
1042 Vehicle_B.Product7_ha = Vehicle_B.Product3_ce * Vehicle_B.Product3_ce;
1043
1044 /* Sum: '<S157>/Sum3' incorporates:
1045  * Constant: '<S157>/Constant'
1046  */
1047 Vehicle_B.Sum3_l = (0.5 - Vehicle_B.Product6_a) - Vehicle_B.Product7_ha;
1048
1049 /* Gain: '<S157>/Gain2' */
1050 Vehicle_B.Gain2_om = 2.0 * Vehicle_B.Sum3_l;
1051
1052 /* Product: '<S157>/Product8' */
1053 Vehicle_B.Product8_p = Vehicle_B.Sum_pb[0] * Vehicle_B.Gain2_om;
1054
1055 /* Product: '<S156>/Product1' */
1056 Vehicle_B.Product1_a = Vehicle_B.Product1_n / Vehicle_B.sqrt_i;
1057
1058 /* Product: '<S157>/Product' */
1059 Vehicle_B.Product_mi = Vehicle_B.Product1_a * Vehicle_B.Product2_nx;
1060
1061 /* Product: '<S156>/Product' */
1062 Vehicle_B.Product_pb = Vehicle_B.Product_fv / Vehicle_B.sqrt_i;
1063
1064 /* Product: '<S157>/Product1' */
1065 Vehicle_B.Product1_lr = Vehicle_B.Product_pb * Vehicle_B.Product3_ce;
1066
1067 /* Sum: '<S157>/Sum1' */
1068 Vehicle_B.Sum1_a = Vehicle_B.Product_mi + Vehicle_B.Product1_lr;
1069
1070 /* Gain: '<S157>/Gain' */
1071 Vehicle_B.Gain_p = 2.0 * Vehicle_B.Sum1_a;
1072
1073 /* Product: '<S157>/Product4' */
1074 Vehicle_B.Product4_i = Vehicle_B.Gain_p * Vehicle_B.Sum_pb[1];
1075
1076 /* Product: '<S157>/Product2' */
1077 Vehicle_B.Product2_nz = Vehicle_B.Product_pb * Vehicle_B.Product2_nx;
1078
1079 /* Product: '<S157>/Product3' */
1080 Vehicle_B.Product3_lu = Vehicle_B.Product1_a * Vehicle_B.Product3_ce;
1081
1082 /* Sum: '<S157>/Sum2' */
1083 Vehicle_B.Sum2_m = Vehicle_B.Product3_lu - Vehicle_B.Product2_nz;
1084
1085 /* Gain: '<S157>/Gain1' */
1086 Vehicle_B.Gain1_c = 2.0 * Vehicle_B.Sum2_m;
1087
1088 /* Product: '<S157>/Product5' */
1089 Vehicle_B.Product5_jx = Vehicle_B.Gain1_c * Vehicle_B.Sum_pb[2];
1090
1091 /* Sum: '<S157>/Sum' */
1092 Vehicle_B.Sum_g5 = (Vehicle_B.Product8_p + Vehicle_B.Product4_i) +
1093     Vehicle_B.Product5_jx;
1094
1095 /* Product: '<S158>/Product' */
1096 Vehicle_B.Product_pe = Vehicle_B.Product1_a * Vehicle_B.Product2_nx;
1097
1098 /* Product: '<S158>/Product1' */
1099 Vehicle_B.Product1_mu = Vehicle_B.Product_pb * Vehicle_B.Product3_ce;
1100
1101 /* Sum: '<S158>/Sum1' */
1102 Vehicle_B.Sum1_k = Vehicle_B.Product_pe - Vehicle_B.Product1_mu;
1103
1104 /* Gain: '<S158>/Gain' */
1105 Vehicle_B.Gain_ay = 2.0 * Vehicle_B.Sum1_k;
1106
1107 /* Product: '<S158>/Product4' */
1108 Vehicle_B.Product4_ok = Vehicle_B.Sum_pb[0] * Vehicle_B.Gain_ay;
1109
1110 /* Product: '<S158>/Product6' */
1111 Vehicle_B.Product6_ld = Vehicle_B.Product1_a * Vehicle_B.Product1_a;
1112
1113 /* Product: '<S158>/Product7' */
1114 Vehicle_B.Product7_ei = Vehicle_B.Product3_ce * Vehicle_B.Product3_ce;
1115
1116 /* Sum: '<S158>/Sum3' incorporates:
1117  * Constant: '<S158>/Constant'
1118  */
1119 Vehicle_B.Sum3_h = (0.5 - Vehicle_B.Product6_ld) - Vehicle_B.Product7_ei;
1120
1121 /* Gain: '<S158>/Gain2' */
1122 Vehicle_B.Gain2_l2 = 2.0 * Vehicle_B.Sum3_h;
1123
1124 /* Product: '<S158>/Product8' */
1125 Vehicle_B.Product8_d = Vehicle_B.Gain2_l2 * Vehicle_B.Sum_pb[1];
1126
1127 /* Product: '<S158>/Product2' */
1128 Vehicle_B.Product2_ea = Vehicle_B.Product_pb * Vehicle_B.Product1_a;
1129
1130 /* Product: '<S158>/Product3' */
1131 Vehicle_B.Product3_ll = Vehicle_B.Product2_nx * Vehicle_B.Product3_ce;
```

```

1132
1133 /* Sum: '<S158>/Sum2' */
1134 Vehicle_B.Sum2_mr = Vehicle_B.Product2_ea + Vehicle_B.Product3_ll;
1135
1136 /* Gain: '<S158>/Gain1' */
1137 Vehicle_B.Gain1_n = 2.0 * Vehicle_B.Sum2_mr;
1138
1139 /* Product: '<S158>/Product5' */
1140 Vehicle_B.Product5_a = Vehicle_B.Gain1_n * Vehicle_B.Sum_pb[2];
1141
1142 /* Sum: '<S158>/Sum' */
1143 Vehicle_B.Sum_gz = (Vehicle_B.Product4_ok + Vehicle_B.Product8_d) +
1144     Vehicle_B.Product5_a;
1145
1146 /* Product: '<S159>/Product' */
1147 Vehicle_B.Product_am = Vehicle_B.Product1_a * Vehicle_B.Product3_ce;
1148
1149 /* Product: '<S159>/Product1' */
1150 Vehicle_B.Product1_bt = Vehicle_B.Product_pb * Vehicle_B.Product2_nx;
1151
1152 /* Sum: '<S159>/Sum1' */
1153 Vehicle_B.Sum1_hk = Vehicle_B.Product_am + Vehicle_B.Product1_bt;
1154
1155 /* Gain: '<S159>/Gain' */
1156 Vehicle_B.Gain_g = 2.0 * Vehicle_B.Sum1_hk;
1157
1158 /* Product: '<S159>/Product4' */
1159 Vehicle_B.Product4_o5 = Vehicle_B.Sum_pb[0] * Vehicle_B.Gain_g;
1160
1161 /* Product: '<S159>/Product2' */
1162 Vehicle_B.Product2_cc = Vehicle_B.Product_pb * Vehicle_B.Product1_a;
1163
1164 /* Product: '<S159>/Product3' */
1165 Vehicle_B.Product3_cd = Vehicle_B.Product2_nx * Vehicle_B.Product3_ce;
1166
1167 /* Sum: '<S159>/Sum2' */
1168 Vehicle_B.Sum2_o = Vehicle_B.Product3_cd - Vehicle_B.Product2_cc;
1169
1170 /* Gain: '<S159>/Gain1' */
1171 Vehicle_B.Gain1_ni = 2.0 * Vehicle_B.Sum2_o;
1172
1173 /* Product: '<S159>/Product5' */
1174 Vehicle_B.Product5_n2 = Vehicle_B.Gain1_ni * Vehicle_B.Sum_pb[1];
1175
1176 /* Product: '<S159>/Product6' */
1177 Vehicle_B.Product6_d5 = Vehicle_B.Product1_a * Vehicle_B.Product1_a;
1178
1179 /* Product: '<S159>/Product7' */
1180 Vehicle_B.Product7_em = Vehicle_B.Product2_nx * Vehicle_B.Product2_nx;
1181
1182 /* Sum: '<S159>/Sum3' incorporates:
1183  * Constant: '<S159>/Constant'
1184  */
1185 Vehicle_B.Sum3_jh = (0.5 - Vehicle_B.Product6_d5) - Vehicle_B.Product7_em;
1186
1187 /* Gain: '<S159>/Gain2' */
1188 Vehicle_B.Gain2_ja = 2.0 * Vehicle_B.Sum3_jh;
1189
1190 /* Product: '<S159>/Product8' */
1191 Vehicle_B.Product8_j = Vehicle_B.Gain2_ja * Vehicle_B.Sum_pb[2];
1192
1193 /* Sum: '<S159>/Sum' */
1194 Vehicle_B.Sum_pw = (Vehicle_B.Product4_o5 + Vehicle_B.Product5_n2) +
1195     Vehicle_B.Product8_j;
1196
1197 /* SignalConversion generated from: '<S178>/Dot Product2' */
1198 Vehicle_B.TmpSignalConversionAtDotProdu_j[0] = Vehicle_B.Sum_g5;
1199 Vehicle_B.TmpSignalConversionAtDotProdu_j[1] = Vehicle_B.Sum_gz;
1200 Vehicle_B.TmpSignalConversionAtDotProdu_j[2] = Vehicle_B.Sum_pw;
1201
1202 /* DotProduct: '<S178>/Dot Product3' */
1203 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_k[0];
1204 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_j[0];
1205 tmpForInput_idx_2 = tmpForInput_idx_1 * tmpForInput_idx_0;
1206
1207 /* DotProduct: '<S178>/Dot Product1' */
1208 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_k[0];
1209 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_k[0];
1210 tmpForInput = tmpForInput_idx_1 * tmpForInput_idx_0;
1211
1212 /* DotProduct: '<S178>/Dot Product2' */
1213 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_j[0];
1214 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_j[0];
1215 tmpForInput_0 = tmpForInput_idx_1 * tmpForInput_idx_0;
1216
1217 /* DotProduct: '<S178>/Dot Product3' */
1218 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_k[1];
1219 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_j[1];
1220 tmpForInput_idx_2 += tmpForInput_idx_1 * tmpForInput_idx_0;
1221
1222 /* DotProduct: '<S178>/Dot Product1' */
1223 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_k[1];
1224 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_k[1];
1225 tmpForInput += tmpForInput_idx_1 * tmpForInput_idx_0;
1226
1227 /* DotProduct: '<S178>/Dot Product2' */
1228 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_j[1];
1229 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_j[1];
1230 tmpForInput_0 += tmpForInput_idx_1 * tmpForInput_idx_0;
1231
1232 /* DotProduct: '<S178>/Dot Product3' */
1233 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_k[2];
1234 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_j[2];
1235 tmpForInput_idx_2 += tmpForInput_idx_1 * tmpForInput_idx_0;

```

```

1236
1237 /* DotProduct: '<S178>/Dot Product1' */
1238 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_k[2];
1239 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_k[2];
1240 tmpForInput += tmpForInput_idx_1 * tmpForInput_idx_0;
1241
1242 /* DotProduct: '<S178>/Dot Product2' */
1243 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_j[2];
1244 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_j[2];
1245 tmpForInput_0 += tmpForInput_idx_1 * tmpForInput_idx_0;
1246
1247 /* DotProduct: '<S178>/Dot Product3' */
1248 Vehicle_B.DotProduct3_ph = tmpForInput_idx_2;
1249
1250 /* DotProduct: '<S178>/Dot Product1' */
1251 Vehicle_B.DotProduct1_o = tmpForInput;
1252
1253 /* DotProduct: '<S178>/Dot Product2' */
1254 Vehicle_B.DotProduct2_g = tmpForInput_0;
1255
1256 /* Product: '<S178>/Divide1' */
1257 Vehicle_B.Divide1_p = Vehicle_B.DotProduct1_o * Vehicle_B.DotProduct2_g;
1258
1259 /* Sqrt: '<S178>/Sqrt3' */
1260 Vehicle_B.Sqrt3_m = std::sqrt(Vehicle_B.Divide1_p);
1261
1262 /* Product: '<S178>/Divide' */
1263 Vehicle_B.Divide_c = Vehicle_B.DotProduct3_ph / Vehicle_B.Sqrt3_m;
1264
1265 /* Bias: '<S178>/Bias' */
1266 Vehicle_B.Bias_h = Vehicle_B.Divide_c + -1.0;
1267
1268 /* Abs: '<S178>/Abs' */
1269 Vehicle_B.Abs_m = std::abs(Vehicle_B.Bias_h);
1270
1271 /* Bias: '<S178>/Bias1' */
1272 Vehicle_B.Bias1_e = Vehicle_B.Divide_c + 1.0;
1273
1274 /* Abs: '<S178>/Abs1' */
1275 Vehicle_B.Abs1_i = std::abs(Vehicle_B.Bias1_e);
1276
1277 /* If: '<S178>/If' */
1278 if (Vehicle_B.Abs_m < 1.0E-6) {
1279     /* Outputs for IfAction SubSystem: '<S178>/If Action Subsystem' incorporates:
1280      * ActionPort: '<S184>/Action Port'
1281      */
1282     //Vehicle_IfActionSubsystem(&Vehicle_B.Merge_1);
1283     Vehicle_B.Merge_1 = 1.0;
1284     /* End of Outputs for SubSystem: '<S178>/If Action Subsystem' */
1285 } else if (Vehicle_B.Abs1_i < 1.0E-6) {
1286     /* Outputs for IfAction SubSystem: '<S178>/If Action Subsystem1' incorporates:
1287      * ActionPort: '<S185>/Action Port'
1288      */
1289     //Vehicle_IfActionSubsystem1(&Vehicle_B.Merge_1);
1290     Vehicle_B.Merge_1 = -1.0;
1291     /* End of Outputs for SubSystem: '<S178>/If Action Subsystem1' */
1292 } else {
1293     /* Outputs for IfAction SubSystem: '<S178>/If Action Subsystem2' incorporates:
1294      * ActionPort: '<S186>/Action Port'
1295      */
1296     // Vehicle_IfActionSubsystem2(&Vehicle_B.Merge_1);
1297     Vehicle_B.Merge_1 = 0.0;
1298     /* End of Outputs for SubSystem: '<S178>/If Action Subsystem2' */
1299 }
1300
1301 /* End of If: '<S178>/If' */
1302
1303 /* Switch: '<S124>/Switch' */
1304 if (Vehicle_B.OR) {
1305     /* Switch: '<S124>/Switch' */
1306     Vehicle_B.Switch_c[0] = Vehicle_B.Product_fv;
1307     Vehicle_B.Switch_c[1] = Vehicle_B.Product1_n;
1308     Vehicle_B.Switch_c[2] = Vehicle_B.Product2_a;
1309     Vehicle_B.Switch_c[3] = Vehicle_B.Product3_pe;
1310 } else {
1311     /* RelationalOperator: '<S179>/Compare' incorporates:
1312      * Constant: '<S179>/Constant'
1313      */
1314     Vehicle_B.Compare_g = (Vehicle_B.Merge_1 != -1.0);
1315
1316     /* Switch: '<S137>/is 180deg Rot' */
1317     if (Vehicle_B.Compare_g) {
1318         /* Product: '<S183>/j x i' */
1319         Vehicle_B.jxi_no = Vehicle_B.Sum_oa * Vehicle_B.Sum_g5;
1320
1321         /* Product: '<S183>/i x k' */
1322         Vehicle_B.ixk_b = Vehicle_B.Sum_oz * Vehicle_B.Sum_pw;
1323
1324         /* Product: '<S183>/k x j' */
1325         Vehicle_B.kxj_n = Vehicle_B.Sum_nx * Vehicle_B.Sum_gz;
1326
1327         /* Product: '<S182>/i x j' */
1328         Vehicle_B.ixj_l = Vehicle_B.Sum_oz * Vehicle_B.Sum_gz;
1329
1330         /* Product: '<S182>/k x i' */
1331         Vehicle_B.kxi_o = Vehicle_B.Sum_nx * Vehicle_B.Sum_g5;
1332
1333         /* Product: '<S182>/j x k' */
1334         Vehicle_B.jxk_d = Vehicle_B.Sum_oa * Vehicle_B.Sum_pw;
1335
1336         /* Sum: '<S177>/Sum' */
1337         Vehicle_B.Sum_b[0] = Vehicle_B.jxk_d - Vehicle_B.kxj_n;
1338         Vehicle_B.Sum_b[1] = Vehicle_B.kxi_o - Vehicle_B.ixk_b;
1339         Vehicle_B.Sum_b[2] = Vehicle_B.ixj_l - Vehicle_B.jxi_no;

```

```

1340
1341 /* Switch: '<S137>/is 180deg Rot' */
1342 Vehicle_B.is180degRot_l[0] = Vehicle_B.Sum_b[0];
1343 Vehicle_B.is180degRot_l[1] = Vehicle_B.Sum_b[1];
1344 Vehicle_B.is180degRot_l[2] = Vehicle_B.Sum_b[2];
1345 } else {
1346 /* RelationalOperator: '<S180>/x>z' */
1347 Vehicle_B.xz = (Vehicle_B.Sum_oz > Vehicle_B.Sum_nx);
1348
1349 /* Switch: '<S180>/Switch3' */
1350 if (Vehicle_B.xz) {
1351 /* Gain: '<S180>/Gain' */
1352 Vehicle_B.Gain_o2 = -Vehicle_B.Sum_oa;
1353
1354 /* Switch: '<S180>/Switch3' incorporates:
1355 * Constant: '<S180>/Constant'
1356 */
1357 Vehicle_B.Switch3[0] = Vehicle_B.Gain_o2;
1358 Vehicle_B.Switch3[1] = Vehicle_B.Sum_oz;
1359 Vehicle_B.Switch3[2] = 0.0;
1360 } else {
1361 /* Gain: '<S180>/Gain2' */
1362 Vehicle_B.Gain2_c = -Vehicle_B.Sum_nx;
1363
1364 /* Switch: '<S180>/Switch3' incorporates:
1365 * Constant: '<S180>/Constant1'
1366 */
1367 Vehicle_B.Switch3[0] = 0.0;
1368 Vehicle_B.Switch3[1] = Vehicle_B.Gain2_c;
1369 Vehicle_B.Switch3[2] = Vehicle_B.Sum_oa;
1370 }
1371
1372 /* End of Switch: '<S180>/Switch3' */
1373
1374 /* Product: '<S189>/j x i' */
1375 Vehicle_B.jxi_m = Vehicle_B.Sum_oa * Vehicle_B.Switch3[0];
1376
1377 /* Product: '<S189>/i x k' */
1378 Vehicle_B.ixk_k = Vehicle_B.Sum_oz * Vehicle_B.Switch3[2];
1379
1380 /* Product: '<S189>/k x j' */
1381 Vehicle_B.kxj_f = Vehicle_B.Sum_nx * Vehicle_B.Switch3[1];
1382
1383 /* Product: '<S188>/i x j' */
1384 Vehicle_B.ixj_k = Vehicle_B.Sum_oz * Vehicle_B.Switch3[1];
1385
1386 /* Product: '<S188>/k x i' */
1387 Vehicle_B.kxi_bj = Vehicle_B.Sum_nx * Vehicle_B.Switch3[0];
1388
1389 /* Product: '<S188>/j x k' */
1390 Vehicle_B.jxk_l = Vehicle_B.Sum_oa * Vehicle_B.Switch3[2];
1391
1392 /* Sum: '<S187>/Sum' */
1393 Vehicle_B.Sum_pd[0] = Vehicle_B.jxk_l - Vehicle_B.kxj_f;
1394 Vehicle_B.Sum_pd[1] = Vehicle_B.kxi_bj - Vehicle_B.ixk_k;
1395 Vehicle_B.Sum_pd[2] = Vehicle_B.ixj_k - Vehicle_B.jxi_m;
1396
1397 /* Switch: '<S137>/is 180deg Rot' */
1398 Vehicle_B.is180degRot_l[0] = Vehicle_B.Sum_pd[0];
1399 Vehicle_B.is180degRot_l[1] = Vehicle_B.Sum_pd[1];
1400 Vehicle_B.is180degRot_l[2] = Vehicle_B.Sum_pd[2];
1401 }
1402
1403 /* End of Switch: '<S137>/is 180deg Rot' */
1404
1405 /* Product: '<S191>/Product3' */
1406 Vehicle_B.Product3_dz = Vehicle_B.is180degRot_l[2] *
1407 Vehicle_B.is180degRot_l[2];
1408
1409 /* Product: '<S191>/Product2' */
1410 Vehicle_B.Product2_eq = Vehicle_B.is180degRot_l[1] *
1411 Vehicle_B.is180degRot_l[1];
1412
1413 /* Product: '<S191>/Product1' */
1414 Vehicle_B.Product1_aa = Vehicle_B.is180degRot_l[0] *
1415 Vehicle_B.is180degRot_l[0];
1416
1417 /* DotProduct: '<S137>/Dot Product2' */
1418 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_j[0];
1419 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_j[0];
1420 tmpForInput_idx_2 = tmpForInput_idx_1 * tmpForInput_idx_0;
1421
1422 /* DotProduct: '<S137>/Dot Product1' */
1423 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_k[0];
1424 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_k[0];
1425 tmpForInput = tmpForInput_idx_1 * tmpForInput_idx_0;
1426
1427 /* DotProduct: '<S137>/Dot Product3' */
1428 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_k[0];
1429 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_j[0];
1430 tmpForInput_0 = tmpForInput_idx_1 * tmpForInput_idx_0;
1431
1432 /* DotProduct: '<S137>/Dot Product2' */
1433 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_j[1];
1434 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_j[1];
1435 tmpForInput_idx_2 += tmpForInput_idx_1 * tmpForInput_idx_0;
1436
1437 /* DotProduct: '<S137>/Dot Product1' */
1438 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_k[1];
1439 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_k[1];
1440 tmpForInput += tmpForInput_idx_1 * tmpForInput_idx_0;
1441
1442 /* DotProduct: '<S137>/Dot Product3' */
1443 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_k[1];

```

```

1444 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_j[1];
1445 tmpForInput_0 += tmpForInput_idx_1 * tmpForInput_idx_0;
1446
1447 /* DotProduct: '<S137>/Dot Product2' */
1448 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_j[2];
1449 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_j[2];
1450 tmpForInput_idx_2 += tmpForInput_idx_1 * tmpForInput_idx_0;
1451
1452 /* DotProduct: '<S137>/Dot Product1' */
1453 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_k[2];
1454 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_k[2];
1455 tmpForInput += tmpForInput_idx_1 * tmpForInput_idx_0;
1456
1457 /* DotProduct: '<S137>/Dot Product3' */
1458 tmpForInput_idx_1 = Vehicle_B.TmpSignalConversionAtDotProdu_k[2];
1459 tmpForInput_idx_0 = Vehicle_B.TmpSignalConversionAtDotProdu_j[2];
1460 tmpForInput_0 += tmpForInput_idx_1 * tmpForInput_idx_0;
1461
1462 /* DotProduct: '<S137>/Dot Product2' */
1463 Vehicle_B.DotProduct2_n = tmpForInput_idx_2;
1464
1465 /* DotProduct: '<S137>/Dot Product1' */
1466 Vehicle_B.DotProduct1_h = tmpForInput;
1467
1468 /* Product: '<S137>/Divide1' */
1469 Vehicle_B.Divide1_h = Vehicle_B.DotProduct1_h * Vehicle_B.DotProduct2_n;
1470
1471 /* Sqrt: '<S137>/Sqrt3' */
1472 Vehicle_B.Sqrt3_mq = std::sqrt(Vehicle_B.Divide1_h);
1473
1474 /* DotProduct: '<S137>/Dot Product3' */
1475 Vehicle_B.DotProduct3_e = tmpForInput_0;
1476
1477 /* Sum: '<S137>/Add2' */
1478 Vehicle_B.Add2_k = Vehicle_B.DotProduct3_e + Vehicle_B.Sqrt3_mq;
1479
1480 /* Product: '<S191>/Product' */
1481 Vehicle_B.Product_er = Vehicle_B.Add2_k * Vehicle_B.Add2_k;
1482
1483 /* Sum: '<S191>/Sum' */
1484 Vehicle_B.Sum_e = ((Vehicle_B.Product_er + Vehicle_B.Product1_aa) +
1485                   Vehicle_B.Product2_eq) + Vehicle_B.Product3_dz;
1486
1487 /* Sqrt: '<S190>/sqrt' */
1488 Vehicle_B.sqrt_j = std::sqrt(Vehicle_B.Sum_e);
1489
1490 /* Product: '<S181>/Product3' */
1491 Vehicle_B.Product3_lzz = Vehicle_B.is180degRot_l[2] / Vehicle_B.sqrt_j;
1492
1493 /* Product: '<S146>/Product3' */
1494 Vehicle_B.Product3_ay = Vehicle_B.Product3_pe * Vehicle_B.Product3_lzz;
1495
1496 /* Product: '<S181>/Product2' */
1497 Vehicle_B.Product2_ep = Vehicle_B.is180degRot_l[1] / Vehicle_B.sqrt_j;
1498
1499 /* Product: '<S146>/Product2' */
1500 Vehicle_B.Product2_bc = Vehicle_B.Product2_a * Vehicle_B.Product2_ep;
1501
1502 /* Product: '<S181>/Product1' */
1503 Vehicle_B.Product1_ge = Vehicle_B.is180degRot_l[0] / Vehicle_B.sqrt_j;
1504
1505 /* Product: '<S146>/Product1' */
1506 Vehicle_B.Product1_fh = Vehicle_B.Product1_n * Vehicle_B.Product1_ge;
1507
1508 /* Product: '<S181>/Product' */
1509 Vehicle_B.Product_ey = Vehicle_B.Add2_k / Vehicle_B.sqrt_j;
1510
1511 /* Product: '<S146>/Product' */
1512 Vehicle_B.Product_d0 = Vehicle_B.Product_fv * Vehicle_B.Product_ey;
1513
1514 /* Sum: '<S146>/Sum' */
1515 Vehicle_B.Sum_ki = ((Vehicle_B.Product_d0 - Vehicle_B.Product1_fh) -
1516                   Vehicle_B.Product2_bc) - Vehicle_B.Product3_ay;
1517
1518 /* Product: '<S147>/Product3' */
1519 Vehicle_B.Product3_aw = Vehicle_B.Product3_pe * Vehicle_B.Product2_ep;
1520
1521 /* Product: '<S147>/Product2' */
1522 Vehicle_B.Product2_pu = Vehicle_B.Product2_a * Vehicle_B.Product3_lzz;
1523
1524 /* Product: '<S147>/Product1' */
1525 Vehicle_B.Product1_b0 = Vehicle_B.Product1_n * Vehicle_B.Product_ey;
1526
1527 /* Product: '<S147>/Product' */
1528 Vehicle_B.Product_cq = Vehicle_B.Product_fv * Vehicle_B.Product1_ge;
1529
1530 /* Sum: '<S147>/Sum' */
1531 Vehicle_B.Sum_oq = ((Vehicle_B.Product_cq + Vehicle_B.Product1_b0) +
1532                   Vehicle_B.Product2_pu) - Vehicle_B.Product3_aw;
1533
1534 /* Product: '<S148>/Product3' */
1535 Vehicle_B.Product3_h5 = Vehicle_B.Product3_pe * Vehicle_B.Product1_ge;
1536
1537 /* Product: '<S148>/Product2' */
1538 Vehicle_B.Product2_i = Vehicle_B.Product2_a * Vehicle_B.Product_ey;
1539
1540 /* Product: '<S148>/Product1' */
1541 Vehicle_B.Product1_ai = Vehicle_B.Product1_n * Vehicle_B.Product3_lzz;
1542
1543 /* Product: '<S148>/Product' */
1544 Vehicle_B.Product_cf = Vehicle_B.Product_fv * Vehicle_B.Product2_ep;
1545
1546 /* Sum: '<S148>/Sum' */
1547 Vehicle_B.Sum_kc = ((Vehicle_B.Product_cf - Vehicle_B.Product1_ai) +

```



