Guidance and Control Software Development Specification

Version 2.3 with Formal Modifications 2.3-1 through 2.3-7

B. Edward Withers (Research Triangle Institute)
Bernice Becher (Lockheed Martin Engineering and Sciences Corp.)

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23681

This document has been prepared for the exclusive use of the Federal Aviation Administration at the 1995 Software Standardization Workshop to be held in Denver, CO, August 1-3, 1995. This document (in part or in whole) is not intended for reproduction or public distribution.
ACKNOWLEDGEMENTS

We wish to acknowledge Peter Padilla for his guidance and assistance in reviewing the requirements for this document and in recommending revisions so that the requirements might more accurately reflect those of an actual lander. We wish to acknowledge Earle Migneault for his work on earlier experiments of this type and for his vision in suggesting the application to be used in this experiment. We also wish to acknowledge Don C. Rich, Douglas S. Lowman, R. C. Buckland, Anita M. Shagena, and Janet R. Dunham, all of Research Triangle Institute, for their earlier work on this project.

reprinted with permission via Kelly J.Hayhurst
NASA Langley Research Center
Preface

The NASA Langley Research Center has been conducting a series of software error studies in an effort to better understand the software failure process and improve development and reliability estimation techniques for avionics software. The Guidance and Control Software (GCS) project is the latest study in the series (ref. 1). This project involves production of guidance and control software for the purpose of gathering failure data from a credible software development environment. To increase the credibility and relevance of this study, guidelines used in the development of commercial aircraft were adopted. The use of the Requirements and Technical Concepts for Aviation RTCA/DO-178B guidelines, "Software Considerations in Airborne Systems and Equipment Certification," is required by the Federal Aviation Administration (FAA) for developing software to be certified for use in commercial aircraft equipment (ref. 2).

This document is one part of the life cycle data required to fulfill the RTCA/DO-178B guidelines. The life cycle data are used to demonstrate compliance with the guidelines by describing the application of the procedures and techniques used during the development of flight software and the results of the development process. For the GCS project, the life cycle data consists of the following:

- Plan for Software Aspects of Certification
- Software Development Standards
- Software Verification Plan
- Software Configuration Management Plan
- Software Quality Assurance Plan
- Software Requirements Document
- Software Design Description
- Source Code
- Executable Object Code
- Software Verification Procedures and Cases
- Software Verification Results
- Software Configuration Index
- Problem Reports
- Software Configuration Management Records
- Software Quality Assurance Records
- Software Accomplishment Summary
- Simulator and Test Environment Description and User's Guide
A GCS implementation (code which fulfills the requirements outlined in the Software Requirements Data) runs in conjunction with a software simulator that provides input based on an expected usage distribution in the operational environment, provides response modeling, and receives data from the implementation. For the purposes of the project, a number of GCS implementations are being developed by different programmers according to the structured approach found in the DO-178B guidelines. The GCS simulator is designed to allow an experimenter to run one or more implementations in a multitasking environment and collect data on the comparison of the results from multiple implementations. Certain constraints have been incorporated in the software requirements due to the nature of the GCS project.
FOREWORD

This specification defines a guidance and control system for a planetary landing vehicle during its terminal phase of descent. It is written for an experienced programmer with two or more years of full-time industrial programming experience using a scientific programming language. The programmer should have an adequate background, either through college courses or job training in mathematics, physics, differential equations, and numerical integration. The specification was written with the assumption that the implementation would be coded in FORTRAN; however, other languages can be used.
# CONTENTS