```

1548         Vehicle_B.Product2_i) + Vehicle_B.Product3_h5;
1549
1550     /* Product: '<S149>/Product3' */
1551     Vehicle_B.Product3_c1 = Vehicle_B.Product3_pe * Vehicle_B.Product_ey;
1552
1553     /* Product: '<S149>/Product2' */
1554     Vehicle_B.Product2_no = Vehicle_B.Product2_a * Vehicle_B.Product1_ge;
1555
1556     /* Product: '<S149>/Product1' */
1557     Vehicle_B.Product1_d1 = Vehicle_B.Product1_n * Vehicle_B.Product2_ep;
1558
1559     /* Product: '<S149>/Product' */
1560     Vehicle_B.Product_gr = Vehicle_B.Product_fv * Vehicle_B.Product3_lzz;
1561
1562     /* Sum: '<S149>/Sum' */
1563     Vehicle_B.Sum_n1 = ((Vehicle_B.Product_gr + Vehicle_B.Product1_d1) -
1564         Vehicle_B.Product2_no) + Vehicle_B.Product3_c1;
1565
1566     /* Switch: '<S124>/Switch' */
1567     Vehicle_B.Switch_c[0] = Vehicle_B.Sum_ki;
1568     Vehicle_B.Switch_c[1] = Vehicle_B.Sum_oq;
1569     Vehicle_B.Switch_c[2] = Vehicle_B.Sum_kc;
1570     Vehicle_B.Switch_c[3] = Vehicle_B.Sum_n1;
1571 }
1572
1573 /* End of Switch: '<S124>/Switch' */
1574
1575 /* Product: '<S273>/Product' */
1576 Vehicle_B.Product_h0 = Vehicle_B.Switch_c[0] * Vehicle_B.Switch_c[0];
1577
1578 /* Product: '<S273>/Product1' */
1579 Vehicle_B.Product1_fm = Vehicle_B.Switch_c[1] * Vehicle_B.Switch_c[1];
1580
1581 /* Product: '<S273>/Product2' */
1582 Vehicle_B.Product2_c2 = Vehicle_B.Switch_c[2] * Vehicle_B.Switch_c[2];
1583
1584 /* Product: '<S273>/Product3' */
1585 Vehicle_B.Product3_pq = Vehicle_B.Switch_c[3] * Vehicle_B.Switch_c[3];
1586
1587 /* Sum: '<S273>/Sum' */
1588 Vehicle_B.Sum_f = ((Vehicle_B.Product_h0 + Vehicle_B.Product1_fm) +
1589     Vehicle_B.Product2_c2) + Vehicle_B.Product3_pq;
1590
1591 /* Sqrt: '<S272>/sqrt' */
1592 Vehicle_B.sqrt_mo = std::sqrt(Vehicle_B.Sum_f);
1593
1594 /* Product: '<S267>/Product' */
1595 Vehicle_B.Product_jc = Vehicle_B.Switch_c[0] / Vehicle_B.sqrt_mo;
1596
1597 /* Product: '<S267>/Product1' */
1598 Vehicle_B.Product1_o = Vehicle_B.Switch_c[1] / Vehicle_B.sqrt_mo;
1599
1600 /* Product: '<S267>/Product2' */
1601 Vehicle_B.Product2_j = Vehicle_B.Switch_c[2] / Vehicle_B.sqrt_mo;
1602
1603 /* Product: '<S267>/Product3' */
1604 Vehicle_B.Product3_lz = Vehicle_B.Switch_c[3] / Vehicle_B.sqrt_mo;
1605
1606 /* Fcn: '<S128>/fcn1' */
1607 Vehicle_B.fcn1_i = (Vehicle_B.Product2_j * Vehicle_B.Product3_lz -
1608     Vehicle_B.Product_jc * Vehicle_B.Product1_o) * -2.0;
1609
1610 /* Fcn: '<S128>/fcn2' */
1611 Vehicle_B.fcn2_f = ((Vehicle_B.Product_jc * Vehicle_B.Product_jc -
1612     Vehicle_B.Product1_o * Vehicle_B.Product1_o) -
1613     Vehicle_B.Product2_j * Vehicle_B.Product2_j) +
1614     Vehicle_B.Product3_lz * Vehicle_B.Product3_lz;
1615
1616 /* Trigonometry: '<S266>/Trigonometric Function1' */
1617 Vehicle_B.VectorConcatenate_a[0] = rt_atan2d_snf(Vehicle_B.fcn1_i,
1618     Vehicle_B.fcn2_f);
1619
1620 /* Fcn: '<S128>/fcn3' */
1621 Vehicle_B.fcn3_o = (Vehicle_B.Product1_o * Vehicle_B.Product3_lz +
1622     Vehicle_B.Product_jc * Vehicle_B.Product2_j) * 2.0;
1623
1624 /* If: '<S268>/If' */
1625 if (Vehicle_B.fcn3_o > 1.0) {
1626     /* Outputs for IfAction SubSystem: '<S268>/If Action Subsystem' incorporates:
1627         * ActionPort: '<S269>/Action Port'
1628         */
1629     /* Merge: '<S268>/Merge' incorporates:
1630         * Constant: '<S269>/Constant'
1631         */
1632     Vehicle_B.Merge_k = 1.0;
1633
1634     /* End of Outputs for SubSystem: '<S268>/If Action Subsystem' */
1635 } else if (Vehicle_B.fcn3_o < -1.0) {
1636     /* Outputs for IfAction SubSystem: '<S268>/If Action Subsystem1' incorporates:
1637         * ActionPort: '<S270>/Action Port'
1638         */
1639     /* Merge: '<S268>/Merge' incorporates:
1640         * Constant: '<S270>/Constant'
1641         */
1642     Vehicle_B.Merge_k = 1.0;
1643
1644     /* End of Outputs for SubSystem: '<S268>/If Action Subsystem1' */
1645 } else {
1646     /* Outputs for IfAction SubSystem: '<S268>/If Action Subsystem2' incorporates:
1647         * ActionPort: '<S271>/Action Port'
1648         */
1649     //Vehicle_IfActionSubsystem2_f(Vehicle_B.fcn3_o, &Vehicle_B.Merge_k);
1650     Vehicle_B.Merge_k = Vehicle_B.fcn3_o;
1651     /* End of Outputs for SubSystem: '<S268>/If Action Subsystem2' */

```



```

1652 }
1653
1654 /* End of If: '<S268>/If' */
1655
1656 /* Trigonometry: '<S266>/trigFcn' */
1657 tmpForInput_idx_0 = Vehicle_B.Merge_k;
1658 if (tmpForInput_idx_0 > 1.0) {
1659     tmpForInput_idx_0 = 1.0;
1660 } else {
1661     if (tmpForInput_idx_0 < -1.0) {
1662         tmpForInput_idx_0 = -1.0;
1663     }
1664 }
1665
1666 Vehicle_B.VectorConcatenate_a[1] = std::asin(tmpForInput_idx_0);
1667
1668 /* End of Trigonometry: '<S266>/trigFcn' */
1669
1670 /* Fcn: '<S128>/fcn4' */
1671 Vehicle_B.fcn4_n = (Vehicle_B.Product1_o * Vehicle_B.Product2_j -
1672     Vehicle_B.Product_jc * Vehicle_B.Product3_lz) * -2.0;
1673
1674 /* Fcn: '<S128>/fcn5' */
1675 Vehicle_B.fcn5_g = ((Vehicle_B.Product_jc * Vehicle_B.Product_jc +
1676     Vehicle_B.Product1_o * Vehicle_B.Product1_o) -
1677     Vehicle_B.Product2_j * Vehicle_B.Product2_j) -
1678     Vehicle_B.Product3_lz * Vehicle_B.Product3_lz;
1679
1680 /* Trigonometry: '<S266>/Trigonometric Function3' */
1681 Vehicle_B.VectorConcatenate_a[2] = rt_atan2d_snf(Vehicle_B.fcn4_n,
1682     Vehicle_B.fcn5_g);
1683
1684 /* Gain: '<S229>/Derivative Gain' */
1685 Vehicle_B.DerivativeGain_k[0] = 0.02 * Vehicle_B.VectorConcatenate_a[0];
1686
1687 /* SampleTimeMath: '<S232>/Tsamp'
1688 *
1689 * About '<S232>/Tsamp':
1690 * y = u * K where K = 1 / ( w * Ts )
1691 * Multiplication by K = weightedTsampQuantized is being
1692 * done implicitly by changing the scaling of the input signal.
1693 * No work needs to be done here. Downstream blocks may need
1694 * to do work to handle the scaling of the output; this happens
1695 * automatically.
1696 */
1697 Vehicle_B.Tsamp_f[0] = Vehicle_B.DerivativeGain_k[0];
1698
1699 /* Delay: '<S230>/UD' */
1700 Vehicle_B.UD_l[0] = Vehicle_DW.UD_DSTATE_d[0];
1701
1702 /* Sum: '<S230>/Diff' */
1703 Vehicle_B.Diff_e[0] = Vehicle_B.Tsamp_f[0] - Vehicle_B.UD_l[0];
1704
1705 /* Gain: '<S234>/Integral Gain' */
1706 Vehicle_B.IntegralGain_p[0] = 2.0E-10 * Vehicle_B.VectorConcatenate_a[0];
1707
1708 /* DiscreteIntegrator: '<S237>/Integrator' */
1709 Vehicle_B.Integrator_k[0] = Vehicle_DW.Integrator_DSTATE_f[0];
1710
1711 /* Gain: '<S242>/Proportional Gain' */
1712 Vehicle_B.ProportionalGain_h[0] = 1.000000000000000E-4 *
1713     Vehicle_B.VectorConcatenate_a[0];
1714
1715 /* Merge: '<S1>/Merge' incorporates:
1716 * Sum: '<S246>/Sum'
1717 */
1718 Vehicle_B.Merge[0] = (Vehicle_B.ProportionalGain_h[0] +
1719     Vehicle_B.Integrator_k[0]) + Vehicle_B.Diff_e[0];
1720
1721 /* Update for Delay: '<S230>/UD' */
1722 Vehicle_DW.UD_DSTATE_d[0] = Vehicle_B.Tsamp_f[0];
1723
1724 /* Update for DiscreteIntegrator: '<S237>/Integrator' */
1725 Vehicle_DW.Integrator_DSTATE_f[0] += Vehicle_B.IntegralGain_p[0];
1726
1727 /* Gain: '<S229>/Derivative Gain' */
1728 Vehicle_B.DerivativeGain_k[1] = 0.02 * Vehicle_B.VectorConcatenate_a[1];
1729
1730 /* SampleTimeMath: '<S232>/Tsamp'
1731 *
1732 * About '<S232>/Tsamp':
1733 * y = u * K where K = 1 / ( w * Ts )
1734 * Multiplication by K = weightedTsampQuantized is being
1735 * done implicitly by changing the scaling of the input signal.
1736 * No work needs to be done here. Downstream blocks may need
1737 * to do work to handle the scaling of the output; this happens
1738 * automatically.
1739 */
1740 Vehicle_B.Tsamp_f[1] = Vehicle_B.DerivativeGain_k[1];
1741
1742 /* Delay: '<S230>/UD' */
1743 Vehicle_B.UD_l[1] = Vehicle_DW.UD_DSTATE_d[1];
1744
1745 /* Sum: '<S230>/Diff' */
1746 Vehicle_B.Diff_e[1] = Vehicle_B.Tsamp_f[1] - Vehicle_B.UD_l[1];
1747
1748 /* Gain: '<S234>/Integral Gain' */
1749 Vehicle_B.IntegralGain_p[1] = 2.0E-10 * Vehicle_B.VectorConcatenate_a[1];
1750
1751 /* DiscreteIntegrator: '<S237>/Integrator' */
1752 Vehicle_B.Integrator_k[1] = Vehicle_DW.Integrator_DSTATE_f[1];
1753
1754 /* Gain: '<S242>/Proportional Gain' */
1755 Vehicle_B.ProportionalGain_h[1] = 1.000000000000000E-4 *

```

```

1756         Vehicle_B.VectorConcatenate_a[1];
1757
1758     /* Merge: '<S1>/Merge' incorporates:
1759      * Sum: '<S246>/Sum'
1760      */
1761     Vehicle_B.Merge[1] = (Vehicle_B.ProportionalGain_h[1] +
1762         Vehicle_B.Integrator_k[1]) + Vehicle_B.Diff_e[1];
1763
1764     /* Update for Delay: '<S230>/UD' */
1765     Vehicle_DW.UD_DSTATE_d[1] = Vehicle_B.Tsamp_f[1];
1766
1767     /* Update for DiscreteIntegrator: '<S237>/Integrator' */
1768     Vehicle_DW.Integrator_DSTATE_f[1] += Vehicle_B.IntegralGain_p[1];
1769
1770     /* Gain: '<S229>/Derivative Gain' */
1771     Vehicle_B.DerivativeGain_k[2] = 0.02 * Vehicle_B.VectorConcatenate_a[2];
1772
1773     /* SampleTimeMath: '<S232>/Tsamp'
1774      * About '<S232>/Tsamp':
1775      * y = u * K where K = 1 / ( w * Ts )
1776      * Multiplication by K = weightedTsampQuantized is being
1777      * done implicitly by changing the scaling of the input signal.
1778      * No work needs to be done here. Downstream blocks may need
1779      * to do work to handle the scaling of the output; this happens
1780      * automatically.
1781      */
1782     Vehicle_B.Tsamp_f[2] = Vehicle_B.DerivativeGain_k[2];
1783
1784     /* Delay: '<S230>/UD' */
1785     Vehicle_B.UD_l[2] = Vehicle_DW.UD_DSTATE_d[2];
1786
1787     /* Sum: '<S230>/Diff' */
1788     Vehicle_B.Diff_e[2] = Vehicle_B.Tsamp_f[2] - Vehicle_B.UD_l[2];
1789
1790     /* Gain: '<S234>/Integral Gain' */
1791     Vehicle_B.IntegralGain_p[2] = 2.0E-10 * Vehicle_B.VectorConcatenate_a[2];
1792
1793     /* DiscreteIntegrator: '<S237>/Integrator' */
1794     Vehicle_B.Integrator_k[2] = Vehicle_DW.Integrator_DSTATE_f[2];
1795
1796     /* Gain: '<S242>/Proportional Gain' */
1797     Vehicle_B.ProportionalGain_h[2] = 1.000000000000000E-4 *
1798         Vehicle_B.VectorConcatenate_a[2];
1799
1800     /* Merge: '<S1>/Merge' incorporates:
1801      * Sum: '<S246>/Sum'
1802      */
1803     Vehicle_B.Merge[2] = (Vehicle_B.ProportionalGain_h[2] +
1804         Vehicle_B.Integrator_k[2]) + Vehicle_B.Diff_e[2];
1805
1806     /* Update for Delay: '<S230>/UD' */
1807     Vehicle_DW.UD_DSTATE_d[2] = Vehicle_B.Tsamp_f[2];
1808
1809     /* Update for DiscreteIntegrator: '<S237>/Integrator' */
1810     Vehicle_DW.Integrator_DSTATE_f[2] += Vehicle_B.IntegralGain_p[2];
1811
1812     /* End of Outputs for SubSystem: '<S1>/SunPointing Mode' */
1813 }
1814

```

Vehicle.h

```

1  /*
2  * Vehicle.h
3  *
4  * Academic License – for use in teaching, academic research, and meeting
5  * course requirements at degree granting institutions only. Not for
6  * government, commercial, or other organizational use.
7  *
8  * Code generation for model "Vehicle".
9  *
10 * Model version          : 2.125
11 * Simulink Coder version : 9.4 (R2020b) 29-Jul-2020
12 * C++ source code generated on : Mon Nov  2 12:16:23 2020
13 *
14 * Target selection: grt.tlc
15 * Note: GRT includes extra infrastructure and instrumentation for prototyping
16 * Embedded hardware selection: 32-bit Generic
17 * Code generation objective: Debugging
18 * Validation result: Not run
19 */
20
21 #ifndef RTW_HEADER_Vehicle_h_
22 #define RTW_HEADER_Vehicle_h_
23 #include <cmath>
24 #include <cstring>
25 #include <stdlib.h>
26 #include "TripleData.h"
27 #include "SingleData.h"
28
29
30
31 typedef void * pointer_T;
32
33 /* My types */
34 typedef double real_T;
35 typedef unsigned int uint32_T;
36 typedef int int32_T;
37 typedef unsigned char boolean_T;
38
39
40 typedef char char_T;

```

```

41|typedef float real32_T;
42|typedef unsigned int uint16_T;
43|typedef unsigned char uint8_T;
44|
45|
46|
47|#define RAD_HARD
48|
49|/* Block signals (default storage) */
50|typedef struct {
51|#ifdef RAD_HARD
52|    HardenedData::TripleData<real_T>    norm_b;
53|
54|    HardenedData::TripleData<real_T>    Merge[3];
55|    HardenedData::TripleData<real_T>    jxk;
56|    HardenedData::TripleData<real_T>    kxi;
57|    HardenedData::TripleData<real_T>    ixj;
58|    HardenedData::TripleData<real_T>    kxj;
59|    HardenedData::TripleData<real_T>    ixk;
60|    HardenedData::TripleData<real_T>    jxi;
61|    HardenedData::TripleData<real_T>    Sum[3];
62|    HardenedData::TripleData<real_T>    zr[3];
63|    HardenedData::TripleData<real_T>    ixj_f;
64|    HardenedData::TripleData<real_T>    jxk_o;
65|    HardenedData::TripleData<real_T>    kxi_e;
66|    HardenedData::TripleData<real_T>    ixk_g;
67|    HardenedData::TripleData<real_T>    jxi_l;
68|    HardenedData::TripleData<real_T>    kxj_a;
69|    HardenedData::TripleData<real_T>    Sum_1[3];
70|    HardenedData::TripleData<real_T>    DotProduct;
71|    HardenedData::SingleData<real_T>    SumofElements;
72|    HardenedData::SingleData<real_T>    MathFunction1;
73|    HardenedData::SingleData<real_T>    DotProduct_p;
74|    HardenedData::SingleData<real_T>    SumofElements_c;
75|    HardenedData::SingleData<real_T>    MathFunction1_b;
76|    HardenedData::SingleData<real_T>    DotProduct_b;
77|    HardenedData::SingleData<real_T>    SumofElements_l;
78|    HardenedData::SingleData<real_T>    MathFunction1_m;
79|    HardenedData::SingleData<real_T>    ORFtoECEF[9];
80|    HardenedData::SingleData<real_T>    ECEFToORF[9];
81|    HardenedData::TripleData<real_T>    Add;
82|    HardenedData::TripleData<real_T>    Merge_i[4];
83|    HardenedData::TripleData<real_T>    Product;
84|    HardenedData::TripleData<real_T>    Product1;
85|    HardenedData::TripleData<real_T>    Product2;
86|    HardenedData::TripleData<real_T>    Product3;
87|    HardenedData::TripleData<real_T>    Sum_i;
88|    HardenedData::TripleData<real_T>    Divide;
89|    HardenedData::TripleData<real_T>    Product_j;
90|    HardenedData::TripleData<real_T>    UnaryMinus;
91|    HardenedData::TripleData<real_T>    Divide1;
92|    HardenedData::TripleData<real_T>    Product1_h;
93|    HardenedData::TripleData<real_T>    UnaryMinus1;
94|    HardenedData::TripleData<real_T>    Divide2;
95|    HardenedData::TripleData<real_T>    Product2_d;
96|    HardenedData::TripleData<real_T>    UnaryMinus2;
97|    HardenedData::TripleData<real_T>    Divide3;
98|    HardenedData::TripleData<real_T>    Product3_c;
99|    HardenedData::TripleData<real_T>    Sum_p;
100|    HardenedData::TripleData<real_T>    Product_b;
101|    HardenedData::TripleData<real_T>    Product_p;
102|    HardenedData::TripleData<real_T>    Product1_i;
103|    HardenedData::TripleData<real_T>    Product2_o;
104|    HardenedData::TripleData<real_T>    Product3_p;
105|    HardenedData::TripleData<real_T>    Sum_iw;
106|    HardenedData::TripleData<real_T>    Product1_in;
107|    HardenedData::TripleData<real_T>    Product_by;
108|    HardenedData::TripleData<real_T>    Product1_f;
109|    HardenedData::TripleData<real_T>    Product2_m;
110|    HardenedData::TripleData<real_T>    Product3_pa;
111|    HardenedData::TripleData<real_T>    Sum_d;
112|    HardenedData::TripleData<real_T>    Product2_n;
113|    HardenedData::TripleData<real_T>    Product_pd;
114|    HardenedData::TripleData<real_T>    Product1_d;
115|    HardenedData::TripleData<real_T>    Product2_nn;
116|    HardenedData::TripleData<real_T>    Product3_l;
117|    HardenedData::TripleData<real_T>    Sum_o;
118|    HardenedData::TripleData<real_T>    Product3_ph;
119|    HardenedData::TripleData<real_T>    Sum_g;
120|    HardenedData::TripleData<real_T>    sqrt_g;
121|    HardenedData::TripleData<real_T>    Product_h;
122|    HardenedData::TripleData<real_T>    Product1_j;
123|    HardenedData::TripleData<real_T>    Product2_dj;
124|    HardenedData::TripleData<real_T>    Product3_e;
125|    HardenedData::TripleData<real_T>    fcn1;
126|    HardenedData::TripleData<real_T>    fcn2;
127|    HardenedData::TripleData<real_T>    fcn3;
128|    HardenedData::TripleData<real_T>    Merge_il;
129|    HardenedData::TripleData<real_T>    fcn4;
130|    HardenedData::TripleData<real_T>    fcn5;
131|    HardenedData::TripleData<real_T>    VectorConcatenate[3];
132|    HardenedData::TripleData<real_T>    DerivativeGain[3];
133|    HardenedData::TripleData<real_T>    Tsamp[3];
134|    HardenedData::TripleData<real_T>    UD[3];
135|    HardenedData::TripleData<real_T>    Diff[3];
136|    HardenedData::TripleData<real_T>    IntegralGain[3];
137|    HardenedData::TripleData<real_T>    Integrator[3];
138|    HardenedData::TripleData<real_T>    ProportionalGain[3];
139|    HardenedData::TripleData<real_T>    Product_a;
140|    HardenedData::TripleData<real_T>    Product1_k;
141|    HardenedData::TripleData<real_T>    Product2_e;
142|    HardenedData::TripleData<real_T>    Product3_g;
143|    HardenedData::TripleData<real_T>    Product4;
144|    HardenedData::TripleData<real_T>    Product5;
145|
146|    /* '<S1>/Merge' */
147|    /* '<S20>/j x k' */
148|    /* '<S20>/k x i' */
149|    /* '<S20>/i x j' */
150|    /* '<S21>/k x j' */
151|    /* '<S21>/i x k' */
152|    /* '<S21>/j x i' */
153|    /* '<S13>/Sum' */
154|    /* '<S7>/z = -r' */
155|    /* '<S18>/i x j' */
156|    /* '<S18>/j x k' */
157|    /* '<S18>/k x i' */
158|    /* '<S19>/i x k' */
159|    /* '<S19>/j x i' */
160|    /* '<S19>/k x j' */
161|    /* '<S12>/Sum' */
162|    /* '<S15>/Dot Product' */
163|    /* '<S15>/Sum of Elements' */
164|    /* '<S15>/Math Function1' */
165|    /* '<S16>/Dot Product' */
166|    /* '<S16>/Sum of Elements' */
167|    /* '<S16>/Math Function1' */
168|    /* '<S17>/Dot Product' */
169|    /* '<S17>/Sum of Elements' */
170|    /* '<S17>/Math Function1' */
171|    /* '<S7>/ORF to ECEF' */
172|    /* '<S7>/ECEFTo ORF' */
173|    /* '<S25>/Add' */
174|    /* '<S14>/Merge' */
175|    /* '<S111>/Product' */
176|    /* '<S111>/Product1' */
177|    /* '<S111>/Product2' */
178|    /* '<S111>/Product3' */
179|    /* '<S111>/Sum' */
180|    /* '<S9>/Divide' */
181|    /* '<S112>/Product' */
182|    /* '<S110>/Unary Minus' */
183|    /* '<S9>/Divide1' */
184|    /* '<S112>/Product1' */
185|    /* '<S110>/Unary Minus1' */
186|    /* '<S9>/Divide2' */
187|    /* '<S112>/Product2' */
188|    /* '<S110>/Unary Minus2' */
189|    /* '<S9>/Divide3' */
190|    /* '<S112>/Product3' */
191|    /* '<S112>/Sum' */
192|    /* '<S123>/Product' */
193|    /* '<S113>/Product' */
194|    /* '<S113>/Product1' */
195|    /* '<S113>/Product2' */
196|    /* '<S113>/Product3' */
197|    /* '<S113>/Sum' */
198|    /* '<S123>/Product1' */
199|    /* '<S114>/Product' */
200|    /* '<S114>/Product1' */
201|    /* '<S114>/Product2' */
202|    /* '<S114>/Product3' */
203|    /* '<S114>/Sum' */
204|    /* '<S123>/Product2' */
205|    /* '<S115>/Product' */
206|    /* '<S115>/Product1' */
207|    /* '<S115>/Product2' */
208|    /* '<S115>/Product3' */
209|    /* '<S115>/Sum' */
210|    /* '<S123>/Product3' */
211|    /* '<S123>/Sum' */
212|    /* '<S122>/sqrt' */
213|    /* '<S117>/Product' */
214|    /* '<S117>/Product1' */
215|    /* '<S117>/Product2' */
216|    /* '<S117>/Product3' */
217|    /* '<S11>/fcn1' */
218|    /* '<S11>/fcn2' */
219|    /* '<S11>/fcn3' */
220|    /* '<S118>/Merge' */
221|    /* '<S11>/fcn4' */
222|    /* '<S11>/fcn5' */
223|    /* '<S116>/Vector Concatenate' */
224|    /* '<S85>/Derivative Gain' */
225|    /* '<S88>/Tsamp' */
226|    /* '<S86>/UD' */
227|    /* '<S86>/Diff' */
228|    /* '<S90>/Integral Gain' */
229|    /* '<S93>/Integrator' */
230|    /* '<S98>/Proportional Gain' */
231|    /* '<S58>/Product' */
232|    /* '<S58>/Product1' */
233|    /* '<S58>/Product2' */
234|    /* '<S58>/Product3' */
235|    /* '<S58>/Product4' */
236|    /* '<S58>/Product5' */

```

```

145 HardenedData :: TripleData<real_T> detMatrix; /* '<S58>/Sum' */
146 HardenedData :: TripleData<real_T> Bias; /* '<S52>/Bias' */
147 HardenedData :: TripleData<real_T> Abs1; /* '<S52>/Abs1' */
148 HardenedData :: TripleData<real_T> MathFunction [9]; /* '<S51>/Math Function' */
149 HardenedData :: TripleData<real_T> Product_c [9]; /* '<S51>/Product' */
150 HardenedData :: TripleData<real_T> Bias1 [9]; /* '<S51>/Bias1' */
151 HardenedData :: TripleData<real_T> Abs2 [9]; /* '<S51>/Abs2' */
152 HardenedData :: TripleData<real_T> Add_m; /* '<S34>/Add' */
153 HardenedData :: TripleData<real_T> sqrt_m; /* '<S26>/sqrt' */
154 HardenedData :: TripleData<real_T> Switch [2]; /* '<S33>/Switch' */
155 HardenedData :: TripleData<real_T> Product_k; /* '<S33>/Product' */
156 HardenedData :: TripleData<real_T> Add_o; /* '<S32>/Add' */
157 HardenedData :: TripleData<real_T> Add_ov; /* '<S30>/Add' */
158 HardenedData :: TripleData<real_T> Add_l; /* '<S31>/Add' */
159 HardenedData :: TripleData<real_T> Product_cy [3]; /* '<S26>/Product' */
160 HardenedData :: TripleData<real_T> Add_f; /* '<S44>/Add' */
161 HardenedData :: TripleData<real_T> sqrt_l; /* '<S28>/sqrt' */
162 HardenedData :: TripleData<real_T> Add_e; /* '<S40>/Add' */
163 HardenedData :: TripleData<real_T> Add_l2; /* '<S41>/Add' */
164 HardenedData :: TripleData<real_T> Add_ek; /* '<S42>/Add' */
165 HardenedData :: TripleData<real_T> Switch_b [2]; /* '<S43>/Switch' */
166 HardenedData :: TripleData<real_T> Product_d; /* '<S43>/Product' */
167 HardenedData :: TripleData<real_T> Product_i [3]; /* '<S28>/Product' */
168 HardenedData :: TripleData<real_T> Add_g; /* '<S39>/Add' */
169 HardenedData :: TripleData<real_T> sqrt_mw; /* '<S27>/sqrt' */
170 HardenedData :: TripleData<real_T> Add_gw; /* '<S37>/Add' */
171 HardenedData :: TripleData<real_T> Add_fc; /* '<S36>/Add' */
172 HardenedData :: TripleData<real_T> Add_d; /* '<S35>/Add' */
173 HardenedData :: TripleData<real_T> Switch_f [2]; /* '<S38>/Switch' */
174 HardenedData :: TripleData<real_T> Product_e; /* '<S38>/Product' */
175 HardenedData :: TripleData<real_T> Product_o [3]; /* '<S27>/Product' */
176 HardenedData :: TripleData<real_T> Sum_gh; /* '<S23>/Sum' */
177 HardenedData :: TripleData<real_T> sqrt_a; /* '<S23>/sqrt' */
178 HardenedData :: TripleData<real_T> Gain1; /* '<S23>/Gain1' */
179 HardenedData :: TripleData<real_T> Add_k; /* '<S46>/Add' */
180 HardenedData :: TripleData<real_T> Add_el; /* '<S45>/Add' */
181 HardenedData :: TripleData<real_T> Add_a; /* '<S47>/Add' */
182 HardenedData :: TripleData<real_T> Merge_m; /* '<S129>/Merge' */
183 HardenedData :: TripleData<real_T> Product_n; /* '<S259>/Product' */
184 HardenedData :: TripleData<real_T> Product1_ii; /* '<S259>/Product1' */
185 HardenedData :: TripleData<real_T> Product2_c; /* '<S259>/Product2' */
186 HardenedData :: TripleData<real_T> Product3_d; /* '<S259>/Product3' */
187 HardenedData :: TripleData<real_T> Sum_j; /* '<S259>/Sum' */
188 HardenedData :: TripleData<real_T> sqrt_n; /* '<S258>/sqrt' */
189 HardenedData :: TripleData<real_T> Product2_oi; /* '<S254>/Product2' */
190 HardenedData :: TripleData<real_T> Product6; /* '<S255>/Product6' */
191 HardenedData :: TripleData<real_T> Product3_a; /* '<S254>/Product3' */
192 HardenedData :: TripleData<real_T> Product7; /* '<S255>/Product7' */
193 HardenedData :: TripleData<real_T> Sum3; /* '<S255>/Sum3' */
194 HardenedData :: TripleData<real_T> Gain2; /* '<S255>/Gain2' */
195 HardenedData :: TripleData<real_T> Product8; /* '<S255>/Product8' */
196 HardenedData :: TripleData<real_T> Product1_fv; /* '<S254>/Product1' */
197 HardenedData :: TripleData<real_T> Product_jb; /* '<S255>/Product' */
198 HardenedData :: TripleData<real_T> Product_cw; /* '<S254>/Product' */
199 HardenedData :: TripleData<real_T> Product1_b; /* '<S255>/Product1' */
200 HardenedData :: TripleData<real_T> Sum1; /* '<S255>/Sum1' */
201 HardenedData :: TripleData<real_T> Gain; /* '<S255>/Gain' */
202 HardenedData :: TripleData<real_T> Product4_p; /* '<S255>/Product4' */
203 HardenedData :: TripleData<real_T> Product2_ox; /* '<S255>/Product2' */
204 HardenedData :: TripleData<real_T> Product3_pv; /* '<S255>/Product3' */
205 HardenedData :: TripleData<real_T> Sum2; /* '<S255>/Sum2' */
206 HardenedData :: TripleData<real_T> Gain1_l; /* '<S255>/Gain1' */
207 HardenedData :: TripleData<real_T> Product5_g; /* '<S255>/Product5' */
208 HardenedData :: TripleData<real_T> Sum_k; /* '<S255>/Sum' */
209 HardenedData :: TripleData<real_T> Product_if; /* '<S256>/Product' */
210 HardenedData :: TripleData<real_T> Product1_e; /* '<S256>/Product1' */
211 HardenedData :: TripleData<real_T> Sum1_b; /* '<S256>/Sum1' */
212 HardenedData :: TripleData<real_T> Gain_a; /* '<S256>/Gain' */
213 HardenedData :: TripleData<real_T> Product4_k; /* '<S256>/Product4' */
214 HardenedData :: TripleData<real_T> Product6_d; /* '<S256>/Product6' */
215 HardenedData :: TripleData<real_T> Product7_e; /* '<S256>/Product7' */
216 HardenedData :: TripleData<real_T> Sum3_p; /* '<S256>/Sum3' */
217 HardenedData :: TripleData<real_T> Gain2_a; /* '<S256>/Gain2' */
218 HardenedData :: TripleData<real_T> Product8_c; /* '<S256>/Product8' */
219 HardenedData :: TripleData<real_T> Product2_l; /* '<S256>/Product2' */
220 HardenedData :: TripleData<real_T> Product3_c5; /* '<S256>/Product3' */
221 HardenedData :: TripleData<real_T> Sum2_k; /* '<S256>/Sum2' */
222 HardenedData :: TripleData<real_T> Gain1_d; /* '<S256>/Gain1' */
223 HardenedData :: TripleData<real_T> Product5_j; /* '<S256>/Product5' */
224 HardenedData :: TripleData<real_T> Sum_lm; /* '<S256>/Sum' */
225 HardenedData :: TripleData<real_T> Product_c3; /* '<S257>/Product' */
226 HardenedData :: TripleData<real_T> Product1_c; /* '<S257>/Product1' */
227 HardenedData :: TripleData<real_T> Sum1_bu; /* '<S257>/Sum1' */
228 HardenedData :: TripleData<real_T> Gain_k; /* '<S257>/Gain' */
229 HardenedData :: TripleData<real_T> Product4_m; /* '<S257>/Product4' */
230 HardenedData :: TripleData<real_T> Product2_k; /* '<S257>/Product2' */
231 HardenedData :: TripleData<real_T> Product3_f; /* '<S257>/Product3' */
232 HardenedData :: TripleData<real_T> Sum2_p; /* '<S257>/Sum2' */
233 HardenedData :: TripleData<real_T> Gain1_i; /* '<S257>/Gain1' */
234 HardenedData :: TripleData<real_T> Product5_d; /* '<S257>/Product5' */
235 HardenedData :: TripleData<real_T> Product6_b; /* '<S257>/Product6' */
236 HardenedData :: TripleData<real_T> Product7_h; /* '<S257>/Product7' */
237 HardenedData :: TripleData<real_T> Sum3_e; /* '<S257>/Sum3' */
238 HardenedData :: TripleData<real_T> Gain2_b; /* '<S257>/Gain2' */
239 HardenedData :: TripleData<real_T> Product8_g; /* '<S257>/Product8' */
240 HardenedData :: TripleData<real_T> Sum_n; /* '<S257>/Sum' */
241 HardenedData :: TripleData<real_T> TmpSignalConversionAtDotProduct [3]; /* '<S265>/Product' */
242 HardenedData :: TripleData<real_T> Product_jz; /* '<S265>/Product1' */
243 HardenedData :: TripleData<real_T> Product1_bp; /* '<S265>/Product2' */
244 HardenedData :: TripleData<real_T> Product2_b; /* '<S265>/Product3' */
245 HardenedData :: TripleData<real_T> Product3_dh; /* '<S265>/Sum' */
246 HardenedData :: TripleData<real_T> Sum_d5; /* '<S264>/sqrt' */
247 HardenedData :: TripleData<real_T> sqrt_k; /* '<S260>/Product2' */
248 HardenedData :: TripleData<real_T> Product2_lq;

```



```

249 HardenedData :: TripleData<real_T> Product6_l; /* '<S261>/Product6' */
250 HardenedData :: TripleData<real_T> Product3_h; /* '<S260>/Product3' */
251 HardenedData :: TripleData<real_T> Product7_d; /* '<S261>/Product7' */
252 HardenedData :: TripleData<real_T> Sum3_j; /* '<S261>/Sum3' */
253 HardenedData :: TripleData<real_T> Gain2_f; /* '<S261>/Gain2' */
254 HardenedData :: TripleData<real_T> Product8_m; /* '<S261>/Product8' */
255 HardenedData :: TripleData<real_T> Product1_b1; /* '<S260>/Product1' */
256 HardenedData :: TripleData<real_T> Product_h1; /* '<S261>/Product' */
257 HardenedData :: TripleData<real_T> Product_f; /* '<S260>/Product' */
258 HardenedData :: TripleData<real_T> Product1_l; /* '<S261>/Product1' */
259 HardenedData :: TripleData<real_T> Sum1_d; /* '<S261>/Sum1' */
260 HardenedData :: TripleData<real_T> Gain_j; /* '<S261>/Gain' */
261 HardenedData :: TripleData<real_T> Product4_o; /* '<S261>/Product4' */
262 HardenedData :: TripleData<real_T> Product2_p; /* '<S261>/Product2' */
263 HardenedData :: TripleData<real_T> Product3_lx; /* '<S261>/Product3' */
264 HardenedData :: TripleData<real_T> Sum2_p3; /* '<S261>/Sum2' */
265 HardenedData :: TripleData<real_T> Gain1_o; /* '<S261>/Gain1' */
266 HardenedData :: TripleData<real_T> Product5_i; /* '<S261>/Product5' */
267 HardenedData :: TripleData<real_T> Sum_kk; /* '<S261>/Sum' */
268 HardenedData :: TripleData<real_T> Product_bg; /* '<S262>/Product' */
269 HardenedData :: TripleData<real_T> Product1_l3; /* '<S262>/Product1' */
270 HardenedData :: TripleData<real_T> Sum1_l; /* '<S262>/Sum1' */
271 HardenedData :: TripleData<real_T> Gain_i; /* '<S262>/Gain' */
272 HardenedData :: TripleData<real_T> Product4_pt; /* '<S262>/Product4' */
273 HardenedData :: TripleData<real_T> Product6_p; /* '<S262>/Product6' */
274 HardenedData :: TripleData<real_T> Product7_dj; /* '<S262>/Product7' */
275 HardenedData :: TripleData<real_T> Sum3_o; /* '<S262>/Sum3' */
276 HardenedData :: TripleData<real_T> Gain2_j; /* '<S262>/Gain2' */
277 HardenedData :: TripleData<real_T> Product8_e; /* '<S262>/Product8' */
278 HardenedData :: TripleData<real_T> Product2_mf; /* '<S262>/Product2' */
279 HardenedData :: TripleData<real_T> Product3_k; /* '<S262>/Product3' */
280 HardenedData :: TripleData<real_T> Sum2_a; /* '<S262>/Sum2' */
281 HardenedData :: TripleData<real_T> Gain1_oi; /* '<S262>/Gain1' */
282 HardenedData :: TripleData<real_T> Product5_b; /* '<S262>/Product5' */
283 HardenedData :: TripleData<real_T> Sum_l; /* '<S262>/Sum' */
284 HardenedData :: TripleData<real_T> Product_l; /* '<S263>/Product' */
285 HardenedData :: TripleData<real_T> Product1_id; /* '<S263>/Product1' */
286 HardenedData :: TripleData<real_T> Sum1_d1; /* '<S263>/Sum1' */
287 HardenedData :: TripleData<real_T> Gain_o; /* '<S263>/Gain' */
288 HardenedData :: TripleData<real_T> Product4_d; /* '<S263>/Product4' */
289 HardenedData :: TripleData<real_T> Product2_f; /* '<S263>/Product2' */
290 HardenedData :: TripleData<real_T> Product3_gx; /* '<S263>/Product3' */
291 HardenedData :: TripleData<real_T> Sum2_l; /* '<S263>/Sum2' */
292 HardenedData :: TripleData<real_T> Gain1_g; /* '<S263>/Gain1' */
293 HardenedData :: TripleData<real_T> Product5_n; /* '<S263>/Product5' */
294 HardenedData :: TripleData<real_T> Product6_j; /* '<S263>/Product6' */
295 HardenedData :: TripleData<real_T> Product7_g; /* '<S263>/Product7' */
296 HardenedData :: TripleData<real_T> Sum3_b; /* '<S263>/Sum3' */
297 HardenedData :: TripleData<real_T> Gain2_o; /* '<S263>/Gain2' */
298 HardenedData :: TripleData<real_T> Product8_a; /* '<S263>/Product8' */
299 HardenedData :: TripleData<real_T> Sum_kb; /* '<S263>/Sum' */
300 HardenedData :: TripleData<real_T> TmpSignalConversionAtDotProdu_d[3];
301 HardenedData :: TripleData<real_T> DotProduct3; /* '<S130>/Dot Product3' */
302 HardenedData :: TripleData<real_T> DotProduct1; /* '<S130>/Dot Product1' */
303 HardenedData :: TripleData<real_T> DotProduct2; /* '<S130>/Dot Product2' */
304 HardenedData :: TripleData<real_T> Divide1_k; /* '<S130>/Divide1' */
305 HardenedData :: TripleData<real_T> Sqrt3; /* '<S130>/Sqrt3' */
306 HardenedData :: TripleData<real_T> Divide_o; /* '<S130>/Divide' */
307 HardenedData :: TripleData<real_T> Bias_k; /* '<S130>/Bias' */
308 HardenedData :: TripleData<real_T> Abs; /* '<S130>/Abs' */
309 HardenedData :: TripleData<real_T> Bias1_j; /* '<S130>/Bias1' */
310 HardenedData :: TripleData<real_T> Abs1_d; /* '<S130>/Abs1' */
311 HardenedData :: TripleData<real_T> Merge_ip; /* '<S130>/Merge' */
312 HardenedData :: TripleData<real_T> DotProduct3_o; /* '<S136>/Dot Product3' */
313 HardenedData :: TripleData<real_T> DotProduct2_c; /* '<S136>/Dot Product2' */
314 HardenedData :: TripleData<real_T> Divide1_d; /* '<S136>/Divide1' */
315 HardenedData :: TripleData<real_T> Sqrt3_j; /* '<S136>/Sqrt3' */
316 HardenedData :: TripleData<real_T> Add2; /* '<S136>/Add2' */
317 HardenedData :: TripleData<real_T> Product_i2; /* '<S176>/Product' */
318 HardenedData :: TripleData<real_T> DotProduct3_p; /* '<S163>/Dot Product3' */
319 HardenedData :: TripleData<real_T> DotProduct2_f; /* '<S163>/Dot Product2' */
320 HardenedData :: TripleData<real_T> Divide1_b; /* '<S163>/Divide1' */
321 HardenedData :: TripleData<real_T> Sqrt3_l; /* '<S163>/Sqrt3' */
322 HardenedData :: TripleData<real_T> Divide_k; /* '<S163>/Divide' */
323 HardenedData :: TripleData<real_T> Bias_b; /* '<S163>/Bias' */
324 HardenedData :: TripleData<real_T> Abs_o; /* '<S163>/Abs' */
325 HardenedData :: TripleData<real_T> Bias1_m; /* '<S163>/Bias1' */
326 HardenedData :: TripleData<real_T> Abs1_l; /* '<S163>/Abs1' */
327 HardenedData :: TripleData<real_T> Merge_d; /* '<S163>/Merge' */
328 HardenedData :: TripleData<real_T> is180degRot[3]; /* '<S136>/is 180deg Rot' */
329 HardenedData :: TripleData<real_T> Product1_g; /* '<S176>/Product1' */
330 HardenedData :: TripleData<real_T> Product2_g; /* '<S176>/Product2' */
331 HardenedData :: TripleData<real_T> Product3_aj; /* '<S176>/Product3' */
332 HardenedData :: TripleData<real_T> Sum_c; /* '<S176>/Sum' */
333 HardenedData :: TripleData<real_T> sqrt_h; /* '<S175>/sqrt' */
334 HardenedData :: TripleData<real_T> Product_fv; /* '<S166>/Product' */
335 HardenedData :: TripleData<real_T> Product2_a; /* '<S166>/Product2' */
336 HardenedData :: TripleData<real_T> Product_l5; /* '<S155>/Product' */
337 HardenedData :: TripleData<real_T> Product1_n; /* '<S166>/Product1' */
338 HardenedData :: TripleData<real_T> Product1_lg; /* '<S155>/Product1' */
339 HardenedData :: TripleData<real_T> Product2_fh; /* '<S155>/Product2' */
340 HardenedData :: TripleData<real_T> Product3_pe; /* '<S166>/Product3' */
341 HardenedData :: TripleData<real_T> Product3_l0; /* '<S155>/Product3' */
342 HardenedData :: TripleData<real_T> Sum_n3; /* '<S155>/Sum' */
343 HardenedData :: TripleData<real_T> sqrt_mv; /* '<S154>/sqrt' */
344 HardenedData :: TripleData<real_T> Product2_ke; /* '<S150>/Product2' */
345 HardenedData :: TripleData<real_T> Product6_e; /* '<S151>/Product6' */
346 HardenedData :: TripleData<real_T> Product3_cc; /* '<S150>/Product3' */
347 HardenedData :: TripleData<real_T> Product7_a; /* '<S151>/Product7' */
348 HardenedData :: TripleData<real_T> Sum3_b5; /* '<S151>/Sum3' */
349 HardenedData :: TripleData<real_T> Gain2_p; /* '<S151>/Gain2' */
350 HardenedData :: TripleData<real_T> Product8_k; /* '<S151>/Product8' */
351 HardenedData :: TripleData<real_T> Product1_l0; /* '<S150>/Product1' */
352 HardenedData :: TripleData<real_T> Product_da; /* '<S151>/Product' */

```