ACKNOWLEDGEMENTS ........................................................................................................... i
PREFACE ................................................................................................................................. iii
FOREWORD ............................................................................................................................... v
LIST OF FIGURES ...................................................................................................................... xi
LIST OF TABLES .......................................................................................................................... xiii
1. INTRODUCTION .................................................................................................................. 1
   PURPOSE OF THE GUIDANCE AND CONTROL SOFTWARE .................................................. 3
   VEHICLE CONFIGURATION .................................................................................................... 3
   TERMINAL DESCENT ........................................................................................................... 6
   VEHICLE DYNAMICS ............................................................................................................ 6
      Frames of Reference ........................................................................................................... 6
      Linear Velocity ................................................................................................................. 7
      Vehicle Position .............................................................................................................. 7
      Angular Velocity ............................................................................................................ 8
      Vehicle Attitude ............................................................................................................. 8
      Acceleration ................................................................................................................... 8
      Further Reading ............................................................................................................ 8
   VEHICLE GUIDANCE ......................................................................................................... 10
   ENGINES ............................................................................................................................... 10
      Axial Engine (Thrust) Control ............................................................................................ 10
      Roll Engine Control ....................................................................................................... 10
   GENERAL INFORMATION ..................................................................................................... 11
   NOTATION ............................................................................................................................. 11
      Matrices and Arrays ......................................................................................................... 11
      Operators ......................................................................................................................... 11
   DEFINITIONS ....................................................................................................................... 12
   CONVENTIONS ..................................................................................................................... 14
      FORTRAN Convention ..................................................................................................... 14
   REQUIREMENTS .................................................................................................................... 14
      Order of Processing ........................................................................................................ 14
      Calls to GCS_SIM_RENDEZVOUS ................................................................................. 14
      Control Signals ............................................................................................................... 14
      Number Representations ................................................................................................. 14
Conversion of Units .................................................. 14
Global Data Store Organization .................................. 14
Use of Variables That Are Not in the Global Data Stores .... 15
Use of Tables .......................................................... 15
Rotation of History Variables ...................................... 15
Precision ................................................................. 15

EXCEPTION HANDLING ........................................... 16
Exception Conditions ................................................ 16
Action to be Taken for Each Specified Exception Condition 17
Output to be Generated for Each Exception Condition ..... 17

2. LEVELS 0 AND 1 SPECIFICATION .............................. 19
   LEVEL 0 SPECIFICATION ....................................... 21
   LEVEL 1 SPECIFICATION ...................................... 26

3. LEVEL 2 SPECIFICATION ......................................... 29
   PROCESS SPECIFICATION (P-Spec) 1: INIT_GCS .......... 31

4. LEVEL 3 FLOW DIAGRAMS AND C-SPECS .................... 35
   SCHEDULING ....................................................... 43

5. P-SPECS FOR LEVELS 3 and 4 ................................. 45
   AECLP -- Axial Engine Control Law Processing (P-Spec 2.3.1) 47
   ARSP -- Altimeter Radar Sensor Processing (P-Spec 2.1.2) ... 53
   ASP -- Accelerometer Sensor Processing (P-Spec 2.1.1) ....... 55
   CP -- Communications Processing (P-Spec 2.4) ................ 59
   CRCP -- Chute Release Control Processing (P-Spec 2.3.3) ... 63
   GP -- Guidance Processing (P-Spec 2.2) ....................... 65
   GSP -- Gyroscope Sensor Processing (P-Spec 2.1.4) .......... 73
   RECLP -- Roll Engine Control Law Processing (P-Spec 2.3.2) 75
   TDLRSP -- Touch Down Landing Radar Sensor Processing (P-Spec 2.1.3) 77
   TDSP -- Touch Down Sensor Processing (P-Spec 2.1.6) ....... 83
   TSP -- Temperature Sensor Processing (P-Spec 2.1.5) ......... 85

6. DATA REQUIREMENTS DICTIONARY ............................ 89
   PART I. DATA ELEMENT DESCRIPTIONS ...................... 91
   PART II. CONTENTS OF DATA STORES ......................... 107
   PART III. CONTROL SIGNALS, DATA CONDITIONS, AND GROUP .... 111
   A. NOTATION FOR LEVELS 0, 1, 2, AND 3 SPECIFICATION ..... 115
   B. IMPLEMENTATION NOTES ..................................... 119
   INTERFACE ......................................................... 121
LIST OF TABLES