```

353 HardenedData::TripleData<real_T> Product_m; /* '<S150>/Product' */
354 HardenedData::TripleData<real_T> Product1_dj; /* '<S151>/Product1' */
355 HardenedData::TripleData<real_T> Sum1_n; /* '<S151>/Sum1' */
356 HardenedData::TripleData<real_T> Gain_1; /* '<S151>/Gain' */
357 HardenedData::TripleData<real_T> Product4_c; /* '<S151>/Product4' */
358 HardenedData::TripleData<real_T> Product2_gk; /* '<S151>/Product2' */
359 HardenedData::TripleData<real_T> Product3_i; /* '<S151>/Product3' */
360 HardenedData::TripleData<real_T> Sum2_g; /* '<S151>/Sum2' */
361 HardenedData::TripleData<real_T> Gain1_e; /* '<S151>/Gain1' */
362 HardenedData::TripleData<real_T> Product5_o; /* '<S151>/Product5' */
363 HardenedData::TripleData<real_T> Sum_oz; /* '<S151>/Sum' */
364 HardenedData::TripleData<real_T> Product_g; /* '<S152>/Product' */
365 HardenedData::TripleData<real_T> Product1_ew; /* '<S152>/Product1' */
366 HardenedData::TripleData<real_T> Sum1_m; /* '<S152>/Sum1' */
367 HardenedData::TripleData<real_T> Gain_b; /* '<S152>/Gain' */
368 HardenedData::TripleData<real_T> Product4_a; /* '<S152>/Product4' */
369 HardenedData::TripleData<real_T> Product6_o; /* '<S152>/Product6' */
370 HardenedData::TripleData<real_T> Product7_k; /* '<S152>/Product7' */
371 HardenedData::TripleData<real_T> Sum3_bi; /* '<S152>/Sum3' */
372 HardenedData::TripleData<real_T> Gain2_l; /* '<S152>/Gain2' */
373 HardenedData::TripleData<real_T> Product8_me; /* '<S152>/Product8' */
374 HardenedData::TripleData<real_T> Product2_fhq; /* '<S152>/Product2' */
375 HardenedData::TripleData<real_T> Product3_n; /* '<S152>/Product3' */
376 HardenedData::TripleData<real_T> Sum2_la; /* '<S152>/Sum2' */
377 HardenedData::TripleData<real_T> Gain1_ox; /* '<S152>/Gain1' */
378 HardenedData::TripleData<real_T> Product5_l; /* '<S152>/Product5' */
379 HardenedData::TripleData<real_T> Sum_oa; /* '<S152>/Sum' */
380 HardenedData::TripleData<real_T> Product_ew; /* '<S153>/Product' */
381 HardenedData::TripleData<real_T> Product1_m; /* '<S153>/Product1' */
382 HardenedData::TripleData<real_T> Sum1_h; /* '<S153>/Sum1' */
383 HardenedData::TripleData<real_T> Gain_kp; /* '<S153>/Gain' */
384 HardenedData::TripleData<real_T> Product4_g; /* '<S153>/Product4' */
385 HardenedData::TripleData<real_T> Product2_gh; /* '<S153>/Product2' */
386 HardenedData::TripleData<real_T> Product3_k4; /* '<S153>/Product3' */
387 HardenedData::TripleData<real_T> Sum2_i; /* '<S153>/Sum2' */
388 HardenedData::TripleData<real_T> Gain1_f; /* '<S153>/Gain1' */
389 HardenedData::TripleData<real_T> Product5_ij; /* '<S153>/Product5' */
390 HardenedData::TripleData<real_T> Product6_c; /* '<S153>/Product6' */
391 HardenedData::TripleData<real_T> Product7_a4; /* '<S153>/Product7' */
392 HardenedData::TripleData<real_T> Sum3_d; /* '<S153>/Sum3' */
393 HardenedData::TripleData<real_T> Gain2_n; /* '<S153>/Gain2' */
394 HardenedData::TripleData<real_T> Product8_f; /* '<S153>/Product8' */
395 HardenedData::TripleData<real_T> Sum_nx; /* '<S153>/Sum' */
396 HardenedData::TripleData<real_T> TmpSignalConversionAtDotProdu_k[3];
397 HardenedData::TripleData<real_T> jxk_b; /* '<S200>/j x k' */
398 HardenedData::TripleData<real_T> kxi_b; /* '<S200>/k x i' */
399 HardenedData::TripleData<real_T> ixj_h; /* '<S200>/i x j' */
400 HardenedData::TripleData<real_T> kxj_l; /* '<S201>/k x j' */
401 HardenedData::TripleData<real_T> ixk_e; /* '<S201>/i x k' */
402 HardenedData::TripleData<real_T> jxi_n; /* '<S201>/j x i' */
403 HardenedData::TripleData<real_T> Sum_m[3]; /* '<S198>/Sum' */
404 HardenedData::TripleData<real_T> jxk_e; /* '<S202>/j x k' */
405 HardenedData::TripleData<real_T> kxi_es; /* '<S202>/k x i' */
406 HardenedData::TripleData<real_T> ixj_n; /* '<S202>/i x j' */
407 HardenedData::TripleData<real_T> kxj_j; /* '<S203>/k x j' */
408 HardenedData::TripleData<real_T> ixk_j; /* '<S203>/i x k' */
409 HardenedData::TripleData<real_T> jxi_f; /* '<S203>/j x i' */
410 HardenedData::TripleData<real_T> Sum_pb[3]; /* '<S199>/Sum' */
411 HardenedData::TripleData<real_T> Product_dj; /* '<S161>/Product' */
412 HardenedData::TripleData<real_T> Product1_ig; /* '<S161>/Product1' */
413 HardenedData::TripleData<real_T> Product2_ad; /* '<S161>/Product2' */
414 HardenedData::TripleData<real_T> Product3_hf; /* '<S161>/Product3' */
415 HardenedData::TripleData<real_T> Sum_gt; /* '<S161>/Sum' */
416 HardenedData::TripleData<real_T> sqrt_i; /* '<S160>/sqrt' */
417 HardenedData::TripleData<real_T> Product2_nx; /* '<S156>/Product2' */
418 HardenedData::TripleData<real_T> Product6_a; /* '<S157>/Product6' */
419 HardenedData::TripleData<real_T> Product3_ce; /* '<S156>/Product3' */
420 HardenedData::TripleData<real_T> Product7_ha; /* '<S157>/Product7' */
421 HardenedData::TripleData<real_T> Sum3_l; /* '<S157>/Sum3' */
422 HardenedData::TripleData<real_T> Gain2_om; /* '<S157>/Gain2' */
423 HardenedData::TripleData<real_T> Product8_p; /* '<S157>/Product8' */
424 HardenedData::TripleData<real_T> Product1_a; /* '<S156>/Product1' */
425 HardenedData::TripleData<real_T> Product_mi; /* '<S157>/Product' */
426 HardenedData::TripleData<real_T> Product_pb; /* '<S156>/Product' */
427 HardenedData::TripleData<real_T> Product1_lr; /* '<S157>/Product1' */
428 HardenedData::TripleData<real_T> Sum1_a; /* '<S157>/Sum1' */
429 HardenedData::TripleData<real_T> Gain_p; /* '<S157>/Gain' */
430 HardenedData::TripleData<real_T> Product4_i; /* '<S157>/Product4' */
431 HardenedData::TripleData<real_T> Product2_nz; /* '<S157>/Product2' */
432 HardenedData::TripleData<real_T> Product3_lu; /* '<S157>/Product3' */
433 HardenedData::TripleData<real_T> Sum2_m; /* '<S157>/Sum2' */
434 HardenedData::TripleData<real_T> Gain1_c; /* '<S157>/Gain1' */
435 HardenedData::TripleData<real_T> Product5_jx; /* '<S157>/Product5' */
436 HardenedData::TripleData<real_T> Sum_g5; /* '<S157>/Sum' */
437 HardenedData::TripleData<real_T> Product_pe; /* '<S158>/Product' */
438 HardenedData::TripleData<real_T> Product1_mu; /* '<S158>/Product1' */
439 HardenedData::TripleData<real_T> Sum1_k; /* '<S158>/Sum1' */
440 HardenedData::TripleData<real_T> Gain_ay; /* '<S158>/Gain' */
441 HardenedData::TripleData<real_T> Product4_ok; /* '<S158>/Product4' */
442 HardenedData::TripleData<real_T> Product6_ld; /* '<S158>/Product6' */
443 HardenedData::TripleData<real_T> Product7_ei; /* '<S158>/Product7' */
444 HardenedData::TripleData<real_T> Sum3_h; /* '<S158>/Sum3' */
445 HardenedData::TripleData<real_T> Gain2_l2; /* '<S158>/Gain2' */
446 HardenedData::TripleData<real_T> Product8_d; /* '<S158>/Product8' */
447 HardenedData::TripleData<real_T> Product2_ea; /* '<S158>/Product2' */
448 HardenedData::TripleData<real_T> Product3_ll; /* '<S158>/Product3' */
449 HardenedData::TripleData<real_T> Sum2_mr; /* '<S158>/Sum2' */
450 HardenedData::TripleData<real_T> Gain1_n; /* '<S158>/Gain1' */
451 HardenedData::TripleData<real_T> Product5_a; /* '<S158>/Product5' */
452 HardenedData::TripleData<real_T> Sum_gz; /* '<S158>/Sum' */
453 HardenedData::TripleData<real_T> Product_am; /* '<S159>/Product' */
454 HardenedData::TripleData<real_T> Product1_bt; /* '<S159>/Product1' */
455 HardenedData::TripleData<real_T> Sum1_hk; /* '<S159>/Sum1' */
456 HardenedData::TripleData<real_T> Gain_g; /* '<S159>/Gain' */

```

```

457 HardenedData::TripleData<real_T> Product4_o5; /* '<S159>/Product4' */
458 HardenedData::TripleData<real_T> Product2_cc; /* '<S159>/Product2' */
459 HardenedData::TripleData<real_T> Product3_cd; /* '<S159>/Product3' */
460 HardenedData::TripleData<real_T> Sum2_o; /* '<S159>/Sum2' */
461 HardenedData::TripleData<real_T> Gain1_ni; /* '<S159>/Gain1' */
462 HardenedData::TripleData<real_T> Product5_n2; /* '<S159>/Product5' */
463 HardenedData::TripleData<real_T> Product6_d5; /* '<S159>/Product6' */
464 HardenedData::TripleData<real_T> Product7_em; /* '<S159>/Product7' */
465 HardenedData::TripleData<real_T> Sum3_jh; /* '<S159>/Sum3' */
466 HardenedData::TripleData<real_T> Gain2_ja; /* '<S159>/Gain2' */
467 HardenedData::TripleData<real_T> Product8_j; /* '<S159>/Product8' */
468 HardenedData::TripleData<real_T> Sum_pw; /* '<S159>/Sum' */
469 HardenedData::TripleData<real_T> TmpSignalConversionAtDotProdu_j[3];
470 HardenedData::TripleData<real_T> DotProduct3_ph; /* '<S178>/Dot Product3' */
471 HardenedData::TripleData<real_T> DotProduct1_o; /* '<S178>/Dot Product1' */
472 HardenedData::TripleData<real_T> DotProduct2_g; /* '<S178>/Dot Product2' */
473 HardenedData::TripleData<real_T> Divide1_p; /* '<S178>/Divide1' */
474 HardenedData::TripleData<real_T> Sqrt3_m; /* '<S178>/Sqrt3' */
475 HardenedData::TripleData<real_T> Divide_c; /* '<S178>/Divide' */
476 HardenedData::TripleData<real_T> Bias_h; /* '<S178>/Bias' */
477 HardenedData::TripleData<real_T> Abs_m; /* '<S178>/Abs' */
478 HardenedData::TripleData<real_T> Bias1_e; /* '<S178>/Bias1' */
479 HardenedData::TripleData<real_T> Abs1_i; /* '<S178>/Abs1' */
480 HardenedData::TripleData<real_T> Merge_l; /* '<S178>/Merge' */
481 HardenedData::TripleData<real_T> Switch_c[4]; /* '<S124>/Switch' */
482 HardenedData::TripleData<real_T> Product_h0; /* '<S273>/Product' */
483 HardenedData::TripleData<real_T> Product1_fm; /* '<S273>/Product1' */
484 HardenedData::TripleData<real_T> Product2_c2; /* '<S273>/Product2' */
485 HardenedData::TripleData<real_T> Product3_pq; /* '<S273>/Product3' */
486 HardenedData::TripleData<real_T> Sum_f; /* '<S273>/Sum' */
487 HardenedData::TripleData<real_T> sqrt_mo; /* '<S272>/sqrt' */
488 HardenedData::TripleData<real_T> Product_jc; /* '<S267>/Product' */
489 HardenedData::TripleData<real_T> Product1_o; /* '<S267>/Product1' */
490 HardenedData::TripleData<real_T> Product2_j; /* '<S267>/Product2' */
491 HardenedData::TripleData<real_T> Product3_lz; /* '<S267>/Product3' */
492 HardenedData::TripleData<real_T> fcn1_i; /* '<S128>/fcn1' */
493 HardenedData::TripleData<real_T> fcn2_f; /* '<S128>/fcn2' */
494 HardenedData::TripleData<real_T> fcn3_o; /* '<S128>/fcn3' */
495 HardenedData::TripleData<real_T> Merge_k; /* '<S268>/Merge' */
496 HardenedData::TripleData<real_T> fcn4_n; /* '<S128>/fcn4' */
497 HardenedData::TripleData<real_T> fcn5_g; /* '<S128>/fcn5' */
498 HardenedData::TripleData<real_T> VectorConcatenate_a[3]; /* '<S266>/Vector Concatenate' */
499 HardenedData::TripleData<real_T> DerivativeGain_k[3]; /* '<S229>/Derivative Gain' */
500 HardenedData::TripleData<real_T> Tsamp_f[3]; /* '<S232>/Tsamp' */
501 HardenedData::TripleData<real_T> UD_l[3]; /* '<S230>/UD' */
502 HardenedData::TripleData<real_T> Diff_e[3]; /* '<S230>/Diff' */
503 HardenedData::TripleData<real_T> IntegralGain_p[3]; /* '<S234>/Integral Gain' */
504 HardenedData::TripleData<real_T> Integrator_k[3]; /* '<S237>/Integrator' */
505 HardenedData::TripleData<real_T> ProportionalGain_h[3]; /* '<S242>/Proportional Gain' */
506 HardenedData::TripleData<real_T> is180degRot_l[3]; /* '<S137>/is 180deg Rot' */
507 HardenedData::TripleData<real_T> Product3_dz; /* '<S191>/Product3' */
508 HardenedData::TripleData<real_T> Product2_eq; /* '<S191>/Product2' */
509 HardenedData::TripleData<real_T> Product1_aa; /* '<S191>/Product1' */
510 HardenedData::TripleData<real_T> DotProduct2_n; /* '<S137>/Dot Product2' */
511 HardenedData::TripleData<real_T> DotProduct1_h; /* '<S137>/Dot Product1' */
512 HardenedData::TripleData<real_T> Divide1_h; /* '<S137>/Divide1' */
513 HardenedData::TripleData<real_T> Sqrt3_mq; /* '<S137>/Sqrt3' */
514 HardenedData::TripleData<real_T> DotProduct3_e; /* '<S137>/Dot Product3' */
515 HardenedData::TripleData<real_T> Add2_k; /* '<S137>/Add2' */
516 HardenedData::TripleData<real_T> Product_er; /* '<S191>/Product' */
517 HardenedData::TripleData<real_T> Sum_e; /* '<S191>/Sum' */
518 HardenedData::TripleData<real_T> sqrt_j; /* '<S190>/sqrt' */
519 HardenedData::TripleData<real_T> Product3_lzz; /* '<S181>/Product3' */
520 HardenedData::TripleData<real_T> Product3_ay; /* '<S146>/Product3' */
521 HardenedData::TripleData<real_T> Product2_ep; /* '<S181>/Product2' */
522 HardenedData::TripleData<real_T> Product2_bc; /* '<S146>/Product2' */
523 HardenedData::TripleData<real_T> Product1_ge; /* '<S181>/Product1' */
524 HardenedData::TripleData<real_T> Product1_fh; /* '<S146>/Product1' */
525 HardenedData::TripleData<real_T> Product_ey; /* '<S181>/Product' */
526 HardenedData::TripleData<real_T> Product_d0; /* '<S146>/Product' */
527 HardenedData::TripleData<real_T> Sum_ki; /* '<S146>/Sum' */
528 HardenedData::TripleData<real_T> Product3_aw; /* '<S147>/Product3' */
529 HardenedData::TripleData<real_T> Product2_pu; /* '<S147>/Product2' */
530 HardenedData::TripleData<real_T> Product1_b0; /* '<S147>/Product1' */
531 HardenedData::TripleData<real_T> Product_cq; /* '<S147>/Product' */
532 HardenedData::TripleData<real_T> Sum_oq; /* '<S147>/Sum' */
533 HardenedData::TripleData<real_T> Product3_h5; /* '<S148>/Product3' */
534 HardenedData::TripleData<real_T> Product2_i; /* '<S148>/Product2' */
535 HardenedData::TripleData<real_T> Product1_ai; /* '<S148>/Product1' */
536 HardenedData::TripleData<real_T> Product_cf; /* '<S148>/Product' */
537 HardenedData::TripleData<real_T> Sum_kc; /* '<S148>/Sum' */
538 HardenedData::TripleData<real_T> Product3_c1; /* '<S149>/Product3' */
539 HardenedData::TripleData<real_T> Product2_no; /* '<S149>/Product2' */
540 HardenedData::TripleData<real_T> Product1_d1; /* '<S149>/Product1' */
541 HardenedData::TripleData<real_T> Product_gr; /* '<S149>/Product' */
542 HardenedData::TripleData<real_T> Sum_n1; /* '<S149>/Sum' */
543 HardenedData::TripleData<real_T> Switch3[3]; /* '<S180>/Switch3' */
544 HardenedData::TripleData<real_T> jxi_m; /* '<S189>/j x i' */
545 HardenedData::TripleData<real_T> ixk_k; /* '<S189>/i x k' */
546 HardenedData::TripleData<real_T> kxj_f; /* '<S189>/k x j' */
547 HardenedData::TripleData<real_T> ixj_k; /* '<S188>/i x j' */
548 HardenedData::TripleData<real_T> kxi_bj; /* '<S188>/k x i' */
549 HardenedData::TripleData<real_T> jxk_l; /* '<S188>/j x k' */
550 HardenedData::TripleData<real_T> Sum_pd[3]; /* '<S187>/Sum' */
551 HardenedData::TripleData<real_T> Gain2_c; /* '<S180>/Gain2' */
552 HardenedData::TripleData<real_T> Gain_o2; /* '<S180>/Gain' */
553 HardenedData::TripleData<real_T> jxi_no; /* '<S183>/j x i' */
554 HardenedData::TripleData<real_T> ixk_b; /* '<S183>/i x k' */
555 HardenedData::TripleData<real_T> kxj_n; /* '<S183>/k x j' */
556 HardenedData::TripleData<real_T> ixj_l; /* '<S182>/i x j' */
557 HardenedData::TripleData<real_T> kxi_o; /* '<S182>/k x i' */
558 HardenedData::TripleData<real_T> jxk_d; /* '<S182>/j x k' */
559 HardenedData::TripleData<real_T> Sum_b[3]; /* '<S177>/Sum' */
560 HardenedData::TripleData<real_T> jxi_k; /* '<S168>/j x i' */

```



```

561 HardenedData :: TripleData<real_T> ixk_a; /* '<S168>/i x k' */
562 HardenedData :: TripleData<real_T> kxj_c; /* '<S168>/k x j' */
563 HardenedData :: TripleData<real_T> ixj_o; /* '<S167>/i x j' */
564 HardenedData :: TripleData<real_T> kxi_p; /* '<S167>/k x i' */
565 HardenedData :: TripleData<real_T> jxk_er; /* '<S167>/j x k' */
566 HardenedData :: TripleData<real_T> Sum_ff [3]; /* '<S162>/Sum' */
567 HardenedData :: TripleData<real_T> Dipole_x [3]; /* '<S2>/Magnetroquer control' */
568 HardenedData :: TripleData<real_T> Dipole_y [3]; /* '<S2>/Magnetroquer control' */
569 HardenedData :: TripleData<real_T> Dipole_z [3]; /* '<S2>/Magnetroquer control' */
570 HardenedData :: TripleData<real_T> bdot [3]; /* '<S2>/Derivative Function' */
571 #else
572 HardenedData :: SingleData<real_T> norm_b;
573 HardenedData :: SingleData<real_T> Merge [3]; /* '<S1>/Merge' */
574 HardenedData :: SingleData<real_T> jxk; /* '<S20>/j x k' */
575 HardenedData :: SingleData<real_T> kxi; /* '<S20>/k x i' */
576 HardenedData :: SingleData<real_T> ixj; /* '<S20>/i x j' */
577 HardenedData :: SingleData<real_T> kxj; /* '<S21>/k x j' */
578 HardenedData :: SingleData<real_T> ixk; /* '<S21>/i x k' */
579 HardenedData :: SingleData<real_T> jxi; /* '<S21>/j x i' */
580 HardenedData :: SingleData<real_T> Sum [3]; /* '<S13>/Sum' */
581 HardenedData :: SingleData<real_T> zr [3]; /* '<S7>/z = -r' */
582 HardenedData :: SingleData<real_T> ixj_f; /* '<S18>/i x j' */
583 HardenedData :: SingleData<real_T> jxk_o; /* '<S18>/j x k' */
584 HardenedData :: SingleData<real_T> kxi_e; /* '<S18>/k x i' */
585 HardenedData :: SingleData<real_T> ixk_g; /* '<S19>/i x k' */
586 HardenedData :: SingleData<real_T> jxi_l; /* '<S19>/j x i' */
587 HardenedData :: SingleData<real_T> kxj_a; /* '<S19>/k x j' */
588 HardenedData :: SingleData<real_T> Sum_l [3]; /* '<S12>/Sum' */
589 HardenedData :: SingleData<real_T> DotProduct; /* '<S15>/Dot Product' */
590 HardenedData :: SingleData<real_T> SumofElements; /* '<S15>/Sum of Elements' */
591 HardenedData :: SingleData<real_T> MathFunction1; /* '<S15>/Math Function1' */
592 HardenedData :: SingleData<real_T> DotProduct_p; /* '<S16>/Dot Product' */
593 HardenedData :: SingleData<real_T> SumofElements_c; /* '<S16>/Sum of Elements' */
594 HardenedData :: SingleData<real_T> MathFunction1_b; /* '<S16>/Math Function1' */
595 HardenedData :: SingleData<real_T> DotProduct_b; /* '<S17>/Dot Product' */
596 HardenedData :: SingleData<real_T> SumofElements_l; /* '<S17>/Sum of Elements' */
597 HardenedData :: SingleData<real_T> MathFunction1_m; /* '<S17>/Math Function1' */
598 HardenedData :: SingleData<real_T> ORFtoECEF [9]; /* '<S7>/ORF to ECEF' */
599 HardenedData :: SingleData<real_T> ECEFTtoORF [9]; /* '<S7>/ECEF to ORF' */
600 HardenedData :: SingleData<real_T> Add; /* '<S25>/Add' */
601 HardenedData :: SingleData<real_T> Merge_i [4]; /* '<S14>/Merge' */
602 HardenedData :: SingleData<real_T> Product; /* '<S111>/Product' */
603 HardenedData :: SingleData<real_T> Product1; /* '<S111>/Product1' */
604 HardenedData :: SingleData<real_T> Product2; /* '<S111>/Product2' */
605 HardenedData :: SingleData<real_T> Product3; /* '<S111>/Product3' */
606 HardenedData :: SingleData<real_T> Sum_i; /* '<S111>/Sum' */
607 HardenedData :: SingleData<real_T> Divide; /* '<S9>/Divide' */
608 HardenedData :: SingleData<real_T> Product_j; /* '<S112>/Product' */
609 HardenedData :: SingleData<real_T> UnaryMinus; /* '<S110>/Unary Minus' */
610 HardenedData :: SingleData<real_T> Divide1; /* '<S9>/Divide1' */
611 HardenedData :: SingleData<real_T> Product1_h; /* '<S112>/Product1' */
612 HardenedData :: SingleData<real_T> UnaryMinus1; /* '<S110>/Unary Minus1' */
613 HardenedData :: SingleData<real_T> Divide2; /* '<S9>/Divide2' */
614 HardenedData :: SingleData<real_T> Product2_d; /* '<S112>/Product2' */
615 HardenedData :: SingleData<real_T> UnaryMinus2; /* '<S110>/Unary Minus2' */
616 HardenedData :: SingleData<real_T> Divide3; /* '<S9>/Divide3' */
617 HardenedData :: SingleData<real_T> Product3_c; /* '<S112>/Product3' */
618 HardenedData :: SingleData<real_T> Sum_p; /* '<S112>/Sum' */
619 HardenedData :: SingleData<real_T> Product_b; /* '<S123>/Product' */
620 HardenedData :: SingleData<real_T> Product_p; /* '<S113>/Product' */
621 HardenedData :: SingleData<real_T> Product1_i; /* '<S113>/Product1' */
622 HardenedData :: SingleData<real_T> Product2_o; /* '<S113>/Product2' */
623 HardenedData :: SingleData<real_T> Product3_p; /* '<S113>/Product3' */
624 HardenedData :: SingleData<real_T> Sum_iw; /* '<S113>/Sum' */
625 HardenedData :: SingleData<real_T> Product1_in; /* '<S123>/Product1' */
626 HardenedData :: SingleData<real_T> Product_by; /* '<S114>/Product' */
627 HardenedData :: SingleData<real_T> Product1_f; /* '<S114>/Product1' */
628 HardenedData :: SingleData<real_T> Product2_m; /* '<S114>/Product2' */
629 HardenedData :: SingleData<real_T> Product3_pa; /* '<S114>/Product3' */
630 HardenedData :: SingleData<real_T> Sum_d; /* '<S114>/Sum' */
631 HardenedData :: SingleData<real_T> Product2_n; /* '<S123>/Product2' */
632 HardenedData :: SingleData<real_T> Product_pd; /* '<S115>/Product' */
633 HardenedData :: SingleData<real_T> Product1_d; /* '<S115>/Product1' */
634 HardenedData :: SingleData<real_T> Product2_nn; /* '<S115>/Product2' */
635 HardenedData :: SingleData<real_T> Product3_l; /* '<S115>/Product3' */
636 HardenedData :: SingleData<real_T> Sum_o; /* '<S115>/Sum' */
637 HardenedData :: SingleData<real_T> Product3_ph; /* '<S123>/Product3' */
638 HardenedData :: SingleData<real_T> Sum_g; /* '<S123>/Sum' */
639 HardenedData :: SingleData<real_T> sqrt_g; /* '<S122>/sqrt' */
640 HardenedData :: SingleData<real_T> Product_h; /* '<S117>/Product' */
641 HardenedData :: SingleData<real_T> Product1_j; /* '<S117>/Product1' */
642 HardenedData :: SingleData<real_T> Product2_dj; /* '<S117>/Product2' */
643 HardenedData :: SingleData<real_T> Product3_e; /* '<S117>/Product3' */
644 HardenedData :: SingleData<real_T> fcn1; /* '<S11>/fcn1' */
645 HardenedData :: SingleData<real_T> fcn2; /* '<S11>/fcn2' */
646 HardenedData :: SingleData<real_T> fcn3; /* '<S11>/fcn3' */
647 HardenedData :: SingleData<real_T> Merge_il; /* '<S118>/Merge' */
648 HardenedData :: SingleData<real_T> fcn4; /* '<S11>/fcn4' */
649 HardenedData :: SingleData<real_T> fcn5; /* '<S11>/fcn5' */
650 HardenedData :: SingleData<real_T> VectorConcatenate [3]; /* '<S116>/Vector Concatenate' */
651 */
652 HardenedData :: SingleData<real_T> DerivativeGain [3]; /* '<S85>/Derivative Gain' */
653 HardenedData :: SingleData<real_T> Tsamp [3]; /* '<S88>/Tsamp' */
654 HardenedData :: SingleData<real_T> UD [3]; /* '<S86>/UD' */
655 HardenedData :: SingleData<real_T> Diff [3]; /* '<S86>/Diff' */
656 HardenedData :: SingleData<real_T> IntegralGain [3]; /* '<S90>/Integral Gain' */
657 HardenedData :: SingleData<real_T> Integrator [3]; /* '<S93>/Integrator' */
658 HardenedData :: SingleData<real_T> ProportionalGain [3]; /* '<S98>/Proportional Gain' */
659 */
660 HardenedData :: SingleData<real_T> Product_a; /* '<S58>/Product' */
661 HardenedData :: SingleData<real_T> Product1_k; /* '<S58>/Product1' */
662 HardenedData :: SingleData<real_T> Product2_e; /* '<S58>/Product2' */
663 HardenedData :: SingleData<real_T> Product3_g; /* '<S58>/Product3' */
664 HardenedData :: SingleData<real_T> Product4; /* '<S58>/Product4' */

```


663	HardenedData :: SingleData<real_T>	Product5;	/* '<S58>/Product5' */
664	HardenedData :: SingleData<real_T>	detMatrix;	/* '<S58>/Sum' */
665	HardenedData :: SingleData<real_T>	Bias;	/* '<S52>/Bias' */
666	HardenedData :: SingleData<real_T>	Abs1;	/* '<S52>/Abs1' */
667	HardenedData :: SingleData<real_T>	MathFunction [9];	/* '<S51>/Math Function' */
668	HardenedData :: SingleData<real_T>	Product_c [9];	/* '<S51>/Product' */
669	HardenedData :: SingleData<real_T>	Bias1 [9];	/* '<S51>/Bias1' */
670	HardenedData :: SingleData<real_T>	Abs2 [9];	/* '<S51>/Abs2' */
671	HardenedData :: SingleData<real_T>	Add_m;	/* '<S34>/Add' */
672	HardenedData :: SingleData<real_T>	sqrt_m;	/* '<S26>/sqrt' */
673	HardenedData :: SingleData<real_T>	Switch [2];	/* '<S33>/Switch' */
674	HardenedData :: SingleData<real_T>	Product_k;	/* '<S33>/Product' */
675	HardenedData :: SingleData<real_T>	Add_o;	/* '<S32>/Add' */
676	HardenedData :: SingleData<real_T>	Add_ov;	/* '<S30>/Add' */
677	HardenedData :: SingleData<real_T>	Add_l;	/* '<S31>/Add' */
678	HardenedData :: SingleData<real_T>	Product_cy [3];	/* '<S26>/Product' */
679	HardenedData :: SingleData<real_T>	Add_f;	/* '<S44>/Add' */
680	HardenedData :: SingleData<real_T>	sqrt_l;	/* '<S28>/sqrt' */
681	HardenedData :: SingleData<real_T>	Add_e;	/* '<S40>/Add' */
682	HardenedData :: SingleData<real_T>	Add_l2;	/* '<S41>/Add' */
683	HardenedData :: SingleData<real_T>	Add_ek;	/* '<S42>/Add' */
684	HardenedData :: SingleData<real_T>	Switch_b [2];	/* '<S43>/Switch' */
685	HardenedData :: SingleData<real_T>	Product_d;	/* '<S43>/Product' */
686	HardenedData :: SingleData<real_T>	Product_i [3];	/* '<S28>/Product' */
687	HardenedData :: SingleData<real_T>	Add_g;	/* '<S39>/Add' */
688	HardenedData :: SingleData<real_T>	sqrt_mw;	/* '<S27>/sqrt' */
689	HardenedData :: SingleData<real_T>	Add_gw;	/* '<S37>/Add' */
690	HardenedData :: SingleData<real_T>	Add_fc;	/* '<S36>/Add' */
691	HardenedData :: SingleData<real_T>	Add_d;	/* '<S35>/Add' */
692	HardenedData :: SingleData<real_T>	Switch_f [2];	/* '<S38>/Switch' */
693	HardenedData :: SingleData<real_T>	Product_e;	/* '<S38>/Product' */
694	HardenedData :: SingleData<real_T>	Product_o [3];	/* '<S27>/Product' */
695	HardenedData :: SingleData<real_T>	Sum_gh;	/* '<S23>/Sum' */
696	HardenedData :: SingleData<real_T>	sqrt_a;	/* '<S23>/sqrt' */
697	HardenedData :: SingleData<real_T>	Gain1;	/* '<S23>/Gain1' */
698	HardenedData :: SingleData<real_T>	Add_k;	/* '<S46>/Add' */
699	HardenedData :: SingleData<real_T>	Add_el;	/* '<S45>/Add' */
700	HardenedData :: SingleData<real_T>	Add_a;	/* '<S47>/Add' */
701	HardenedData :: SingleData<real_T>	Merge_m;	/* '<S129>/Merge' */
702	HardenedData :: SingleData<real_T>	Product_n;	/* '<S259>/Product' */
703	HardenedData :: SingleData<real_T>	Product1_ii;	/* '<S259>/Product1' */
704	HardenedData :: SingleData<real_T>	Product2_c;	/* '<S259>/Product2' */
705	HardenedData :: SingleData<real_T>	Product3_d;	/* '<S259>/Product3' */
706	HardenedData :: SingleData<real_T>	Sum_j;	/* '<S259>/Sum' */
707	HardenedData :: SingleData<real_T>	sqrt_n;	/* '<S258>/sqrt' */
708	HardenedData :: SingleData<real_T>	Product2_oi;	/* '<S254>/Product2' */
709	HardenedData :: SingleData<real_T>	Product6;	/* '<S255>/Product6' */
710	HardenedData :: SingleData<real_T>	Product3_a;	/* '<S254>/Product3' */
711	HardenedData :: SingleData<real_T>	Product7;	/* '<S255>/Product7' */
712	HardenedData :: SingleData<real_T>	Sum3;	/* '<S255>/Sum3' */
713	HardenedData :: SingleData<real_T>	Gain2;	/* '<S255>/Gain2' */
714	HardenedData :: SingleData<real_T>	Product8;	/* '<S255>/Product8' */
715	HardenedData :: SingleData<real_T>	Product1_fv;	/* '<S254>/Product1' */
716	HardenedData :: SingleData<real_T>	Product_jb;	/* '<S255>/Product' */
717	HardenedData :: SingleData<real_T>	Product_cw;	/* '<S254>/Product' */
718	HardenedData :: SingleData<real_T>	Product1_b;	/* '<S255>/Product1' */
719	HardenedData :: SingleData<real_T>	Sum1;	/* '<S255>/Sum1' */
720	HardenedData :: SingleData<real_T>	Gain;	/* '<S255>/Gain' */
721	HardenedData :: SingleData<real_T>	Product4_p;	/* '<S255>/Product4' */
722	HardenedData :: SingleData<real_T>	Product2_ox;	/* '<S255>/Product2' */
723	HardenedData :: SingleData<real_T>	Product3_pv;	/* '<S255>/Product3' */
724	HardenedData :: SingleData<real_T>	Sum2;	/* '<S255>/Sum2' */
725	HardenedData :: SingleData<real_T>	Gain1_l;	/* '<S255>/Gain1' */
726	HardenedData :: SingleData<real_T>	Product5_g;	/* '<S255>/Product5' */
727	HardenedData :: SingleData<real_T>	Sum_k;	/* '<S255>/Sum' */
728	HardenedData :: SingleData<real_T>	Product_if;	/* '<S256>/Product' */
729	HardenedData :: SingleData<real_T>	Product1_e;	/* '<S256>/Product1' */
730	HardenedData :: SingleData<real_T>	Sum1_b;	/* '<S256>/Sum1' */
731	HardenedData :: SingleData<real_T>	Gain_a;	/* '<S256>/Gain' */
732	HardenedData :: SingleData<real_T>	Product4_k;	/* '<S256>/Product4' */
733	HardenedData :: SingleData<real_T>	Product6_d;	/* '<S256>/Product6' */
734	HardenedData :: SingleData<real_T>	Product7_e;	/* '<S256>/Product7' */
735	HardenedData :: SingleData<real_T>	Sum3_p;	/* '<S256>/Sum3' */
736	HardenedData :: SingleData<real_T>	Gain2_a;	/* '<S256>/Gain2' */
737	HardenedData :: SingleData<real_T>	Product8_c;	/* '<S256>/Product8' */
738	HardenedData :: SingleData<real_T>	Product2_l;	/* '<S256>/Product2' */
739	HardenedData :: SingleData<real_T>	Product3_c5;	/* '<S256>/Product3' */
740	HardenedData :: SingleData<real_T>	Sum2_k;	/* '<S256>/Sum2' */
741	HardenedData :: SingleData<real_T>	Gain1_d;	/* '<S256>/Gain1' */
742	HardenedData :: SingleData<real_T>	Product5_j;	/* '<S256>/Product5' */
743	HardenedData :: SingleData<real_T>	Sum_lm;	/* '<S256>/Sum' */
744	HardenedData :: SingleData<real_T>	Product_c3;	/* '<S257>/Product' */
745	HardenedData :: SingleData<real_T>	Product1_c;	/* '<S257>/Product1' */
746	HardenedData :: SingleData<real_T>	Sum1_bu;	/* '<S257>/Sum1' */
747	HardenedData :: SingleData<real_T>	Gain_k;	/* '<S257>/Gain' */
748	HardenedData :: SingleData<real_T>	Product4_m;	/* '<S257>/Product4' */
749	HardenedData :: SingleData<real_T>	Product2_k;	/* '<S257>/Product2' */
750	HardenedData :: SingleData<real_T>	Product3_f;	/* '<S257>/Product3' */
751	HardenedData :: SingleData<real_T>	Sum2_p;	/* '<S257>/Sum2' */
752	HardenedData :: SingleData<real_T>	Gain1_i;	/* '<S257>/Gain1' */
753	HardenedData :: SingleData<real_T>	Product5_d;	/* '<S257>/Product5' */
754	HardenedData :: SingleData<real_T>	Product6_b;	/* '<S257>/Product6' */
755	HardenedData :: SingleData<real_T>	Product7_h;	/* '<S257>/Product7' */
756	HardenedData :: SingleData<real_T>	Sum3_e;	/* '<S257>/Sum3' */
757	HardenedData :: SingleData<real_T>	Gain2_b;	/* '<S257>/Gain2' */
758	HardenedData :: SingleData<real_T>	Product8_g;	/* '<S257>/Product8' */
759	HardenedData :: SingleData<real_T>	Sum_n;	/* '<S257>/Sum' */
760	HardenedData :: SingleData<real_T>	TmpSignalConversionAtDotProduct [3];	
761	HardenedData :: SingleData<real_T>	Product_jz;	/* '<S265>/Product' */
762	HardenedData :: SingleData<real_T>	Product1_bp;	/* '<S265>/Product1' */
763	HardenedData :: SingleData<real_T>	Product2_b;	/* '<S265>/Product2' */
764	HardenedData :: SingleData<real_T>	Product3_dh;	/* '<S265>/Product3' */
765	HardenedData :: SingleData<real_T>	Sum_d5;	/* '<S265>/Sum' */
766	HardenedData :: SingleData<real_T>	sqrt_k;	/* '<S264>/sqrt' */

767	HardenedData :: SingleData<real_T>	Product2_lq;	/* '<S260>/Product2' */
768	HardenedData :: SingleData<real_T>	Product6_l;	/* '<S261>/Product6' */
769	HardenedData :: SingleData<real_T>	Product3_h;	/* '<S260>/Product3' */
770	HardenedData :: SingleData<real_T>	Product7_d;	/* '<S261>/Product7' */
771	HardenedData :: SingleData<real_T>	Sum3_j;	/* '<S261>/Sum3' */
772	HardenedData :: SingleData<real_T>	Gain2_f;	/* '<S261>/Gain2' */
773	HardenedData :: SingleData<real_T>	Product8_m;	/* '<S261>/Product8' */
774	HardenedData :: SingleData<real_T>	Product1_b1;	/* '<S260>/Product1' */
775	HardenedData :: SingleData<real_T>	Product_h1;	/* '<S261>/Product' */
776	HardenedData :: SingleData<real_T>	Product_f;	/* '<S260>/Product' */
777	HardenedData :: SingleData<real_T>	Product1_l;	/* '<S261>/Product1' */
778	HardenedData :: SingleData<real_T>	Sum1_d;	/* '<S261>/Sum1' */
779	HardenedData :: SingleData<real_T>	Gain_j;	/* '<S261>/Gain' */
780	HardenedData :: SingleData<real_T>	Product4_o;	/* '<S261>/Product4' */
781	HardenedData :: SingleData<real_T>	Product2_p;	/* '<S261>/Product2' */
782	HardenedData :: SingleData<real_T>	Product3_lx;	/* '<S261>/Product3' */
783	HardenedData :: SingleData<real_T>	Sum2_p3;	/* '<S261>/Sum2' */
784	HardenedData :: SingleData<real_T>	Gain1_o;	/* '<S261>/Gain1' */
785	HardenedData :: SingleData<real_T>	Product5_i;	/* '<S261>/Product5' */
786	HardenedData :: SingleData<real_T>	Sum_kk;	/* '<S261>/Sum' */
787	HardenedData :: SingleData<real_T>	Product_bg;	/* '<S262>/Product' */
788	HardenedData :: SingleData<real_T>	Product1_l3;	/* '<S262>/Product1' */
789	HardenedData :: SingleData<real_T>	Sum1_l;	/* '<S262>/Sum1' */
790	HardenedData :: SingleData<real_T>	Gain_i;	/* '<S262>/Gain' */
791	HardenedData :: SingleData<real_T>	Product4_pt;	/* '<S262>/Product4' */
792	HardenedData :: SingleData<real_T>	Product6_p;	/* '<S262>/Product6' */
793	HardenedData :: SingleData<real_T>	Product7_dj;	/* '<S262>/Product7' */
794	HardenedData :: SingleData<real_T>	Sum3_o;	/* '<S262>/Sum3' */
795	HardenedData :: SingleData<real_T>	Gain2_j;	/* '<S262>/Gain2' */
796	HardenedData :: SingleData<real_T>	Product8_e;	/* '<S262>/Product8' */
797	HardenedData :: SingleData<real_T>	Product2_mf;	/* '<S262>/Product2' */
798	HardenedData :: SingleData<real_T>	Product3_k;	/* '<S262>/Product3' */
799	HardenedData :: SingleData<real_T>	Sum2_a;	/* '<S262>/Sum2' */
800	HardenedData :: SingleData<real_T>	Gain1_oi;	/* '<S262>/Gain1' */
801	HardenedData :: SingleData<real_T>	Product5_b;	/* '<S262>/Product5' */
802	HardenedData :: SingleData<real_T>	Sum_ln;	/* '<S262>/Sum' */
803	HardenedData :: SingleData<real_T>	Product_l;	/* '<S263>/Product' */
804	HardenedData :: SingleData<real_T>	Product1_id;	/* '<S263>/Product1' */
805	HardenedData :: SingleData<real_T>	Sum1_d1;	/* '<S263>/Sum1' */
806	HardenedData :: SingleData<real_T>	Gain_o;	/* '<S263>/Gain' */
807	HardenedData :: SingleData<real_T>	Product4_d;	/* '<S263>/Product4' */
808	HardenedData :: SingleData<real_T>	Product2_f;	/* '<S263>/Product2' */
809	HardenedData :: SingleData<real_T>	Product3_gx;	/* '<S263>/Product3' */
810	HardenedData :: SingleData<real_T>	Sum2_l;	/* '<S263>/Sum2' */
811	HardenedData :: SingleData<real_T>	Gain1_g;	/* '<S263>/Gain1' */
812	HardenedData :: SingleData<real_T>	Product5_n;	/* '<S263>/Product5' */
813	HardenedData :: SingleData<real_T>	Product6_j;	/* '<S263>/Product6' */
814	HardenedData :: SingleData<real_T>	Product7_g;	/* '<S263>/Product7' */
815	HardenedData :: SingleData<real_T>	Sum3_b;	/* '<S263>/Sum3' */
816	HardenedData :: SingleData<real_T>	Gain2_o;	/* '<S263>/Gain2' */
817	HardenedData :: SingleData<real_T>	Product8_a;	/* '<S263>/Product8' */
818	HardenedData :: SingleData<real_T>	Sum_kb;	/* '<S263>/Sum' */
819	HardenedData :: SingleData<real_T>	TmpSignalConversionAtDotProdu_d [3];	
820	HardenedData :: SingleData<real_T>	DotProduct3;	/* '<S130>/Dot Product3' */
821	HardenedData :: SingleData<real_T>	DotProduct1;	/* '<S130>/Dot Product1' */
822	HardenedData :: SingleData<real_T>	DotProduct2;	/* '<S130>/Dot Product2' */
823	HardenedData :: SingleData<real_T>	Divide1_k;	/* '<S130>/Divide1' */
824	HardenedData :: SingleData<real_T>	Sqrt3;	/* '<S130>/Sqrt3' */
825	HardenedData :: SingleData<real_T>	Divide_o;	/* '<S130>/Divide' */
826	HardenedData :: SingleData<real_T>	Bias_k;	/* '<S130>/Bias' */
827	HardenedData :: SingleData<real_T>	Abs;	/* '<S130>/Abs' */
828	HardenedData :: SingleData<real_T>	Bias1_j;	/* '<S130>/Bias1' */
829	HardenedData :: SingleData<real_T>	Abs1_d;	/* '<S130>/Abs1' */
830	HardenedData :: SingleData<real_T>	Merge_ip;	/* '<S130>/Merge' */
831	HardenedData :: SingleData<real_T>	DotProduct3_o;	/* '<S136>/Dot Product3' */
832	HardenedData :: SingleData<real_T>	DotProduct2_c;	/* '<S136>/Dot Product2' */
833	HardenedData :: SingleData<real_T>	Divide1_d;	/* '<S136>/Divide1' */
834	HardenedData :: SingleData<real_T>	Sqrt3_j;	/* '<S136>/Sqrt3' */
835	HardenedData :: SingleData<real_T>	Add2;	/* '<S136>/Add2' */
836	HardenedData :: SingleData<real_T>	Product_i2;	/* '<S176>/Product' */
837	HardenedData :: SingleData<real_T>	DotProduct3_p;	/* '<S163>/Dot Product3' */
838	HardenedData :: SingleData<real_T>	DotProduct2_f;	/* '<S163>/Dot Product2' */
839	HardenedData :: SingleData<real_T>	Divide1_b;	/* '<S163>/Divide1' */
840	HardenedData :: SingleData<real_T>	Sqrt3_l;	/* '<S163>/Sqrt3' */
841	HardenedData :: SingleData<real_T>	Divide_k;	/* '<S163>/Divide' */
842	HardenedData :: SingleData<real_T>	Bias_b;	/* '<S163>/Bias' */
843	HardenedData :: SingleData<real_T>	Abs_o;	/* '<S163>/Abs' */
844	HardenedData :: SingleData<real_T>	Bias1_m;	/* '<S163>/Bias1' */
845	HardenedData :: SingleData<real_T>	Abs1_l;	/* '<S163>/Abs1' */
846	HardenedData :: SingleData<real_T>	Merge_d;	/* '<S163>/Merge' */
847	HardenedData :: SingleData<real_T>	is180degRot [3];	/* '<S136>/is 180deg Rot' */
848	HardenedData :: SingleData<real_T>	Product1_g;	/* '<S176>/Product1' */
849	HardenedData :: SingleData<real_T>	Product2_g;	/* '<S176>/Product2' */
850	HardenedData :: SingleData<real_T>	Product3_a;	/* '<S176>/Product3' */
851	HardenedData :: SingleData<real_T>	Sum_c;	/* '<S176>/Sum' */
852	HardenedData :: SingleData<real_T>	sqrt_h;	/* '<S175>/sqrt' */
853	HardenedData :: SingleData<real_T>	Product_fv;	/* '<S166>/Product' */
854	HardenedData :: SingleData<real_T>	Product2_a;	/* '<S166>/Product2' */
855	HardenedData :: SingleData<real_T>	Product_l5;	/* '<S155>/Product' */
856	HardenedData :: SingleData<real_T>	Product1_n;	/* '<S166>/Product1' */
857	HardenedData :: SingleData<real_T>	Product1_lg;	/* '<S155>/Product1' */
858	HardenedData :: SingleData<real_T>	Product2_fh;	/* '<S155>/Product2' */
859	HardenedData :: SingleData<real_T>	Product3_pe;	/* '<S166>/Product3' */
860	HardenedData :: SingleData<real_T>	Product3_l0;	/* '<S155>/Product3' */
861	HardenedData :: SingleData<real_T>	Sum_n3;	/* '<S155>/Sum' */
862	HardenedData :: SingleData<real_T>	sqrt_mv;	/* '<S154>/sqrt' */
863	HardenedData :: SingleData<real_T>	Product2_ke;	/* '<S150>/Product2' */
864	HardenedData :: SingleData<real_T>	Product6_e;	/* '<S151>/Product6' */
865	HardenedData :: SingleData<real_T>	Product3_cc;	/* '<S150>/Product3' */
866	HardenedData :: SingleData<real_T>	Product7_a;	/* '<S151>/Product7' */
867	HardenedData :: SingleData<real_T>	Sum3_b5;	/* '<S151>/Sum3' */
868	HardenedData :: SingleData<real_T>	Gain2_p;	/* '<S151>/Gain2' */
869	HardenedData :: SingleData<real_T>	Product8_k;	/* '<S151>/Product8' */
870	HardenedData :: SingleData<real_T>	Product1_l0;	/* '<S150>/Product1' */

871	HardenedData :: SingleData<real_T>	Product_da;	/* '<S151>/Product' */
872	HardenedData :: SingleData<real_T>	Product_m;	/* '<S150>/Product' */
873	HardenedData :: SingleData<real_T>	Product1_dj;	/* '<S151>/Product1' */
874	HardenedData :: SingleData<real_T>	Sum1_n;	/* '<S151>/Sum1' */
875	HardenedData :: SingleData<real_T>	Gain_l;	/* '<S151>/Gain' */
876	HardenedData :: SingleData<real_T>	Product4_c;	/* '<S151>/Product4' */
877	HardenedData :: SingleData<real_T>	Product2_gk;	/* '<S151>/Product2' */
878	HardenedData :: SingleData<real_T>	Product3_i;	/* '<S151>/Product3' */
879	HardenedData :: SingleData<real_T>	Sum2_g;	/* '<S151>/Sum2' */
880	HardenedData :: SingleData<real_T>	Gain1_e;	/* '<S151>/Gain1' */
881	HardenedData :: SingleData<real_T>	Product5_o;	/* '<S151>/Product5' */
882	HardenedData :: SingleData<real_T>	Sum_oz;	/* '<S151>/Sum' */
883	HardenedData :: SingleData<real_T>	Product_g;	/* '<S152>/Product' */
884	HardenedData :: SingleData<real_T>	Product1_ew;	/* '<S152>/Product1' */
885	HardenedData :: SingleData<real_T>	Sum1_m;	/* '<S152>/Sum1' */
886	HardenedData :: SingleData<real_T>	Gain_b;	/* '<S152>/Gain' */
887	HardenedData :: SingleData<real_T>	Product4_a;	/* '<S152>/Product4' */
888	HardenedData :: SingleData<real_T>	Product6_o;	/* '<S152>/Product6' */
889	HardenedData :: SingleData<real_T>	Product7_k;	/* '<S152>/Product7' */
890	HardenedData :: SingleData<real_T>	Sum3_bi;	/* '<S152>/Sum3' */
891	HardenedData :: SingleData<real_T>	Gain2_l;	/* '<S152>/Gain2' */
892	HardenedData :: SingleData<real_T>	Product8_me;	/* '<S152>/Product8' */
893	HardenedData :: SingleData<real_T>	Product2_fhq;	/* '<S152>/Product2' */
894	HardenedData :: SingleData<real_T>	Product3_n;	/* '<S152>/Product3' */
895	HardenedData :: SingleData<real_T>	Sum2_la;	/* '<S152>/Sum2' */
896	HardenedData :: SingleData<real_T>	Gain1_ox;	/* '<S152>/Gain1' */
897	HardenedData :: SingleData<real_T>	Product5_l;	/* '<S152>/Product5' */
898	HardenedData :: SingleData<real_T>	Sum_oa;	/* '<S152>/Sum' */
899	HardenedData :: SingleData<real_T>	Product_ew;	/* '<S153>/Product' */
900	HardenedData :: SingleData<real_T>	Product1_m;	/* '<S153>/Product1' */
901	HardenedData :: SingleData<real_T>	Sum1_h;	/* '<S153>/Sum1' */
902	HardenedData :: SingleData<real_T>	Gain_kp;	/* '<S153>/Gain' */
903	HardenedData :: SingleData<real_T>	Product4_g;	/* '<S153>/Product4' */
904	HardenedData :: SingleData<real_T>	Product2_gh;	/* '<S153>/Product2' */
905	HardenedData :: SingleData<real_T>	Product3_k4;	/* '<S153>/Product3' */
906	HardenedData :: SingleData<real_T>	Sum2_i;	/* '<S153>/Sum2' */
907	HardenedData :: SingleData<real_T>	Gain1_f;	/* '<S153>/Gain1' */
908	HardenedData :: SingleData<real_T>	Product5_ij;	/* '<S153>/Product5' */
909	HardenedData :: SingleData<real_T>	Product6_c;	/* '<S153>/Product6' */
910	HardenedData :: SingleData<real_T>	Product7_a4;	/* '<S153>/Product7' */
911	HardenedData :: SingleData<real_T>	Sum3_d;	/* '<S153>/Sum3' */
912	HardenedData :: SingleData<real_T>	Gain2_n;	/* '<S153>/Gain2' */
913	HardenedData :: SingleData<real_T>	Product8_f;	/* '<S153>/Product8' */
914	HardenedData :: SingleData<real_T>	Sum_nx;	/* '<S153>/Sum' */
915	HardenedData :: SingleData<real_T>	TmpSignalConversionAtDotProdu_k[3];	/* '<S200>/j x k' */
916	HardenedData :: SingleData<real_T>	jxk_b;	/* '<S200>/k x i' */
917	HardenedData :: SingleData<real_T>	kxi_b;	/* '<S200>/i x j' */
918	HardenedData :: SingleData<real_T>	ixj_h;	/* '<S201>/k x j' */
919	HardenedData :: SingleData<real_T>	kxj_l;	/* '<S201>/i x k' */
920	HardenedData :: SingleData<real_T>	ixk_e;	/* '<S201>/j x i' */
921	HardenedData :: SingleData<real_T>	jxi_n;	/* '<S198>/Sum' */
922	HardenedData :: SingleData<real_T>	Sum_m[3];	/* '<S202>/j x k' */
923	HardenedData :: SingleData<real_T>	jxk_e;	/* '<S202>/k x i' */
924	HardenedData :: SingleData<real_T>	kxi_es;	/* '<S202>/i x j' */
925	HardenedData :: SingleData<real_T>	ixj_n;	/* '<S203>/k x j' */
926	HardenedData :: SingleData<real_T>	kxj_j;	/* '<S203>/i x k' */
927	HardenedData :: SingleData<real_T>	ixk_j;	/* '<S203>/j x i' */
928	HardenedData :: SingleData<real_T>	jxi_f;	/* '<S199>/Sum' */
929	HardenedData :: SingleData<real_T>	Sum_pb[3];	/* '<S161>/Product' */
930	HardenedData :: SingleData<real_T>	Product_dj;	/* '<S161>/Product1' */
931	HardenedData :: SingleData<real_T>	Product1_ig;	/* '<S161>/Product2' */
932	HardenedData :: SingleData<real_T>	Product2_ad;	/* '<S161>/Product3' */
933	HardenedData :: SingleData<real_T>	Product3_hf;	/* '<S161>/Sum' */
934	HardenedData :: SingleData<real_T>	Sum_gt;	/* '<S160>/sqrt' */
935	HardenedData :: SingleData<real_T>	sqrt_i;	/* '<S156>/Product2' */
936	HardenedData :: SingleData<real_T>	Product2_nx;	/* '<S157>/Product6' */
937	HardenedData :: SingleData<real_T>	Product6_a;	/* '<S156>/Product3' */
938	HardenedData :: SingleData<real_T>	Product3_ce;	/* '<S157>/Product7' */
939	HardenedData :: SingleData<real_T>	Product7_ha;	/* '<S157>/Sum3' */
940	HardenedData :: SingleData<real_T>	Sum3_l;	/* '<S157>/Gain2' */
941	HardenedData :: SingleData<real_T>	Gain2_om;	/* '<S157>/Product8' */
942	HardenedData :: SingleData<real_T>	Product8_p;	/* '<S156>/Product1' */
943	HardenedData :: SingleData<real_T>	Product1_a;	/* '<S157>/Product' */
944	HardenedData :: SingleData<real_T>	Product_mi;	/* '<S156>/Product' */
945	HardenedData :: SingleData<real_T>	Product_pb;	/* '<S157>/Product1' */
946	HardenedData :: SingleData<real_T>	Product1_lr;	/* '<S157>/Sum1' */
947	HardenedData :: SingleData<real_T>	Sum1_a;	/* '<S157>/Gain' */
948	HardenedData :: SingleData<real_T>	Gain_p;	/* '<S157>/Product4' */
949	HardenedData :: SingleData<real_T>	Product4_i;	/* '<S157>/Product2' */
950	HardenedData :: SingleData<real_T>	Product2_nz;	/* '<S157>/Product3' */
951	HardenedData :: SingleData<real_T>	Product3_lu;	/* '<S157>/Sum2' */
952	HardenedData :: SingleData<real_T>	Sum2_m;	/* '<S157>/Gain1' */
953	HardenedData :: SingleData<real_T>	Gain1_c;	/* '<S157>/Product5' */
954	HardenedData :: SingleData<real_T>	Product5_jx;	/* '<S157>/Sum' */
955	HardenedData :: SingleData<real_T>	Sum_g5;	/* '<S158>/Product' */
956	HardenedData :: SingleData<real_T>	Product_pe;	/* '<S158>/Product1' */
957	HardenedData :: SingleData<real_T>	Product1_mu;	/* '<S158>/Sum1' */
958	HardenedData :: SingleData<real_T>	Sum1_k;	/* '<S158>/Gain' */
959	HardenedData :: SingleData<real_T>	Gain_ay;	/* '<S158>/Product4' */
960	HardenedData :: SingleData<real_T>	Product4_ok;	/* '<S158>/Product6' */
961	HardenedData :: SingleData<real_T>	Product6_ld;	/* '<S158>/Product7' */
962	HardenedData :: SingleData<real_T>	Product7_ei;	/* '<S158>/Sum3' */
963	HardenedData :: SingleData<real_T>	Sum3_h;	/* '<S158>/Gain2' */
964	HardenedData :: SingleData<real_T>	Gain2_l2;	/* '<S158>/Product8' */
965	HardenedData :: SingleData<real_T>	Product8_d;	/* '<S158>/Product2' */
966	HardenedData :: SingleData<real_T>	Product2_ea;	/* '<S158>/Product3' */
967	HardenedData :: SingleData<real_T>	Product3_ll;	/* '<S158>/Sum2' */
968	HardenedData :: SingleData<real_T>	Sum2_mr;	/* '<S158>/Gain1' */
969	HardenedData :: SingleData<real_T>	Gain1_n;	/* '<S158>/Product5' */
970	HardenedData :: SingleData<real_T>	Product5_a;	/* '<S158>/Sum' */
971	HardenedData :: SingleData<real_T>	Sum_gz;	/* '<S159>/Product' */
972	HardenedData :: SingleData<real_T>	Product_am;	/* '<S159>/Product1' */
973	HardenedData :: SingleData<real_T>	Product1_bt;	/* '<S159>/Sum1' */
974	HardenedData :: SingleData<real_T>	Sum1_hk;	