1.1 ROTATION OF VARIABLES ................................................................. 15

2.1 CONTROL SPECIFICATION (C-SPEC) 0: GCS .................................................. 28

3.1 C-Spec 2: RUN_GCS .......................................................................... 34

4.1 C-Spec 2.1: SP -- SENSOR PROCESSING ................................................. 39

4.2 C-Spec 2.3: CLP -- CONTROL LAW PROCESSING ............................. 42

4.3 FUNCTIONAL UNIT SCHEDULING ............................................................. 43

5.1 DETERMINATION OF AXIAL ENGINE TEMPERATURE ......................... 48

5.2 DETERMINATION OF ERROR TERMS .................................................... 50

5.3 DETERMINATION OF AXIAL ENGINE COMMANDS ............................... 50

5.4 DETERMINATION OF ALTITUDE STATUS ............................................... 54

5.5 PACKET VARIABLES ........................................................................... 61

5.6 SAMPLE MASK .................................................................................... 61

5.7 PACKET BYTE STRUCTURE ................................................................. 62

5.8 DIFFERENTIAL EQUATIONS ................................................................. 66

5.9 DETERMINATION OF AXIAL AND ROLL ENGINE ON/OFF SWITCHES .... 67

5.10 DETERMINATION OF GUIDANCE PHASE .............................................. 70

5.11 DETERMINATION OF RADAR BEAM STATES ........................................ 79

5.12 PROCESSING OF DOPPLER RADAR BEAMS IN LOCK ..................... 81

5.13 DETERMINATION OF TOUCH DOWN SENSOR AND STATUS ............. 83

6.1 DATA STORE: GUIDANCE_STATE ......................................................... 107

6.2 DATA STORE: EXTERNAL ..................................................................... 108

6.3 DATA STORE: SENSOR_OUTPUT ............................................................ 108

6.4 DATA STORE: RUN_PARAMETERS ......................................................... 109

6.5 CONTROL SIGNALS ............................................................................. 111

6.6 DATA CONDITIONS ............................................................................... 111

6.7 INITIALIZATION DATA .......................................................................... 112

6.8 TEMP_DATA ......................................................................................... 114

6.9 SENSOR_DATA ..................................................................................... 114

6.10 OUTPUT_DATA .................................................................................... 114

6.11 OUTPUT_CONTROL ........................................................................... 114

6.12 FRAME_DATA ...................................................................................... 114

C.1 INITIAL VALUES PROVIDED FOR USE IN INTEGRATION ..................... 128
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>THE LANDING VEHICLE DURING DESCENT</td>
<td>4</td>
</tr>
<tr>
<td>1.2</td>
<td>A TYPICAL TERMINAL DESCENT TRAJECTORY</td>
<td>5</td>
</tr>
<tr>
<td>1.3</td>
<td>ENGINEERING ILLUSTRATION OF VEHICLE</td>
<td>9</td>
</tr>
<tr>
<td>2.1</td>
<td>STRUCTURE OF THE GCS SPECIFICATION</td>
<td>23</td>
</tr>
<tr>
<td>2.2</td>
<td>DATA CONTEXT DIAGRAM: LANDER</td>
<td>24</td>
</tr>
<tr>
<td>2.3</td>
<td>CONTROL CONTEXT DIAGRAM: LANDER</td>
<td>25</td>
</tr>
<tr>
<td>2.4</td>
<td>DATA FLOW DIAGRAM (DFD) 0: GCS</td>
<td>26</td>
</tr>
<tr>
<td>2.5</td>
<td>CONTROL FLOW DIAGRAM (CFD) 0: GCS</td>
<td>27</td>
</tr>
<tr>
<td>3.1</td>
<td>DFD 2: RUN_GCS</td>
<td>32</td>
</tr>
<tr>
<td>3.2</td>
<td>CFD 2: RUN_GCS</td>
<td>33</td>
</tr>
<tr>
<td>4.1</td>
<td>DFD 2.1: SP -- SENSOR PROCESSING</td>
<td>37</td>
</tr>
<tr>
<td>4.2</td>
<td>CFD 2.1: SP -- SENSOR PROCESSING</td>
<td>38</td>
</tr>
<tr>
<td>4.3</td>
<td>DFD 2.3: CLP -- CONTROL LAW PROCESSING</td>
<td>40</td>
</tr>
<tr>
<td>4.4</td>
<td>CFD 2.3: CLP -- CONTROL LAW PROCESSING</td>
<td>41</td>
</tr>
<tr>
<td>5.1</td>
<td>VELOCITY-ALTITUDE CONTOUR</td>
<td>69</td>
</tr>
<tr>
<td>5.2</td>
<td>GRAPH FOR DERIVING ROLL ENGINE COMMANDS</td>
<td>76</td>
</tr>
<tr>
<td>5.3</td>
<td>DOPPLER RADAR BEAM LOCATIONS</td>
<td>78</td>
</tr>
<tr>
<td>5.4</td>
<td>CALIBRATION OF THERMOCOUPLE PAIR</td>
<td>86</td>
</tr>
<tr>
<td>A.1</td>
<td>GRAPHICAL SYMBOLS USED IN STRUCTURED ANALYSIS DIAGRAMS</td>
<td>118</td>
</tr>
<tr>
<td>B.1</td>
<td>DIAGRAM OF STORAGE AS SEEN BY GCS IMPLEMENTATIONS</td>
<td>122</td>
</tr>
</tbody>
</table>
1. INTRODUCTION
PURPOSE OF THE GUIDANCE AND CONTROL SOFTWARE