```

975 HardenedData :: SingleData<real_T> Gain_g; /* '<S159>/Gain' */
976 HardenedData :: SingleData<real_T> Product4_o5; /* '<S159>/Product4' */
977 HardenedData :: SingleData<real_T> Product2_cc; /* '<S159>/Product2' */
978 HardenedData :: SingleData<real_T> Product3_cd; /* '<S159>/Product3' */
979 HardenedData :: SingleData<real_T> Sum2_o; /* '<S159>/Sum2' */
980 HardenedData :: SingleData<real_T> Gain1_ni; /* '<S159>/Gain1' */
981 HardenedData :: SingleData<real_T> Product5_n2; /* '<S159>/Product5' */
982 HardenedData :: SingleData<real_T> Product6_d5; /* '<S159>/Product6' */
983 HardenedData :: SingleData<real_T> Product7_em; /* '<S159>/Product7' */
984 HardenedData :: SingleData<real_T> Sum3_jh; /* '<S159>/Sum3' */
985 HardenedData :: SingleData<real_T> Gain2_ja; /* '<S159>/Gain2' */
986 HardenedData :: SingleData<real_T> Product8_j; /* '<S159>/Product8' */
987 HardenedData :: SingleData<real_T> Sum_pw; /* '<S159>/Sum' */
988 HardenedData :: SingleData<real_T> TmpSignalConversionAtDotProdu_j[3];
989 HardenedData :: SingleData<real_T> DotProduct3_ph; /* '<S178>/Dot Product3' */
990 HardenedData :: SingleData<real_T> DotProduct1_o; /* '<S178>/Dot Product1' */
991 HardenedData :: SingleData<real_T> DotProduct2_g; /* '<S178>/Dot Product2' */
992 HardenedData :: SingleData<real_T> Divide1_p; /* '<S178>/Divide1' */
993 HardenedData :: SingleData<real_T> Sqrt3_m; /* '<S178>/Sqrt3' */
994 HardenedData :: SingleData<real_T> Divide_c; /* '<S178>/Divide' */
995 HardenedData :: SingleData<real_T> Bias_h; /* '<S178>/Bias' */
996 HardenedData :: SingleData<real_T> Abs_m; /* '<S178>/Abs' */
997 HardenedData :: SingleData<real_T> Bias1_e; /* '<S178>/Bias1' */
998 HardenedData :: SingleData<real_T> Abs1_i; /* '<S178>/Abs1' */
999 HardenedData :: SingleData<real_T> Merge_l; /* '<S178>/Merge' */
1000 HardenedData :: SingleData<real_T> Switch_c[4]; /* '<S124>/Switch' */
1001 HardenedData :: SingleData<real_T> Product_h0; /* '<S273>/Product' */
1002 HardenedData :: SingleData<real_T> Product1_fm; /* '<S273>/Product1' */
1003 HardenedData :: SingleData<real_T> Product2_c2; /* '<S273>/Product2' */
1004 HardenedData :: SingleData<real_T> Product3_pq; /* '<S273>/Product3' */
1005 HardenedData :: SingleData<real_T> Sum_f; /* '<S273>/Sum' */
1006 HardenedData :: SingleData<real_T> sqrt_mo; /* '<S272>/sqrt' */
1007 HardenedData :: SingleData<real_T> Product_jc; /* '<S267>/Product' */
1008 HardenedData :: SingleData<real_T> Product1_o; /* '<S267>/Product1' */
1009 HardenedData :: SingleData<real_T> Product2_j; /* '<S267>/Product2' */
1010 HardenedData :: SingleData<real_T> Product3_lz; /* '<S267>/Product3' */
1011 HardenedData :: SingleData<real_T> fcn1_i; /* '<S128>/fcn1' */
1012 HardenedData :: SingleData<real_T> fcn2_f; /* '<S128>/fcn2' */
1013 HardenedData :: SingleData<real_T> fcn3_o; /* '<S128>/fcn3' */
1014 HardenedData :: SingleData<real_T> Merge_k; /* '<S268>/Merge' */
1015 HardenedData :: SingleData<real_T> fcn4_n; /* '<S128>/fcn4' */
1016 HardenedData :: SingleData<real_T> fcn5_g; /* '<S128>/fcn5' */
1017 HardenedData :: SingleData<real_T> VectorConcatenate_a[3]; /* '<S266>/Vector Concatenate' */
*/
1018 HardenedData :: SingleData<real_T> DerivativeGain_k[3]; /* '<S229>/Derivative Gain' */
1019 HardenedData :: SingleData<real_T> Tsamp_f[3]; /* '<S232>/Tsamp' */
1020 HardenedData :: SingleData<real_T> UD_l[3]; /* '<S230>/UD' */
1021 HardenedData :: SingleData<real_T> Diff_e[3]; /* '<S230>/Diff' */
1022 HardenedData :: SingleData<real_T> IntegralGain_p[3]; /* '<S234>/Integral Gain' */
1023 HardenedData :: SingleData<real_T> Integrator_k[3]; /* '<S237>/Integrator' */
1024 HardenedData :: SingleData<real_T> ProportionalGain_h[3]; /* '<S242>/Proportional Gain' */
*/
1025 HardenedData :: SingleData<real_T> is180degRot_l[3]; /* '<S137>/is 180deg Rot' */
1026 HardenedData :: SingleData<real_T> Product3_dz; /* '<S191>/Product3' */
1027 HardenedData :: SingleData<real_T> Product2_eq; /* '<S191>/Product2' */
1028 HardenedData :: SingleData<real_T> Product1_aa; /* '<S191>/Product1' */
1029 HardenedData :: SingleData<real_T> DotProduct2_n; /* '<S137>/Dot Product2' */
1030 HardenedData :: SingleData<real_T> DotProduct1_h; /* '<S137>/Dot Product1' */
1031 HardenedData :: SingleData<real_T> Divide1_h; /* '<S137>/Divide1' */
1032 HardenedData :: SingleData<real_T> Sqrt3_mq; /* '<S137>/Sqrt3' */
1033 HardenedData :: SingleData<real_T> DotProduct3_e; /* '<S137>/Dot Product3' */
1034 HardenedData :: SingleData<real_T> Add2_k; /* '<S137>/Add2' */
1035 HardenedData :: SingleData<real_T> Product_er; /* '<S191>/Product' */
1036 HardenedData :: SingleData<real_T> Sum_e; /* '<S191>/Sum' */
1037 HardenedData :: SingleData<real_T> sqrt_j; /* '<S190>/sqrt' */
1038 HardenedData :: SingleData<real_T> Product3_lzz; /* '<S181>/Product3' */
1039 HardenedData :: SingleData<real_T> Product3_ay; /* '<S146>/Product3' */
1040 HardenedData :: SingleData<real_T> Product2_ep; /* '<S181>/Product2' */
1041 HardenedData :: SingleData<real_T> Product2_bc; /* '<S146>/Product2' */
1042 HardenedData :: SingleData<real_T> Product1_ge; /* '<S181>/Product1' */
1043 HardenedData :: SingleData<real_T> Product1_fh; /* '<S146>/Product1' */
1044 HardenedData :: SingleData<real_T> Product_ey; /* '<S181>/Product' */
1045 HardenedData :: SingleData<real_T> Product_d0; /* '<S146>/Product' */
1046 HardenedData :: SingleData<real_T> Sum_ki; /* '<S146>/Sum' */
1047 HardenedData :: SingleData<real_T> Product3_aw; /* '<S147>/Product3' */
1048 HardenedData :: SingleData<real_T> Product2_pu; /* '<S147>/Product2' */
1049 HardenedData :: SingleData<real_T> Product1_b0; /* '<S147>/Product1' */
1050 HardenedData :: SingleData<real_T> Product_cq; /* '<S147>/Product' */
1051 HardenedData :: SingleData<real_T> Sum_oq; /* '<S147>/Sum' */
1052 HardenedData :: SingleData<real_T> Product3_h5; /* '<S148>/Product3' */
1053 HardenedData :: SingleData<real_T> Product2_i; /* '<S148>/Product2' */
1054 HardenedData :: SingleData<real_T> Product1_ai; /* '<S148>/Product1' */
1055 HardenedData :: SingleData<real_T> Product_cf; /* '<S148>/Product' */
1056 HardenedData :: SingleData<real_T> Sum_kc; /* '<S148>/Sum' */
1057 HardenedData :: SingleData<real_T> Product3_cl; /* '<S149>/Product3' */
1058 HardenedData :: SingleData<real_T> Product2_no; /* '<S149>/Product2' */
1059 HardenedData :: SingleData<real_T> Product1_d1; /* '<S149>/Product1' */
1060 HardenedData :: SingleData<real_T> Product_gr; /* '<S149>/Product' */
1061 HardenedData :: SingleData<real_T> Sum_n1; /* '<S149>/Sum' */
1062 HardenedData :: SingleData<real_T> Switch3[3]; /* '<S180>/Switch3' */
1063 HardenedData :: SingleData<real_T> jxi_m; /* '<S189>/j x i' */
1064 HardenedData :: SingleData<real_T> ixk_k; /* '<S189>/i x k' */
1065 HardenedData :: SingleData<real_T> kxj_f; /* '<S189>/k x j' */
1066 HardenedData :: SingleData<real_T> ixj_k; /* '<S188>/i x j' */
1067 HardenedData :: SingleData<real_T> kxi_bj; /* '<S188>/k x i' */
1068 HardenedData :: SingleData<real_T> jxk_l; /* '<S188>/j x k' */
1069 HardenedData :: SingleData<real_T> Sum_pd[3]; /* '<S187>/Sum' */
1070 HardenedData :: SingleData<real_T> Gain2_c; /* '<S180>/Gain2' */
1071 HardenedData :: SingleData<real_T> Gain_o2; /* '<S180>/Gain' */
1072 HardenedData :: SingleData<real_T> jxi_no; /* '<S183>/j x i' */
1073 HardenedData :: SingleData<real_T> ixk_b; /* '<S183>/i x k' */
1074 HardenedData :: SingleData<real_T> kxj_n; /* '<S183>/k x j' */
1075 HardenedData :: SingleData<real_T> ixj_l; /* '<S182>/i x j' */
1076 HardenedData :: SingleData<real_T> kxi_o; /* '<S182>/k x i' */

```

```

1077         HardenedData :: SingleData<real_T>      jxk_d;          /* '<S182>/j x k' */
1078         HardenedData :: SingleData<real_T>      Sum_b[3];        /* '<S177>/Sum' */
1079         HardenedData :: SingleData<real_T>      jxi_k;          /* '<S168>/j x i' */
1080         HardenedData :: SingleData<real_T>      ixk_a;          /* '<S168>/i x k' */
1081         HardenedData :: SingleData<real_T>      kxj_c;          /* '<S168>/k x j' */
1082         HardenedData :: SingleData<real_T>      ixj_o;          /* '<S167>/i x j' */
1083         HardenedData :: SingleData<real_T>      kxi_p;          /* '<S167>/k x i' */
1084         HardenedData :: SingleData<real_T>      jxk_er;         /* '<S167>/j x k' */
1085         HardenedData :: SingleData<real_T>      Sum_ff[3];       /* '<S162>/Sum' */
1086         HardenedData :: SingleData<real_T>      Dipole_x[3];     /* '<S2>/Magnetorquer control' */
1087         HardenedData :: SingleData<real_T>      Dipole_y[3];     /* '<S2>/Magnetorquer control' */
1088         HardenedData :: SingleData<real_T>      Dipole_z[3];     /* '<S2>/Magnetorquer control' */
1089         HardenedData :: SingleData<real_T>      bdot[3];         /* '<S2>/Derivative Function' */
1090 #endif
1091         boolean_T Compare;          /* '<S59>/Compare' */
1092         boolean_T Compare_h[9];     /* '<S57>/Compare' */
1093         boolean_T LogicalOperator1; /* '<S51>/Logical Operator1' */
1094         boolean_T Compare_k;        /* '<S131>/Compare' */
1095         boolean_T Compare_a;        /* '<S132>/Compare' */
1096         boolean_T OR;               /* '<S124>/OR' */
1097         boolean_T Compare_m;        /* '<S164>/Compare' */
1098         boolean_T Compare_g;        /* '<S179>/Compare' */
1099         boolean_T xz;               /* '<S180>/x>z' */
1100     } B_Vehicle_T;
1101
1102 /* Block states (default storage) for system '<Root>' */
1103 typedef struct {
1104 #ifdef RAD_HARD
1105     HardenedData :: TripleData<real_T> UD_DSTATE[3];          /* '<S86>/UD' */
1106     HardenedData :: TripleData<real_T> Integrator_DSTATE[3];  /* '<S93>/Integrator' */
1107     HardenedData :: TripleData<real_T> UD_DSTATE_d[3];        /* '<S230>/UD' */
1108     HardenedData :: TripleData<real_T> Integrator_DSTATE_f[3]; /* '<S237>/Integrator' */
1109     HardenedData :: TripleData<real_T> state[3];              /* '<S2>/Derivative Function' */
1110 #else
1111     HardenedData :: SingleData<real_T> UD_DSTATE[3];          /* '<S86>/UD' */
1112     HardenedData :: SingleData<real_T> Integrator_DSTATE[3];  /* '<S93>/Integrator' */
1113     HardenedData :: SingleData<real_T> UD_DSTATE_d[3];        /* '<S230>/UD' */
1114     HardenedData :: SingleData<real_T> Integrator_DSTATE_f[3]; /* '<S237>/Integrator' */
1115     HardenedData :: SingleData<real_T> state[3];              /* '<S2>/Derivative Function' */
1116 #endif
1117 } DW_Vehicle_T;
1118
1119 /* Invariant block signals (default storage) */
1120 typedef struct {
1121 #ifdef RAD_HARD
1122     HardenedData :: TripleData<real_T>      DotProduct3;      /* '<S129>/Dot Product3' */
1123     HardenedData :: TripleData<real_T>      DotProduct1;      /* '<S129>/Dot Product1' */
1124     HardenedData :: TripleData<real_T>      DotProduct2;      /* '<S129>/Dot Product2' */
1125     HardenedData :: TripleData<real_T>      Divide1;          /* '<S129>/Divide1' */
1126     HardenedData :: TripleData<real_T>      Sqrt3;            /* '<S129>/Sqrt3' */
1127     HardenedData :: TripleData<real_T>      Divide;           /* '<S129>/Divide' */
1128     HardenedData :: TripleData<real_T>      Bias;             /* '<S129>/Bias' */
1129     HardenedData :: TripleData<real_T>      Abs;              /* '<S129>/Abs' */
1130     HardenedData :: TripleData<real_T>      Bias1;            /* '<S129>/Bias1' */
1131     HardenedData :: TripleData<real_T>      Abs1;             /* '<S129>/Abs1' */
1132     HardenedData :: TripleData<real_T>      DotProduct1_g;     /* '<S163>/Dot Product1' */
1133     HardenedData :: TripleData<real_T>      DotProduct1_c;     /* '<S136>/Dot Product1' */
1134     HardenedData :: TripleData<real_T>      Gain;              /* '<S165>/Gain' */
1135     HardenedData :: TripleData<real_T>      Gain2;             /* '<S165>/Gain2' */
1136     HardenedData :: TripleData<real_T>      Switch3[3];        /* '<S165>/Switch3' */
1137     HardenedData :: TripleData<real_T>      ixj;              /* '<S173>/i x j' */
1138     HardenedData :: TripleData<real_T>      jxk;              /* '<S173>/j x k' */
1139     HardenedData :: TripleData<real_T>      kxi;              /* '<S173>/k x i' */
1140     HardenedData :: TripleData<real_T>      ixk;              /* '<S174>/i x k' */
1141     HardenedData :: TripleData<real_T>      jxi;              /* '<S174>/j x i' */
1142     HardenedData :: TripleData<real_T>      kxj;              /* '<S174>/k x j' */
1143     HardenedData :: TripleData<real_T>      Sum[3];            /* '<S172>/Sum' */
1144     HardenedData :: TripleData<real_T>      ixj_g;            /* '<S194>/i x j' */
1145     HardenedData :: TripleData<real_T>      jxk_e;            /* '<S194>/j x k' */
1146     HardenedData :: TripleData<real_T>      kxi_c;            /* '<S194>/k x i' */
1147     HardenedData :: TripleData<real_T>      ixk_d;            /* '<S195>/i x k' */
1148     HardenedData :: TripleData<real_T>      jxi_e;            /* '<S195>/j x i' */
1149     HardenedData :: TripleData<real_T>      kxj_o;            /* '<S195>/k x j' */
1150     HardenedData :: TripleData<real_T>      Sum_d[3];          /* '<S192>/Sum' */
1151     HardenedData :: TripleData<real_T>      ixj_a;            /* '<S196>/i x j' */
1152     HardenedData :: TripleData<real_T>      jxk_n;            /* '<S196>/j x k' */
1153     HardenedData :: TripleData<real_T>      kxi_p;            /* '<S196>/k x i' */
1154     HardenedData :: TripleData<real_T>      ixk_c;            /* '<S197>/i x k' */
1155     HardenedData :: TripleData<real_T>      jxi_o;            /* '<S197>/j x i' */
1156     HardenedData :: TripleData<real_T>      kxj_n;            /* '<S197>/k x j' */
1157     HardenedData :: TripleData<real_T>      Sum_d0[3];         /* '<S193>/Sum' */
1158 #else
1159     HardenedData :: SingleData<real_T>      DotProduct3;      /* '<S129>/Dot Product3' */
1160     HardenedData :: SingleData<real_T>      DotProduct1;      /* '<S129>/Dot Product1' */
1161     HardenedData :: SingleData<real_T>      DotProduct2;      /* '<S129>/Dot Product2' */
1162     HardenedData :: SingleData<real_T>      Divide1;          /* '<S129>/Divide1' */
1163     HardenedData :: SingleData<real_T>      Sqrt3;            /* '<S129>/Sqrt3' */
1164     HardenedData :: SingleData<real_T>      Divide;           /* '<S129>/Divide' */
1165     HardenedData :: SingleData<real_T>      Bias;             /* '<S129>/Bias' */
1166     HardenedData :: SingleData<real_T>      Abs;              /* '<S129>/Abs' */
1167     HardenedData :: SingleData<real_T>      Bias1;            /* '<S129>/Bias1' */
1168     HardenedData :: SingleData<real_T>      Abs1;             /* '<S129>/Abs1' */
1169     HardenedData :: SingleData<real_T>      DotProduct1_g;     /* '<S163>/Dot Product1' */
1170     HardenedData :: SingleData<real_T>      DotProduct1_c;     /* '<S136>/Dot Product1' */
1171     HardenedData :: SingleData<real_T>      Gain;              /* '<S165>/Gain' */
1172     HardenedData :: SingleData<real_T>      Gain2;            /* '<S165>/Gain2' */
1173     HardenedData :: SingleData<real_T>      Switch3[3];        /* '<S165>/Switch3' */
1174     HardenedData :: SingleData<real_T>      ixj;              /* '<S173>/i x j' */
1175     HardenedData :: SingleData<real_T>      jxk;              /* '<S173>/j x k' */
1176     HardenedData :: SingleData<real_T>      kxi;              /* '<S173>/k x i' */
1177     HardenedData :: SingleData<real_T>      ixk;              /* '<S174>/i x k' */
1178     HardenedData :: SingleData<real_T>      jxi;              /* '<S174>/j x i' */
1179     HardenedData :: SingleData<real_T>      kxj;              /* '<S174>/k x j' */
1180     HardenedData :: SingleData<real_T>      Sum[3];            /* '<S172>/Sum' */

```