The Guidance and Control Software (GCS) represents the Viking lander [20] on-board navigational software. The purpose of this software is to:

1. provide guidance and engine control of the vehicle (shown in Figure 1.1) during its terminal phase of descent onto a surface and
2. communicate sensory information about the vehicle and its descent to some other receiving device.

A typical descent trajectory is shown in Figure 1.2.

The initialization of the GCS starts the sensing of vehicle altitude. When a predefined engine ignition altitude is sensed by the altimeter radar, the GCS begins guidance and control of the vehicle. The axial engines are ignited: while the axial engines are warming up, the parachute remains connected to the vehicle. During this engine warm-up phase, the aerodynamics of the parachute dictate the trajectory followed by the vehicle. Vehicle attitude is maintained by firing the engines in a throttled-down condition. Once the main engines become hot, the parachute is released and the GCS performs an attitude correction maneuver and then follows a controlled acceleration descent until a predetermined velocity-altitude contour is crossed (see Figure 5.1). The GCS then attempts to maintain the descent of the vehicle along this predetermined velocity-altitude contour. The vehicle descends along this contour until a predefined engine shut off altitude is reached or touchdown is sensed. After all engines are shut off, the vehicle free-falls to the surface.

VEHICLE CONFIGURATION

The vehicle to be controlled is a guidance package containing sensors which obtain information about the vehicle state, a guidance and control computer, and actuators providing the thrust necessary for maintaining a safe descent. The vehicle has three accelerometers (one for each body axis), one doppler radar with four beams, one altimeter radar, two temperature sensors, three strapped-down gyroscopes, three opposed pairs of roll engines, three axial thrust engines, one parachute release actuator, and a touchdown sensor. The vehicle has a hexagonal, box-like shape with three legs and a surface sensing rod protruding from its undersurface.
Figure 1.1 THE LANDING VEHICLE DURING DESCENT
Figure 1.2 A TYPICAL TERMINAL DESCENT TRAJECTORY

Parachute Descent (GCS Starts)

Engines are Turned on and Begin Warmup

Chute is Released (Terminal Descent Begins)

Engines are Turned off

Touch Down (GCS Stops)
TERMINAL DESCENT

Prior to the terminal descent phase, the vehicle falls with a parachute attached. This parachute is released seconds after the engines ignite, and terminal descent begins. During terminal descent, the vehicle follows a modified gravity-turn guidance law until a predetermined altitude is reached. The atmosphere introduces drag forces, including the random effects of wind. Independently throttled engines slow the vehicle down. These engines can control the vehicle's orientation, and roll engines control the vehicle's roll rate. Roll control is necessary to keep the doppler radars in lock and insure that the desired touch down attitude (land on two legs prior to the third) is maintained.

The velocity during descent follows the predetermined velocity-altitude contour. At a specific altitude above the planet surface, the vehicle is maintained at a constant descent velocity. Once the surface is sensed, all engines are shut down and the vehicle free falls to the surface.

VEHICLE DYNAMICS

Frames of Reference

Terminal descent is described in terms of two coordinate systems:

1. the surface-oriented coordinate system, and
2. the vehicle-oriented coordinate system.

In the surface coordinate system, the $\bar{z}_p$ axis is viewed as normal to the surface and points down as shown in Figure 1.2. The $\bar{x}_p$ axis points north, and the $\bar{y}_p$ points east.