```

1181         HardenedData::SingleData<real_T>    ixj_g;          /* '<S194>/i x j' */
1182         HardenedData::SingleData<real_T>    jxk_e;          /* '<S194>/j x k' */
1183         HardenedData::SingleData<real_T>    kxi_c;          /* '<S194>/k x i' */
1184         HardenedData::SingleData<real_T>    ixk_d;          /* '<S195>/i x k' */
1185         HardenedData::SingleData<real_T>    jxi_e;          /* '<S195>/j x i' */
1186         HardenedData::SingleData<real_T>    kxj_o;          /* '<S195>/k x j' */
1187         HardenedData::SingleData<real_T>    Sum_d[3];        /* '<S192>/Sum' */
1188         HardenedData::SingleData<real_T>    ixj_a;          /* '<S196>/i x j' */
1189         HardenedData::SingleData<real_T>    jxk_n;          /* '<S196>/j x k' */
1190         HardenedData::SingleData<real_T>    kxi_p;          /* '<S196>/k x i' */
1191         HardenedData::SingleData<real_T>    ixk_c;          /* '<S197>/i x k' */
1192         HardenedData::SingleData<real_T>    jxi_o;          /* '<S197>/j x i' */
1193         HardenedData::SingleData<real_T>    kxj_n;          /* '<S197>/k x j' */
1194         HardenedData::SingleData<real_T>    Sum_d0[3];       /* '<S193>/Sum' */
1195     #endif
1196     boolean_T xz;          /* '<S165>/x>z' */
1197 } ConstB_Vehicle_T;
1198
1199 /* External inputs (root inport signals with default storage) */
1200 typedef struct {
1201     #ifdef RAD_HARD
1202         HardenedData::TripleData<real_T>    Bmgm[3];
1203         HardenedData::TripleData<real_T>    q_eci2b[4];          /* '<Root>/q_eci2b' */
1204         HardenedData::TripleData<real_T>    xSunECI[3];          /* '<Root>/xSunECI' */
1205         HardenedData::TripleData<real_T>    qECECF2b[4];          /* '<Root>/qECECF2b' */
1206         HardenedData::TripleData<real_T>    vECECF[3];           /* '<Root>/vECECF' */
1207         HardenedData::TripleData<real_T>    xECECF[3];           /* '<Root>/xECECF' */
1208         HardenedData::TripleData<real_T>    Mode;                /* '<Root>/Mode' */
1209     #else
1210         HardenedData::SingleData<real_T>    Bmgm[3];
1211         HardenedData::SingleData<real_T>    q_eci2b[4];          /* '<Root>/q_eci2b' */
1212         HardenedData::SingleData<real_T>    xSunECI[3];          /* '<Root>/xSunECI' */
1213         HardenedData::SingleData<real_T>    qECECF2b[4];          /* '<Root>/qECECF2b' */
1214         HardenedData::SingleData<real_T>    vECECF[3];           /* '<Root>/vECECF' */
1215         HardenedData::SingleData<real_T>    xECECF[3];           /* '<Root>/xECECF' */
1216         HardenedData::SingleData<real_T>    Mode;                /* '<Root>/Mode' */
1217     #endif
1218 } ExtU_Vehicle_T;
1219
1220 /* External outputs (root outports fed by signals with default storage) */
1221 typedef struct {
1222     #ifdef RAD_HARD
1223         HardenedData::TripleData<real_T>    Commands[3];          /* '<Root>/Commands' */
1224         HardenedData::TripleData<real_T>    Bbody[3];             /* '<Root>/Bbody' */
1225     #else
1226         HardenedData::SingleData<real_T>    Commands[3];          /* '<Root>/Commands' */
1227         HardenedData::SingleData<real_T>    Bbody[3];             /* '<Root>/Bbody' */
1228     #endif
1229 } ExtY_Vehicle_T;
1230
1231
1232
1233 /* Block signals (default storage) */
1234 #ifdef __cplusplus
1235
1236 extern "C" {
1237
1238 #endif
1239
1240
1241
1242 #ifdef __cplusplus
1243 }
1244 #endif
1245
1246
1247 #ifdef __cplusplus
1248
1249 extern "C" {
1250
1251 #endif
1252
1253
1254
1255 #ifdef __cplusplus
1256 }
1257 #endif
1258
1259
1260 //extern const ConstB_Vehicle_T Vehicle_ConstB; /* constant block i/o */
1261
1262 #ifdef __cplusplus
1263
1264 extern "C" {
1265
1266 #endif
1267
1268
1269 /* Model entry point functions */
1270 extern void Vehicle_initialize();
1271 extern void Vehicle_step(void);
1272 extern void Vehicle_terminate(void);
1273 extern void FlightSoftware(const double B_mgm[3], const double q_eci2b[4], const double xSunECI[3],
1274                             const double qECECF2b[4], const double vECECF[3], const double xECECF[3],
1275                             const double mode, double Commands[3], double Bbody[3], unsigned int RadiationNoiseFlag, unsigned int
1276                             NumberOfBitFlipEvents, double AttitudeError[3]);
1277
1278
1279 extern real_T rt_atan2d_snf(real_T u0, real_T u1);
1280
1281 extern void Detumble();
1282 extern void NadirPointing();
1283 extern void SunPointing();

```

```

1284     extern void SEU();
1285
1286     /* Block signals (default storage) */
1287     extern B_Vehicle_T Vehicle_B;
1288
1289     /* Block states (default storage) */
1290     extern DW_Vehicle_T Vehicle_DW;
1291
1292     /* External inputs (root inport signals with default storage) */
1293     extern ExtU_Vehicle_T Vehicle_U;
1294
1295     /* External outputs (root outports fed by signals with default storage) */
1296     extern ExtY_Vehicle_T Vehicle_Y;
1297
1298     /*Constants*/
1299     extern ConstB_Vehicle_T Vehicle_ConstB;
1300
1301
1302
1303     #ifdef __cplusplus
1304     }
1305     #endif
1306
1307     /* Real-time Model object */
1308     #ifdef __cplusplus
1309     extern "C" {
1310
1311     #endif
1312
1313     // extern RT_MODEL_Vehicle_T *const Vehicle_M;
1314
1315     #ifdef __cplusplus
1316     }
1317     #endif
1318
1319     /* RTW_HEADER_Vehicle_h_ */
1320
1321
1322

```

RadiationEffects.cpp

```

1  #include "Vehicle.h"
2
3  enum {
4      Inputs_Noise=1,
5      Outputs_Noise ,
6      States_Noise ,
7      Consts_Noise ,
8      BlockSignals_Noise
9  };
10
11  /*Local Functions Declaration*/
12  void Inputs_Corruption(int index);
13  void Outputs_Corruption(int index);
14  void States_Corruption(int index);
15  void Consts_Corruption(int index);
16  void BlockSignals_Corruption(int index);
17
18  /*Single Event Upset function which is called from FlightSoftware*/
19  void SEU()
20  {
21      int selection;
22      int index;
23
24
25      selection = ( rand() % 4) + 1;  //1-5
26
27
28      switch (selection)
29      {
30      case Inputs_Noise:
31          index = (rand() % 20) +1 ;  //1 - 21
32          Inputs_Corruption(index);
33          break;
34      case Outputs_Noise:
35          index = (rand() % 5) +1 ;  //1-6
36          Outputs_Corruption(index);
37          break;
38      case States_Noise:
39          index = (rand() % 14) + 1 ;  //1-15
40          States_Corruption(index);
41          break;
42      case Consts_Noise:
43          index = (rand() % 43) + 1;  //1-44
44          Consts_Corruption(index);
45          break;
46      case BlockSignals_Noise:
47          index = (rand() % 613)+1 ;  //1-614
48          BlockSignals_Corruption(index);
49          break;
50      }
51
52  }
53
54
55  /*Inputs Corruption function*/
56  void Inputs_Corruption(int index)
57  {
58      switch (index)
59      {
60          /*Inputs corruption*/

```

```

61     case 1:      Vehicle_U.Bmgm[0].a()    = (real_T)index; break; //random number
62     case 2:      Vehicle_U.Bmgm[1].a()    = (real_T)index; break;
63     case 3:      Vehicle_U.Bmgm[2].a()    = (real_T)index; break;
64     case 4:      Vehicle_U.Mode.a()       = (real_T) index; break;
65     case 5:      Vehicle_U.qECEF2b[0].a() = (real_T) index; break;
66     case 6:      Vehicle_U.qECEF2b[1].a() = (real_T) index; break;
67     case 7:      Vehicle_U.qECEF2b[2].a() = (real_T) index; break;
68     case 8:      Vehicle_U.qECEF2b[3].a() = (real_T) index; break;
69     case 9:      Vehicle_U.q_eci2b[0].a() = (real_T) index; break;
70     case 10:     Vehicle_U.q_eci2b[1].a() = (real_T) index; break;
71     case 11:     Vehicle_U.q_eci2b[2].a() = (real_T) index; break;
72     case 12:     Vehicle_U.q_eci2b[3].a() = (real_T) index; break;
73     case 13:     Vehicle_U.vECEf[0].a()   = (real_T) index; break;
74     case 14:     Vehicle_U.vECEf[1].a()   = (real_T) index; break;
75     case 15:     Vehicle_U.vECEf[2].a()   = (real_T) index; break;
76     case 16:     Vehicle_U.xECEf[0].a()   = (real_T) index; break;
77     case 17:     Vehicle_U.xECEf[1].a()   = (real_T) index; break;
78     case 18:     Vehicle_U.xECEf[2].a()   = (real_T) index; break;
79     case 19:     Vehicle_U.xSunECI[0].a() = (real_T) index; break;
80     case 20:     Vehicle_U.xSunECI[1].a() = (real_T) index; break;
81     case 21:     Vehicle_U.xSunECI[2].a() = (real_T) index; break;
82     }
83 }
84
85 /*Outputs Corruption function*/
86 void Outputs_Corruption(int index)
87 {
88     switch (index)
89     {
90         /*Outputs corruption*/
91         case 1:      Vehicle_Y.Bbody[0].a()    = (real_T)index; break;
92         case 2:      Vehicle_Y.Bbody[1].a()    = (real_T)index; break;
93         case 3:      Vehicle_Y.Bbody[2].a()    = (real_T)index; break;
94         case 4:      Vehicle_Y.Commands[0].a() = (real_T)index; break;
95         case 5:      Vehicle_Y.Commands[1].a() = (real_T)index; break;
96         case 6:      Vehicle_Y.Commands[2].a() = (real_T)index; break;
97     }
98 }
99
100
101 /*States Corruption function*/
102 void States_Corruption(int index)
103 {
104     switch (index)
105     {
106         /*states corruption*/
107         case 1:      Vehicle_DW.state[0].a()    = (real_T)index; break;
108         case 2:      Vehicle_DW.state[1].a()    = (real_T)index; break;
109         case 3:      Vehicle_DW.state[2].a()    = (real_T)index; break;
110         case 4:      Vehicle_DW.Integrator_DSTATE[0].a() = (real_T)index; break;
111         case 5:      Vehicle_DW.Integrator_DSTATE[1].a() = (real_T)index; break;
112         case 6:      Vehicle_DW.Integrator_DSTATE[2].a() = (real_T)index; break;
113         case 7:      Vehicle_DW.Integrator_DSTATE_f[0].a() = (real_T)index; break;
114         case 8:      Vehicle_DW.Integrator_DSTATE_f[1].a() = (real_T)index; break;
115         case 9:      Vehicle_DW.Integrator_DSTATE_f[2].a() = (real_T)index; break;
116         case 10:     Vehicle_DW.UD_DSTATE[0].a() = (real_T)index; break;
117         case 11:     Vehicle_DW.UD_DSTATE[1].a() = (real_T)index; break;
118         case 12:     Vehicle_DW.UD_DSTATE[2].a() = (real_T)index; break;
119         case 13:     Vehicle_DW.UD_DSTATE_d[0].a() = (real_T)index; break;
120         case 14:     Vehicle_DW.UD_DSTATE_d[1].a() = (real_T)index; break;
121         case 15:     Vehicle_DW.UD_DSTATE_d[2].a() = (real_T)index; break;
122     }
123 }
124
125
126 /*Constants Corruption function*/
127 void Consts_Corruption(int index)
128 {
129     switch (index)
130     {
131         /*states corruption*/
132         case 1:      Vehicle_ConstB.Abs.a()    = (real_T)index; break;
133         case 2:      Vehicle_ConstB.Abs1.a()   = (real_T)index; break;
134         case 3:      Vehicle_ConstB.Bias.a()    = (real_T)index; break;
135         case 4:      Vehicle_ConstB.Bias1.a()   = (real_T)index; break;
136         case 5:      Vehicle_ConstB.Divide.a()  = (real_T)index; break;
137         case 6:      Vehicle_ConstB.Divide1.a() = (real_T)index; break;
138         case 7:      Vehicle_ConstB.DotProduct1.a() = (real_T)index; break;
139         case 8:      Vehicle_ConstB.DotProduct1_c.a() = (real_T)index; break;
140         case 9:      Vehicle_ConstB.DotProduct1_g.a() = (real_T)index; break;
141         case 10:     Vehicle_ConstB.DotProduct2.a() = (real_T)index; break;
142         case 11:     Vehicle_ConstB.DotProduct3.a() = (real_T)index; break;
143         case 12:     Vehicle_ConstB.Gain.a()    = (real_T)index; break;
144         case 13:     Vehicle_ConstB.Gain2.a()   = (real_T)index; break;
145         case 14:     Vehicle_ConstB.Sqrt3.a()   = (real_T)index; break;
146         case 15:     Vehicle_ConstB.Sum[0].a()  = (real_T)index; break;
147         case 16:     Vehicle_ConstB.Sum[1].a()  = (real_T)index; break;
148         case 17:     Vehicle_ConstB.Sum[2].a()  = (real_T)index; break;
149         case 18:     Vehicle_ConstB.Sum_d[0].a() = (real_T)index; break;
150         case 19:     Vehicle_ConstB.Sum_d[1].a() = (real_T)index; break;
151         case 20:     Vehicle_ConstB.Sum_d[2].a() = (real_T)index; break;
152         case 21:     Vehicle_ConstB.Sum_d0[0].a() = (real_T)index; break;
153         case 22:     Vehicle_ConstB.Sum_d0[1].a() = (real_T)index; break;
154         case 23:     Vehicle_ConstB.Sum_d0[2].a() = (real_T)index; break;
155         case 24:     Vehicle_ConstB.Switch3[0].a() = (real_T)index; break;
156         case 25:     Vehicle_ConstB.Switch3[1].a() = (real_T)index; break;
157         case 26:     Vehicle_ConstB.Switch3[2].a() = (real_T)index; break;
158         case 27:     Vehicle_ConstB.ixj.a()      = (real_T)index; break;
159         case 28:     Vehicle_ConstB.ixj_a.a()    = (real_T)index; break;
160         case 29:     Vehicle_ConstB.ixj_g.a()    = (real_T)index; break;
161         case 30:     Vehicle_ConstB.ixk.a()      = (real_T)index; break;
162         case 31:     Vehicle_ConstB.ixk_c.a()    = (real_T)index; break;
163         case 32:     Vehicle_ConstB.ixk_d.a()    = (real_T)index; break;
164         case 33:     Vehicle_ConstB.jxi.a()      = (real_T)index; break;

```



```

165     case 34:      Vehicle_ConstB.jxi_e.a()      = (real_T)index; break;
166     case 35:      Vehicle_ConstB.jxi_o.a()      = (real_T)index; break;
167     case 36:      Vehicle_ConstB.jxk.a()         = (real_T)index; break;
168     case 37:      Vehicle_ConstB.jxk_e.a()       = (real_T)index; break;
169     case 38:      Vehicle_ConstB.jxk_n.a()       = (real_T)index; break;
170     case 39:      Vehicle_ConstB.kxi.a()         = (real_T)index; break;
171     case 40:      Vehicle_ConstB.kxi_c.a()       = (real_T)index; break;
172     case 41:      Vehicle_ConstB.kxi_p.a()       = (real_T)index; break;
173     case 42:      Vehicle_ConstB.kxj.a()         = (real_T)index; break;
174     case 43:      Vehicle_ConstB.kxj_n.a()       = (real_T)index; break;
175     case 44:      Vehicle_ConstB.kxj_o.a()       = (real_T)index; break;
176     //case 45:    Vehicle_ConstB.xz.a()          = (real_T)index; break;
177 }
178 }
179
180
181 /*Block Signals Corruption function*/
182 void BlockSignals_Corruption(int index)
183 {
184     switch (index)
185     {
186         /*Inputs corruption*/
187         case 1:      Vehicle_B.Abs.a()            = (real_T)index; break;
188         case 2:      Vehicle_B.Abs1.a()           = (real_T)index; break;
189         case 3:      Vehicle_B.Abs1_d.a()         = (real_T)index; break;
190         case 4:      Vehicle_B.Abs1_i.a()         = (real_T)index; break;
191         case 5:      Vehicle_B.Abs1_l.a()         = (real_T)index; break;
192         case 6:      Vehicle_B.Abs2[0].a()        = (real_T)index; break;
193         case 7:      Vehicle_B.Abs2[1].a()        = (real_T)index; break;
194         case 8:      Vehicle_B.Abs2[2].a()        = (real_T)index; break;
195         case 9:      Vehicle_B.Abs2[3].a()        = (real_T)index; break;
196         case 10:     Vehicle_B.Abs2[4].a()        = (real_T)index; break;
197         case 11:     Vehicle_B.Abs2[5].a()        = (real_T)index; break;
198         case 12:     Vehicle_B.Abs2[6].a()        = (real_T)index; break;
199         case 13:     Vehicle_B.Abs2[7].a()        = (real_T)index; break;
200         case 14:     Vehicle_B.Abs2[8].a()        = (real_T)index; break;
201         case 15:     Vehicle_B.Abs_m.a()          = (real_T)index; break;
202         case 16:     Vehicle_B.Abs_o.a()          = (real_T)index; break;
203         case 17:     Vehicle_B.Add.a()            = (real_T)index; break;
204         case 18:     Vehicle_B.Add2.a()           = (real_T)index; break;
205         case 19:     Vehicle_B.Add2_k.a()         = (real_T)index; break;
206         case 20:     Vehicle_B.Add_a.a()          = (real_T)index; break;
207         case 21:     Vehicle_B.Add_d.a()          = (real_T)index; break;
208         case 22:     Vehicle_B.Add_e.a()          = (real_T)index; break;
209         case 23:     Vehicle_B.Add_ek.a()         = (real_T)index; break;
210         case 24:     Vehicle_B.Add_el.a()         = (real_T)index; break;
211         case 25:     Vehicle_B.Add_f.a()          = (real_T)index; break;
212         case 26:     Vehicle_B.Add_fc.a()         = (real_T)index; break;
213         case 27:     Vehicle_B.Add_g.a()          = (real_T)index; break;
214         case 28:     Vehicle_B.Add_gw.a()         = (real_T)index; break;
215         case 29:     Vehicle_B.Add_k.a()          = (real_T)index; break;
216         case 30:     Vehicle_B.Add_l.a()          = (real_T)index; break;
217         case 31:     Vehicle_B.Add_l2.a()         = (real_T)index; break;
218         case 32:     Vehicle_B.Add_m.a()          = (real_T)index; break;
219         case 33:     Vehicle_B.Add_o.a()          = (real_T)index; break;
220         case 34:     Vehicle_B.Add_ov.a()         = (real_T)index; break;
221
222         case 35:     Vehicle_B.Bias.a()           = (real_T)index; break;
223         case 36:     Vehicle_B.Bias1[0].a()       = (real_T)index; break;
224         case 37:     Vehicle_B.Bias1[1].a()       = (real_T)index; break;
225         case 38:     Vehicle_B.Bias1[2].a()       = (real_T)index; break;
226         case 39:     Vehicle_B.Bias1[3].a()       = (real_T)index; break;
227         case 40:     Vehicle_B.Bias1[4].a()       = (real_T)index; break;
228         case 41:     Vehicle_B.Bias1[5].a()       = (real_T)index; break;
229         case 42:     Vehicle_B.Bias1[6].a()       = (real_T)index; break;
230         case 43:     Vehicle_B.Bias1[7].a()       = (real_T)index; break;
231         case 44:     Vehicle_B.Bias1[8].a()       = (real_T)index; break;
232         case 45:     Vehicle_B.Bias1_e.a()        = (real_T)index; break;
233         case 46:     Vehicle_B.Bias1_j.a()        = (real_T)index; break;
234         case 47:     Vehicle_B.Bias1_m.a()        = (real_T)index; break;
235         case 48:     Vehicle_B.Bias_b.a()         = (real_T)index; break;
236         case 49:     Vehicle_B.Bias_h.a()         = (real_T)index; break;
237         case 50:     Vehicle_B.Bias_k.a()         = (real_T)index; break;
238         case 51:     Vehicle_B.bdot[0].a()        = (real_T)index; break;
239         case 52:     Vehicle_B.bdot[1].a()        = (real_T)index; break;
240         case 53:     Vehicle_B.bdot[2].a()        = (real_T)index; break;
241
242         case 54:     Vehicle_B.DerivativeGain[0].a() = (real_T)index; break;
243         case 55:     Vehicle_B.DerivativeGain[1].a() = (real_T)index; break;
244         case 56:     Vehicle_B.DerivativeGain[2].a() = (real_T)index; break;
245         case 57:     Vehicle_B.DerivativeGain_k[0].a() = (real_T)index; break;
246         case 58:     Vehicle_B.DerivativeGain_k[1].a() = (real_T)index; break;
247         case 59:     Vehicle_B.DerivativeGain_k[2].a() = (real_T)index; break;
248         case 60:     Vehicle_B.Diff[0].a()         = (real_T)index; break;
249         case 61:     Vehicle_B.Diff[1].a()         = (real_T)index; break;
250         case 62:     Vehicle_B.Diff[2].a()         = (real_T)index; break;
251         case 63:     Vehicle_B.Diff_e[0].a()       = (real_T)index; break;
252         case 64:     Vehicle_B.Diff_e[1].a()       = (real_T)index; break;
253         case 65:     Vehicle_B.Diff_e[2].a()       = (real_T)index; break;
254         case 66:     Vehicle_B.Dipole_x[0].a()     = (real_T)index; break;
255         case 67:     Vehicle_B.Dipole_x[1].a()     = (real_T)index; break;
256         case 68:     Vehicle_B.Dipole_x[2].a()     = (real_T)index; break;
257         case 69:     Vehicle_B.Dipole_y[0].a()     = (real_T)index; break;
258         case 70:     Vehicle_B.Dipole_y[1].a()     = (real_T)index; break;
259         case 71:     Vehicle_B.Dipole_y[2].a()     = (real_T)index; break;
260         case 613:    Vehicle_B.Dipole_z[0].a()     = (real_T)index; break;
261         case 72:     Vehicle_B.Dipole_z[1].a()     = (real_T)index; break;
262         case 73:     Vehicle_B.Dipole_z[2].a()     = (real_T)index; break;
263         case 74:     Vehicle_B.Divide.a()          = (real_T)index; break;
264         case 75:     Vehicle_B.Divide1.a()         = (real_T)index; break;
265         case 76:     Vehicle_B.Divide1_b.a()       = (real_T)index; break;
266         case 77:     Vehicle_B.Divide1_d.a()       = (real_T)index; break;
267         case 78:     Vehicle_B.Divide1_h.a()       = (real_T)index; break;
268         case 79:     Vehicle_B.Divide1_k.a()       = (real_T)index; break;

```

```

269 case 80: Vehicle_B.Divide1_p.a() = (real_T)index; break;
270 case 81: Vehicle_B.Divide2.a() = (real_T)index; break;
271 case 82: Vehicle_B.Divide3.a() = (real_T)index; break;
272 case 83: Vehicle_B.Divide_c.a() = (real_T)index; break;
273 case 84: Vehicle_B.Divide_k.a() = (real_T)index; break;
274 case 85: Vehicle_B.Divide_o.a() = (real_T)index; break;
275 case 86: Vehicle_B.DotProduct.a() = (real_T)index; break;
276 case 87: Vehicle_B.DotProduct1.a() = (real_T)index; break;
277 case 88: Vehicle_B.DotProduct1_h.a() = (real_T)index; break;
278 case 89: Vehicle_B.DotProduct1_o.a() = (real_T)index; break;
279 case 90: Vehicle_B.DotProduct2.a() = (real_T)index; break;
280 case 91: Vehicle_B.DotProduct2_c.a() = (real_T)index; break;
281 case 92: Vehicle_B.DotProduct2_f.a() = (real_T)index; break;
282 case 93: Vehicle_B.DotProduct2_g.a() = (real_T)index; break;
283 case 94: Vehicle_B.DotProduct2_n.a() = (real_T)index; break;
284 case 95: Vehicle_B.DotProduct3.a() = (real_T)index; break;
285 case 96: Vehicle_B.DotProduct3_e.a() = (real_T)index; break;
286 case 97: Vehicle_B.DotProduct3_o.a() = (real_T)index; break;
287 case 98: Vehicle_B.DotProduct3_p.a() = (real_T)index; break;
288 case 99: Vehicle_B.DotProduct3_ph.a() = (real_T)index; break;
289 case 100: Vehicle_B.DotProduct_b.a() = (real_T)index; break;
290 case 101: Vehicle_B.DotProduct_p.a() = (real_T)index; break;
291
292 case 102: Vehicle_B.ECEFToORF[0].a() = (real_T)index; break;
293 case 103: Vehicle_B.ECEFToORF[1].a() = (real_T)index; break;
294 case 104: Vehicle_B.ECEFToORF[2].a() = (real_T)index; break;
295 case 105: Vehicle_B.ECEFToORF[3].a() = (real_T)index; break;
296 case 106: Vehicle_B.ECEFToORF[4].a() = (real_T)index; break;
297 case 107: Vehicle_B.ECEFToORF[5].a() = (real_T)index; break;
298 case 108: Vehicle_B.ECEFToORF[6].a() = (real_T)index; break;
299 case 109: Vehicle_B.ECEFToORF[7].a() = (real_T)index; break;
300 case 110: Vehicle_B.ECEFToORF[8].a() = (real_T)index; break;
301
302 case 111: Vehicle_B.Gain.a() = (real_T)index; break;
303 case 112: Vehicle_B.Gain1.a() = (real_T)index; break;
304 case 113: Vehicle_B.Gain1_c.a() = (real_T)index; break;
305 case 114: Vehicle_B.Gain1_d.a() = (real_T)index; break;
306 case 115: Vehicle_B.Gain1_e.a() = (real_T)index; break;
307 case 116: Vehicle_B.Gain1_f.a() = (real_T)index; break;
308 case 117: Vehicle_B.Gain1_g.a() = (real_T)index; break;
309 case 118: Vehicle_B.Gain1_i.a() = (real_T)index; break;
310 case 119: Vehicle_B.Gain1_l.a() = (real_T)index; break;
311 case 120: Vehicle_B.Gain1_n.a() = (real_T)index; break;
312 case 121: Vehicle_B.Gain1_ni.a() = (real_T)index; break;
313 case 122: Vehicle_B.Gain1_o.a() = (real_T)index; break;
314 case 123: Vehicle_B.Gain1_oi.a() = (real_T)index; break;
315 case 124: Vehicle_B.Gain1_ox.a() = (real_T)index; break;
316 case 125: Vehicle_B.Gain2.a() = (real_T)index; break;
317 case 126: Vehicle_B.Gain2_a.a() = (real_T)index; break;
318 case 127: Vehicle_B.Gain2_b.a() = (real_T)index; break;
319 case 128: Vehicle_B.Gain2_c.a() = (real_T)index; break;
320 case 129: Vehicle_B.Gain2_f.a() = (real_T)index; break;
321 case 130: Vehicle_B.Gain2_j.a() = (real_T)index; break;
322 case 131: Vehicle_B.Gain2_ja.a() = (real_T)index; break;
323 case 132: Vehicle_B.Gain2_l.a() = (real_T)index; break;
324 case 133: Vehicle_B.Gain2_l2.a() = (real_T)index; break;
325 case 134: Vehicle_B.Gain2_n.a() = (real_T)index; break;
326 case 135: Vehicle_B.Gain2_o.a() = (real_T)index; break;
327 case 136: Vehicle_B.Gain2_om.a() = (real_T)index; break;
328 case 137: Vehicle_B.Gain2_p.a() = (real_T)index; break;
329 case 138: Vehicle_B.Gain_o2.a() = (real_T)index; break;
330 case 139: Vehicle_B.Gain_a.a() = (real_T)index; break;
331 case 140: Vehicle_B.Gain_ay.a() = (real_T)index; break;
332 case 141: Vehicle_B.Gain_b.a() = (real_T)index; break;
333 case 142: Vehicle_B.Gain_g.a() = (real_T)index; break;
334 case 143: Vehicle_B.Gain_i.a() = (real_T)index; break;
335 case 144: Vehicle_B.Gain_j.a() = (real_T)index; break;
336 case 145: Vehicle_B.Gain_k.a() = (real_T)index; break;
337 case 146: Vehicle_B.Gain_kp.a() = (real_T)index; break;
338 case 147: Vehicle_B.Gain_l.a() = (real_T)index; break;
339 case 148: Vehicle_B.Gain_o.a() = (real_T)index; break;
340 case 149: Vehicle_B.Gain_o2.a() = (real_T)index; break;
341 case 150: Vehicle_B.Gain_p.a() = (real_T)index; break;
342
343 case 151: Vehicle_B.IntegralGain[0].a() = (real_T)index; break;
344 case 152: Vehicle_B.IntegralGain[1].a() = (real_T)index; break;
345 case 153: Vehicle_B.IntegralGain[2].a() = (real_T)index; break;
346 case 154: Vehicle_B.IntegralGain_p[0].a() = (real_T)index; break;
347 case 155: Vehicle_B.IntegralGain_p[1].a() = (real_T)index; break;
348 case 156: Vehicle_B.IntegralGain_p[2].a() = (real_T)index; break;
349 case 157: Vehicle_B.Integrator[0].a() = (real_T)index; break;
350 case 158: Vehicle_B.Integrator[1].a() = (real_T)index; break;
351 case 159: Vehicle_B.Integrator[2].a() = (real_T)index; break;
352 case 160: Vehicle_B.Integrator_k[0].a() = (real_T)index; break;
353 case 161: Vehicle_B.Integrator_k[1].a() = (real_T)index; break;
354 case 162: Vehicle_B.Integrator_k[2].a() = (real_T)index; break;
355 case 163: Vehicle_B.is180degRot[0].a() = (real_T)index; break;
356 case 164: Vehicle_B.is180degRot[1].a() = (real_T)index; break;
357 case 165: Vehicle_B.is180degRot[2].a() = (real_T)index; break;
358 case 166: Vehicle_B.is180degRot_l[0].a() = (real_T)index; break;
359 case 167: Vehicle_B.is180degRot_l[1].a() = (real_T)index; break;
360 case 168: Vehicle_B.is180degRot_l[2].a() = (real_T)index; break;
361 case 169: Vehicle_B.ixj.a() = (real_T)index; break;
362 case 170: Vehicle_B.ixj_f.a() = (real_T)index; break;
363 case 171: Vehicle_B.ixj_h.a() = (real_T)index; break;
364 case 172: Vehicle_B.ixj_k.a() = (real_T)index; break;
365 case 173: Vehicle_B.ixj_l.a() = (real_T)index; break;
366 case 174: Vehicle_B.ixj_n.a() = (real_T)index; break;
367 case 175: Vehicle_B.ixj_o.a() = (real_T)index; break;
368 case 176: Vehicle_B.ixk.a() = (real_T)index; break;
369 case 177: Vehicle_B.ixk_a.a() = (real_T)index; break;
370 case 178: Vehicle_B.ixk_b.a() = (real_T)index; break;
371 case 179: Vehicle_B.ixk_e.a() = (real_T)index; break;
372 case 180: Vehicle_B.ixk_g.a() = (real_T)index; break;

```

[illegible]

[illegible]


```

581: case 387: Vehicle_B.Product8_e.a() = (real_T)index; break;
582: case 388: Vehicle_B.Product8_f.a() = (real_T)index; break;
583: case 389: Vehicle_B.Product8_g.a() = (real_T)index; break;
584: case 390: Vehicle_B.Product8_j.a() = (real_T)index; break;
585: case 391: Vehicle_B.Product8_k.a() = (real_T)index; break;
586: case 392: Vehicle_B.Product8_m.a() = (real_T)index; break;
587: case 393: Vehicle_B.Product8_me.a() = (real_T)index; break;
588: case 394: Vehicle_B.Product8_p.a() = (real_T)index; break;
589: case 395: Vehicle_B.Product_a.a() = (real_T)index; break;
590: case 396: Vehicle_B.Product_am.a() = (real_T)index; break;
591: case 397: Vehicle_B.Product_b.a() = (real_T)index; break;
592: case 398: Vehicle_B.Product_bg.a() = (real_T)index; break;
593: case 399: Vehicle_B.Product_by.a() = (real_T)index; break;
594: case 400: Vehicle_B.Product_c[0].a() = (real_T)index; break;
595: case 401: Vehicle_B.Product_c[1].a() = (real_T)index; break;
596: case 402: Vehicle_B.Product_c[2].a() = (real_T)index; break;
597: case 403: Vehicle_B.Product_c[3].a() = (real_T)index; break;
598: case 404: Vehicle_B.Product_c[4].a() = (real_T)index; break;
599: case 405: Vehicle_B.Product_c[5].a() = (real_T)index; break;
600: case 406: Vehicle_B.Product_c[6].a() = (real_T)index; break;
601: case 407: Vehicle_B.Product_c[7].a() = (real_T)index; break;
602: case 408: Vehicle_B.Product_c[8].a() = (real_T)index; break;
603: case 409: Vehicle_B.Product_c3.a() = (real_T)index; break;
604: case 410: Vehicle_B.Product_cf.a() = (real_T)index; break;
605: case 411: Vehicle_B.Product_cq.a() = (real_T)index; break;
606: case 412: Vehicle_B.Product_cw.a() = (real_T)index; break;
607: case 413: Vehicle_B.Product_cy[0].a() = (real_T)index; break;
608: case 414: Vehicle_B.Product_cy[1].a() = (real_T)index; break;
609: case 415: Vehicle_B.Product_cy[2].a() = (real_T)index; break;
610: case 416: Vehicle_B.Product_d.a() = (real_T)index; break;
611: case 417: Vehicle_B.Product_d0.a() = (real_T)index; break;
612: case 418: Vehicle_B.Product_da.a() = (real_T)index; break;
613: case 419: Vehicle_B.Product_dj.a() = (real_T)index; break;
614: case 420: Vehicle_B.Product_e.a() = (real_T)index; break;
615: case 421: Vehicle_B.Product_er.a() = (real_T)index; break;
616: case 422: Vehicle_B.Product_ew.a() = (real_T)index; break;
617: case 423: Vehicle_B.Product_ey.a() = (real_T)index; break;
618: case 424: Vehicle_B.Product_f.a() = (real_T)index; break;
619: case 425: Vehicle_B.Product_fv.a() = (real_T)index; break;
620: case 426: Vehicle_B.Product_g.a() = (real_T)index; break;
621: case 427: Vehicle_B.Product_gr.a() = (real_T)index; break;
622: case 428: Vehicle_B.Product_h.a() = (real_T)index; break;
623: case 429: Vehicle_B.Product_h0.a() = (real_T)index; break;
624: case 430: Vehicle_B.Product_h1.a() = (real_T)index; break;
625: case 431: Vehicle_B.Product_i[0].a() = (real_T)index; break;
626: case 432: Vehicle_B.Product_i[1].a() = (real_T)index; break;
627: case 433: Vehicle_B.Product_i[2].a() = (real_T)index; break;
628: case 434: Vehicle_B.Product_i2.a() = (real_T)index; break;
629: case 435: Vehicle_B.Product_if.a() = (real_T)index; break;
630: case 436: Vehicle_B.Product_j.a() = (real_T)index; break;
631: case 437: Vehicle_B.Product_jb.a() = (real_T)index; break;
632: case 438: Vehicle_B.Product_jc.a() = (real_T)index; break;
633: case 439: Vehicle_B.Product_jz.a() = (real_T)index; break;
634: case 440: Vehicle_B.Product_k.a() = (real_T)index; break;
635: case 441: Vehicle_B.Product_l.a() = (real_T)index; break;
636: case 442: Vehicle_B.Product_l5.a() = (real_T)index; break;
637: case 443: Vehicle_B.Product_m.a() = (real_T)index; break;
638: case 444: Vehicle_B.Product_mi.a() = (real_T)index; break;
639: case 445: Vehicle_B.Product_n.a() = (real_T)index; break;
640: case 446: Vehicle_B.Product_o[0].a() = (real_T)index; break;
641: case 447: Vehicle_B.Product_o[1].a() = (real_T)index; break;
642: case 448: Vehicle_B.Product_o[2].a() = (real_T)index; break;
643: case 449: Vehicle_B.Product_p.a() = (real_T)index; break;
644: case 450: Vehicle_B.Product_pb.a() = (real_T)index; break;
645: case 451: Vehicle_B.Product_pd.a() = (real_T)index; break;
646: case 452: Vehicle_B.Product_pe.a() = (real_T)index; break;
647: case 453: Vehicle_B.ProportionalGain[0].a() = (real_T)index; break;
648: case 454: Vehicle_B.ProportionalGain[1].a() = (real_T)index; break;
649: case 455: Vehicle_B.ProportionalGain[2].a() = (real_T)index; break;
650: case 456: Vehicle_B.ProportionalGain_h[0].a() = (real_T)index; break;
651: case 457: Vehicle_B.ProportionalGain_h[1].a() = (real_T)index; break;
652: case 458: Vehicle_B.ProportionalGain_h[2].a() = (real_T)index; break;
653:
654: case 459: Vehicle_B.Sqrt3.a() = (real_T)index; break;
655: case 460: Vehicle_B.Sqrt3_j.a() = (real_T)index; break;
656: case 461: Vehicle_B.Sqrt3_l.a() = (real_T)index; break;
657: case 462: Vehicle_B.Sqrt3_m.a() = (real_T)index; break;
658: case 463: Vehicle_B.Sqrt3_mq.a() = (real_T)index; break;
659: case 464: Vehicle_B.sqrt_a.a() = (real_T)index; break;
660: case 465: Vehicle_B.sqrt_g.a() = (real_T)index; break;
661: case 466: Vehicle_B.sqrt_h.a() = (real_T)index; break;
662: case 467: Vehicle_B.sqrt_i.a() = (real_T)index; break;
663: case 468: Vehicle_B.sqrt_j.a() = (real_T)index; break;
664: case 469: Vehicle_B.sqrt_k.a() = (real_T)index; break;
665: case 470: Vehicle_B.sqrt_l.a() = (real_T)index; break;
666: case 471: Vehicle_B.sqrt_m.a() = (real_T)index; break;
667: case 472: Vehicle_B.sqrt_mo.a() = (real_T)index; break;
668: case 473: Vehicle_B.sqrt_mv.a() = (real_T)index; break;
669: case 474: Vehicle_B.sqrt_mw.a() = (real_T)index; break;
670: case 475: Vehicle_B.sqrt_n.a() = (real_T)index; break;
671:
672: case 476: Vehicle_B.Sum[0].a() = (real_T)index; break;
673: case 477: Vehicle_B.Sum[1].a() = (real_T)index; break;
674: case 478: Vehicle_B.Sum[2].a() = (real_T)index; break;
675: case 479: Vehicle_B.Sum1.a() = (real_T)index; break;
676: case 480: Vehicle_B.Sum1_a.a() = (real_T)index; break;
677: case 481: Vehicle_B.Sum1_b.a() = (real_T)index; break;
678: case 482: Vehicle_B.Sum1_bu.a() = (real_T)index; break;
679: case 483: Vehicle_B.Sum1_d.a() = (real_T)index; break;
680: case 484: Vehicle_B.Sum1_d1.a() = (real_T)index; break;
681: case 485: Vehicle_B.Sum1_h.a() = (real_T)index; break;
682: case 486: Vehicle_B.Sum1_hk.a() = (real_T)index; break;
683: case 487: Vehicle_B.Sum1_k.a() = (real_T)index; break;
684: case 488: Vehicle_B.Sum1_l.a() = (real_T)index; break;

```

```

685:         case 489: Vehicle_B.Sum1_m.a() = (real_T)index; break;
686:         case 490: Vehicle_B.Sum1_n.a() = (real_T)index; break;
687:         case 491: Vehicle_B.Sum2.a() = (real_T)index; break;
688:         case 492: Vehicle_B.Sum2_a.a() = (real_T)index; break;
689:         case 493: Vehicle_B.Sum2_g.a() = (real_T)index; break;
690:         case 494: Vehicle_B.Sum2_i.a() = (real_T)index; break;
691:         case 495: Vehicle_B.Sum2_k.a() = (real_T)index; break;
692:         case 496: Vehicle_B.Sum2_l.a() = (real_T)index; break;
693:         case 497: Vehicle_B.Sum2_la.a() = (real_T)index; break;
694:         case 498: Vehicle_B.Sum2_m.a() = (real_T)index; break;
695:         case 499: Vehicle_B.Sum2_mr.a() = (real_T)index; break;
696:         case 500: Vehicle_B.Sum2_o.a() = (real_T)index; break;
697:         case 501: Vehicle_B.Sum2_p.a() = (real_T)index; break;
698:         case 502: Vehicle_B.Sum2_p3.a() = (real_T)index; break;
699:         case 503: Vehicle_B.Sum3.a() = (real_T)index; break;
700:         case 504: Vehicle_B.Sum3_b.a() = (real_T)index; break;
701:         case 505: Vehicle_B.Sum3_b5.a() = (real_T)index; break;
702:         case 506: Vehicle_B.Sum3_bi.a() = (real_T)index; break;
703:         case 507: Vehicle_B.Sum3_d.a() = (real_T)index; break;
704:         case 508: Vehicle_B.Sum3_e.a() = (real_T)index; break;
705:         case 509: Vehicle_B.Sum3_h.a() = (real_T)index; break;
706:         case 510: Vehicle_B.Sum3_j.a() = (real_T)index; break;
707:         case 511: Vehicle_B.Sum3_jh.a() = (real_T)index; break;
708:         case 512: Vehicle_B.Sum3_l.a() = (real_T)index; break;
709:         case 513: Vehicle_B.Sum3_o.a() = (real_T)index; break;
710:         case 514: Vehicle_B.Sum3_p.a() = (real_T)index; break;
711:         case 515: Vehicle_B.Sum_b[0].a() = (real_T)index; break;
712:         case 516: Vehicle_B.Sum_b[1].a() = (real_T)index; break;
713:         case 517: Vehicle_B.Sum_b[2].a() = (real_T)index; break;
714:         case 518: Vehicle_B.Sum_c.a() = (real_T)index; break;
715:         case 519: Vehicle_B.Sum_d.a() = (real_T)index; break;
716:         case 520: Vehicle_B.Sum_d5.a() = (real_T)index; break;
717:         case 521: Vehicle_B.Sum_e.a() = (real_T)index; break;
718:         case 523: Vehicle_B.Sum_f.a() = (real_T)index; break;
719:         case 524: Vehicle_B.Sum_ff[0].a() = (real_T)index; break;
720:         case 525: Vehicle_B.Sum_ff[1].a() = (real_T)index; break;
721:         case 526: Vehicle_B.Sum_ff[2].a() = (real_T)index; break;
722:         case 527: Vehicle_B.Sum_g.a() = (real_T)index; break;
723:         case 528: Vehicle_B.Sum_g5.a() = (real_T)index; break;
724:         case 529: Vehicle_B.Sum_gh.a() = (real_T)index; break;
725:         case 530: Vehicle_B.Sum_gt.a() = (real_T)index; break;
726:         case 531: Vehicle_B.Sum_gz.a() = (real_T)index; break;
727:         case 532: Vehicle_B.Sum_i.a() = (real_T)index; break;
728:         case 533: Vehicle_B.Sum_iw.a() = (real_T)index; break;
729:         case 534: Vehicle_B.Sum_j.a() = (real_T)index; break;
730:         case 535: Vehicle_B.Sum_k.a() = (real_T)index; break;
731:         case 536: Vehicle_B.Sum_kb.a() = (real_T)index; break;
732:         case 537: Vehicle_B.Sum_kc.a() = (real_T)index; break;
733:         case 538: Vehicle_B.Sum_ki.a() = (real_T)index; break;
734:         case 539: Vehicle_B.Sum_kk.a() = (real_T)index; break;
735:         case 540: Vehicle_B.Sum_l[0].a() = (real_T)index; break;
736:         case 541: Vehicle_B.Sum_l[1].a() = (real_T)index; break;
737:         case 542: Vehicle_B.Sum_l[2].a() = (real_T)index; break;
738:         case 543: Vehicle_B.Sum_lm.a() = (real_T)index; break;
739:         case 544: Vehicle_B.Sum_ln.a() = (real_T)index; break;
740:         case 545: Vehicle_B.Sum_m[0].a() = (real_T)index; break;
741:         case 546: Vehicle_B.Sum_m[1].a() = (real_T)index; break;
742:         case 547: Vehicle_B.Sum_m[2].a() = (real_T)index; break;
743:         case 548: Vehicle_B.Sum_n.a() = (real_T)index; break;
744:         case 549: Vehicle_B.Sum_n1.a() = (real_T)index; break;
745:         case 550: Vehicle_B.Sum_n3.a() = (real_T)index; break;
746:         case 551: Vehicle_B.Sum_nx.a() = (real_T)index; break;
747:         case 552: Vehicle_B.Sum_o.a() = (real_T)index; break;
748:         case 553: Vehicle_B.Sum_oa.a() = (real_T)index; break;
749:         case 554: Vehicle_B.Sum_oq.a() = (real_T)index; break;
750:         case 555: Vehicle_B.Sum_oz.a() = (real_T)index; break;
751:         case 556: Vehicle_B.Sum_p.a() = (real_T)index; break;
752:         case 557: Vehicle_B.Sum_pb[0].a() = (real_T)index; break;
753:         case 558: Vehicle_B.Sum_pb[1].a() = (real_T)index; break;
754:         case 559: Vehicle_B.Sum_pb[2].a() = (real_T)index; break;
755:         case 560: Vehicle_B.Sum_pd[0].a() = (real_T)index; break;
756:         case 561: Vehicle_B.Sum_pd[1].a() = (real_T)index; break;
757:         case 562: Vehicle_B.Sum_pd[2].a() = (real_T)index; break;
758:         case 563: Vehicle_B.Sum_pw.a() = (real_T)index; break;
759:         case 564: Vehicle_B.SumofElements_c.a() = (real_T)index; break;
760:         case 565: Vehicle_B.SumofElements_l.a() = (real_T)index; break;
761:         case 566: Vehicle_B.Switch[0].a() = (real_T)index; break;
762:         case 567: Vehicle_B.Switch[1].a() = (real_T)index; break;
763:         case 568: Vehicle_B.Switch3[0].a() = (real_T)index; break;
764:         case 569: Vehicle_B.Switch3[1].a() = (real_T)index; break;
765:         case 570: Vehicle_B.Switch3[2].a() = (real_T)index; break;
766:         case 571: Vehicle_B.Switch_b[0].a() = (real_T)index; break;
767:         case 572: Vehicle_B.Switch_b[1].a() = (real_T)index; break;
768:         case 573: Vehicle_B.Switch_c[0].a() = (real_T)index; break;
769:         case 574: Vehicle_B.Switch_c[1].a() = (real_T)index; break;
770:         case 575: Vehicle_B.Switch_c[2].a() = (real_T)index; break;
771:         case 576: Vehicle_B.Switch_c[3].a() = (real_T)index; break;
772:         case 577: Vehicle_B.Switch_f[0].a() = (real_T)index; break;
773:         case 578: Vehicle_B.Switch_f[1].a() = (real_T)index; break;
774:         case 579: Vehicle_B.Switch_f[0].a() = (real_T)index; break;
775:
776:         case 580: Vehicle_B.TmpSignalConversionAtDotProdu_d[0].a() = (real_T)index; break
777: ;
778:         case 581: Vehicle_B.TmpSignalConversionAtDotProdu_d[1].a() = (real_T)index; break
779 ;
780:         case 582: Vehicle_B.TmpSignalConversionAtDotProdu_d[2].a() = (real_T)index; break
781 ;
782:         case 583: Vehicle_B.TmpSignalConversionAtDotProdu_j[0].a() = (real_T)index; break
783 ;
784:         case 584: Vehicle_B.TmpSignalConversionAtDotProdu_j[1].a() = (real_T)index; break
785 ;
786:         case 585: Vehicle_B.TmpSignalConversionAtDotProdu_j[2].a() = (real_T)index; break
787 ;

```

```

782         case 586: Vehicle_B.TmpSignalConversionAtDotProdu_k[0].a() = (real_T)index; break;
783         ;
784         case 587: Vehicle_B.TmpSignalConversionAtDotProdu_k[1].a() = (real_T)index; break;
785         ;
786         case 588: Vehicle_B.TmpSignalConversionAtDotProdu_k[2].a() = (real_T)index; break;
787         ;
788         case 589: Vehicle_B.Tsamp[0].a() = (real_T)index; break;
789         case 590: Vehicle_B.Tsamp[1].a() = (real_T)index; break;
790         case 591: Vehicle_B.Tsamp[2].a() = (real_T)index; break;
791         case 592: Vehicle_B.Tsamp_f[0].a() = (real_T)index; break;
792         case 593: Vehicle_B.Tsamp_f[1].a() = (real_T)index; break;
793         case 594: Vehicle_B.Tsamp_f[2].a() = (real_T)index; break;
794         case 595: Vehicle_B.UD[0].a() = (real_T)index; break;
795         case 596: Vehicle_B.UD[1].a() = (real_T)index; break;
796         case 597: Vehicle_B.UD[2].a() = (real_T)index; break;
797         case 598: Vehicle_B.UD_l[0].a() = (real_T)index; break;
798         case 599: Vehicle_B.UD_l[1].a() = (real_T)index; break;
799         case 600: Vehicle_B.UD_l[2].a() = (real_T)index; break;
800         case 601: Vehicle_B.UnaryMinus.a() = (real_T)index; break;
801         case 602: Vehicle_B.UnaryMinus1.a() = (real_T)index; break;
802         case 603: Vehicle_B.UnaryMinus2.a() = (real_T)index; break;
803         case 604: Vehicle_B.VectorConcatenate[0].a() = (real_T)index; break;
804         case 605: Vehicle_B.VectorConcatenate[1].a() = (real_T)index; break;
805         case 606: Vehicle_B.VectorConcatenate[2].a() = (real_T)index; break;
806         case 607: Vehicle_B.VectorConcatenate_a[0].a() = (real_T)index; break;
807         case 608: Vehicle_B.VectorConcatenate_a[1].a() = (real_T)index; break;
808         case 609: Vehicle_B.VectorConcatenate_a[2].a() = (real_T)index; break;
809         case 610: Vehicle_B.zr[0].a() = (real_T)index; break;
810         case 611: Vehicle_B.zr[1].a() = (real_T)index; break;
811         case 612: Vehicle_B.zr[2].a() = (real_T)index; break;
812         case 614: Vehicle_B.norm_b.a() = (real_T)index; break;
813
814     }
815 }
816

```


Bibliography

- [1] Leonardo reyneri, Alejandro cases, Yolanda morilla, Sergio Asensi, and Antonio Martinez. «A compact model to evaluate the effects of high level C++ code hardening in radiation environments». In: 1 (Apr. 2019) (cit. on p. 2).
- [2] Ibrahim Kök. «Comparison and Analysis of Attitude Control Systems of a Satellite Using Reaction Wheel Actuators». In: (Oct. 2012) (cit. on pp. 4, 6, 7, 28, 29).
- [3] Muhammad Yasir. «Development and Implementation of the attitude control algorithms for the micro-satellite Flying Laptop». In: (Mar. 2010) (cit. on pp. 4, 5, 8, 32).
- [4] «<https://i.stack.imgur.com/A33El.jpg>». In: () (cit. on p. 6).
- [5] «<https://www.allaboutcircuits.com/technical-articles/dont-get-lost-in-deep-space-understanding-quaternions/#:~:text=Quaternions/20are/20an/20alternate/20way,not/20suffer/20from/20gimbal/20lock.>». In: (2017) (cit. on p. 8).
- [6] Catanzaro Alessandro Rocco. «Design and simulation analysis of Attitude and Determination Control System for the CubeSat». In: () (cit. on p. 9).
- [7] Nnamdi N. Jibiri Victor U. J. Nwankwo and Michael T. Kio. «The Impact of Space Radiation Environment on Satellites Operation in Near-Earth Space». In: (June 2020) (cit. on pp. 9–11).
- [8] «<https://it.mathworks.com/matlabcentral/fileexchange/70030-aerospace-blockset-cubesat-simulation-library>». In: (Sept. 2020) (cit. on p. 13).
- [9] «<https://www.mathworks.com/help/releases/R2020b/aeroblks/sphericalharmonicgravitymodel.html>». In: (2010) (cit. on p. 15).
- [10] Paweł Zagórski. «Modeling disturbances influencing an Earth-orbiting satellite». In: (May 2012) (cit. on p. 17).