By defining a unit vector as a vector of length equal to one unit along each axis in both the planetary and vehicular frames of reference, a relation between these two frames of reference may be established. Any vector can then be defined as a multiple of the unit vector along each of the axes defined in the frame of reference. Thus, the velocity of the vehicle $\vec{V}$ may be defined in the vehicle's frame of reference as: $V_x \hat{i}_c + V_y \hat{j}_c + V_z \hat{k}_c$, where $\hat{i}_c, \hat{j}_c$, and $\hat{k}_c$ are the unit vectors in the $x, y,$ and $z$ directions of the vehicles coordinate system (unit vectors are usually represented by lower case $i, j,$ or $k$ with a hat to show that they are unit vectors). $V_x, V_y,$ and $V_z,$ represent the components of the vehicle velocity in the given direction. At the same time, the velocity of the vehicle may be described in the planetary coordinate system as: $V_{xp} \hat{i}_p + V_{yp} \hat{j}_p + V_{zp} \hat{k}_p$, where the subscript $p$ represents planetary rather than vehicle coordinates. Note, since the two coordinate systems are not oriented in the same direction, the values of $V_{zp}$ will not be equal to $V_{zp}$, but the magnitude of the total vector $\vec{V}$ will be the same in both systems. Also the difference in the magnitudes of individual components represents the difference in relative orientation between the two coordinate systems.

The dot product $(\vec{a} \cdot \vec{b})$ is defined as the magnitude of $\vec{a}$ multiplied by the magnitude of $\vec{b}$ and then by the cosine of the angle between the vectors.
\[ \vec{a} \cdot \vec{b} = |a||b| \cos \angle \vec{a} \vec{b} \]

The dot product is used to project \( \vec{a} \) onto \( \vec{b} \) and can be used to project a vector in one frame of reference onto another one. Rather than calculate the needed cosines each time a vector must be transformed from one frame of reference into another, the cosines of the angles between each unit vector of the vehicular and planetary coordinate systems are computed and placed into a direction cosine matrix. This matrix is then used along with the vector's magnitude in each dimension of the original frame of reference to compute a dot product. This product gives the vector's magnitude in each dimension of the new frame of reference.

The transformation between the vehicle and the surface coordinate systems at time \( t \) is specified by a matrix of direction cosines,

\[
\begin{pmatrix}
  l_1 & l_2 & l_3 \\
  m_1 & m_2 & m_3 \\
  n_1 & n_2 & n_3
\end{pmatrix} =
\begin{pmatrix}
  \cos \theta(i, i_p) & \cos \theta(i, j_p) & \cos \theta(i, k_p) \\
  \cos \theta(j, i_p) & \cos \theta(j, j_p) & \cos \theta(j, k_p) \\
  \cos \theta(k, i_p) & \cos \theta(k, j_p) & \cos \theta(k, k_p)
\end{pmatrix},
\]

where \( \theta(i, j) \) denotes the angle between vectors \( \hat{i} \) and \( \hat{j} \), etc.

The change in orientation of the vehicle during descent makes the update of the direction cosine matrix necessary at each time step. This update is specified in the following equation:

\[
\frac{d}{dt} \begin{pmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{pmatrix} = \begin{pmatrix} 0 & r_v & -q_v \\ -r_v & 0 & p_v \\ q_v & -p_v & 0 \end{pmatrix} \begin{pmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{pmatrix},
\]

where the matrix containing the \( p_v \), \( q_v \), and \( r_v \) terms is the rate of rotation about the axes of the vehicle which may be obtained from sensor values.

**Linear Velocity**

The linear components of velocity for the vehicle during terminal descent are denoted by \( \dot{x}_v \), \( \dot{y}_v \), and \( \dot{z}_v \) in the vehicle coordinate system and by \( \dot{x}_p \), \( \dot{y}_p \), and \( \dot{z}_p \) in the surface coordinate system, where the dot (\( \dot{\cdot} \)) notation indicates derivatives with respect to time.

**Vehicle Position**

Vehicle position is expressed in terms of the surface coordinate system by transforming change in position (velocity) in the vehicle coordinate system into change in position in the surface frame and integrating as follows:
\[
\begin{pmatrix}
\dot{x}_p \\
\dot{y}_p \\
\dot{z}_p \\
\end{pmatrix} =
\begin{pmatrix}
l_1 & m_1 & n_1 \\
l_2 & m_2 & n_2 \\
l_3 & m_3 & n_3 \\
\end{pmatrix}
\begin{pmatrix}
x_v \\
y_v \\
z_v \\
\end{pmatrix}
\]

and

\[
\begin{pmatrix}
x_p \\
y_p \\
z_p \\
\end{pmatrix} = \int \begin{pmatrix}
\dot{x}_p \\
\dot{y}_p \\
\dot{z}_p \\
\end{pmatrix}
d\tau
\]

**Angular Velocity**

Roll, pitch, and yaw angular velocities are represented by the quantities \( p_v, q_v, \) and \( r_v \), in the vehicle frame of reference only. Roll is about the \( \bar{x}_v \) axis, pitch is about the \( \bar{y}_v \) axis, and yaw is about the \( \bar{z}_v \) axis, as shown in Figure 1.3. A more in-depth explanation of angular velocity naming conventions and other related material may be found in section II, part B of Reference [3].

**Vehicle Attitude**

The vehicle attitude at time \( t \) is a function of the vehicle attitude (known by reference to celestial objects) at the start of descent at time \( t_0 \) and the cumulative changes in attitude from time \( t_0 \) to time \( t \).

**Acceleration**

The linear components of acceleration for the vehicle in the vehicle frame of reference during terminal descent are denoted by \( \ddot{x}_v, \ddot{y}_v, \) and \( \ddot{z}_v \), respectively.

**Further Reading**

The subjects of vector mathematics, transformations between frames of references, vector calculus, and rotating coordinate systems may not be sufficiently covered here for the user; however, such depth is not intended for this document. Chapter 4 of *Classical Mechanics* [4] contains a detailed explanation of rigid body motion and transformation of vectors into multiple frames of reference or coordinate systems. Chapters 15 and 16 of *Engineering Mechanics* [5] contains a more basic approach to the same ideas of multiple frames of reference and vector mechanics. Chapter 14 of [6] and Chapter 5 of [7] also discuss rotational motion and multiple frames of reference, as well as vector mechanics and calculus. Two other books of possible interest are [8] and [9]. Both cover the mechanics of particles and dynamics, with strong references to particle trajectories and rocket dynamics. Also, these texts are basic in nature and require only a rudimentary knowledge of physics, math, or engineering.
Figure 1.3 ENGINEERING ILLUSTRATION OF VEHICLE

Bottom View (x out of page)

Axial Engine (3)

Foot Pad (3)

Roll Engine (3)

$y_v$

$z_v$

Side View (y out of page)

$y_v$

$z_v$

Side View (z into page)

$y_v$

$z_v$

positive axial thrust

$+q$ (pitch)

$+p$ (roll)

positive roll thrust

$+r$ (yaw)
VEHICLE GUIDANCE

Vehicle guidance is accomplished by varying the engine thrust so that the vehicle follows a single predetermined velocity-altitude contour. This contour is made available during GCS initialization. Applying too great a deceleration early in the descent brings the vehicle velocity to its terminal value too high above the surface, resulting in insufficient propellant for final descent. Applying too small a thrust lets the vehicle impact the surface with too great a velocity. Either condition could be disastrous. As soon as the touch down sensor touches the surface, the engines are shut off. Approximately ninety percent of propellant or thrust is used to minimize gravity losses; the remaining ten percent is used for steering.

A gravity-turn steering law is mechanized by rotating the vehicle in pitch and yaw until the body's lateral axis velocities are zero (causing the thrust axis to point along the total velocity vector). The action of gravity causes the thrust axis to rotate toward the vertical as the total velocity is reduced. An arbitrary roll orientation is maintained with an attitude hold mode during the descent.

ENGINES

The vehicle has three axial engines that supply the force necessary to slow the vehicle and allow it to safely land. Roll is controlled by three pairs of roll engines on the lander supplying rotational thrust. Figure 1.3 shows the axial and roll engines and the resulting thrust forces they impart to the vehicle.

Axial Engine (Thrust) Control

Three thrust engines first orient the vehicle so that their combined thrust vector opposes the vehicle's velocity vector. Thrust (axial direction) engine control is a function of pitch error, yaw error, thrust error, and deviation from the velocity-altitude contour. A combination of proportional and integral control (PI) logic is applied to pitch and yaw control. The integral portion helps to reduce the steady-state pitch and yaw error.

If no thrust error or velocity-altitude contour deviation occurs, then axial engine response provides only pitch and yaw control via the PI control law. Use of this control law implies that the overshoot problem for pitch-yaw control is probably small.

Thrust control is implemented by a proportional-integral-derivative (PID) control law. The derivative control added here damps out overshoot.

Roll Engine Control

Roll control is attained by pulsing the three pairs of roll engines and is a function of roll angle deviation and roll rate (p_r) about the x axis. Roll engine specific impulse and thrust per unit time are constant with the integrated thrust controlled by pulse rate. Angle deviations are controlled within a very small range of 0.25 to 0.35 degrees.
GENERAL INFORMATION

NOTATION

Matrices and Arrays

It should be noted that throughout this specification, the words matrix and array are often interchanged. No
significance should be placed upon the use of one word as opposed to use of the other.

All matrices are referenced with the row index first and the column index second. In the cases where there is a
time history (see definition of history variable below), the last index is the time index.

When the name of an array which contains a time history is given without any index for the time history
being specified, the most recent value is implied.

Operators

Throughout this specification, matrix operations (particularly multiplication) are required, and on some
occasions, non-standard operations are used upon matrices. The following symbols are used to denote the types of
multiplication to be applied.

Dots (·) Small dots are used to denote scalar multiplication. For example:

\[ 3 \cdot 4 = 12 \]

Multiplication sign (×) This symbol is used to denote standard matrix multiplication. This does
NOT imply a cross product, nor strictly a dot product. The definition of this type of
operation is given below:

\[ A \times B = C \]

where

\[ \forall i \text{ and } \forall j, \quad C_{ij} = \sum \limits_{k} A_{ik} \cdot B_{kj} \]

Asterisks (*) Asterisks are used in conjunction with index markers to show that the operations are
to be conducted on individual elements of arrays or vectors as if they were scalars. This is
often the case when calculating sensor values or other similar functions when multiple
scalars are grouped together for convenience. For example, the following equation is listed in
ASP:

\[ A_{ACCELERATION_M}(i) = A_{BIAS}(i) + A_{GAIN}(i) * A_{COUNTER}(i) \]
where \( i \) ranges from 1 to 3 and represents the three directions \( x, y, \) and \( z \). In this case, the first element of \( A_{ACCELERATION,M} \) would be calculated as follows:

\[
A_{ACCELERATION,M}(1) = A_{BIAS}(1) + A_{GAIN}(1) \cdot A_{COUNTER}(1)
\]

No Operator: In those cases where variables, matrices, or scalars are located directly beside each other with no operator between, standard multiplication is implied. Thus two matrices collocated would be multiplied as if they had the \( \times \) operator between them, while two scalars would be multiplied as if they had the \( \cdot \) operator between them. Also, if a scalar and a matrix (of one or more dimensions) were collocated, then the scalar would be multiplied by each element of the matrix and a new matrix of equal dimensions would be generated.

**DEFINITIONS**

**Implementation**

Computer code which fulfills all of the requirements outlined in the GCS Development Specification.

**Functional Unit**

Chapter 5 is divided into eleven subsections, each of which describes the requirements for a particular function to be performed by the GCS software. Throughout this specification, the term "functional unit" will be used to refer to one of these eleven functions. Note that there is not necessarily a one-to-one correspondence between a "functional unit" and a distinct unit or module of software code in an implementation.

**Frame**

A frame is the length of time necessary to execute all scheduled functional units. Each frame has two different time values associated with it. The first is the actual c.p.u. time that it takes to execute the GCS software on the simulation host computer, while the second is the allotted time for a frame on the actual lander. The global variable \( \text{DELTA}_T \) represents the time for one frame on the actual lander and is needed in the GCS code for the integration of the dynamic equations for the lander.

**Subframe**

A subframe is one of the three individual units of time which together make up a frame. The three subframes are named the Sensor Processing subframe (subframe 1), the Guidance Processing subframe (subframe 2), and the Control Law Processing subframe (subframe 3). In each frame, subframe 1 is executed first, subframe 2 is executed second, and subframe 3 is the last subframe executed.
Data Store

The definition for a data or control store given in Hatley [13] is "A data or control store is simply a data or control flow frozen in time. The data or control information it contains may be used any time after that information is stored and in any order." In this specification, all stores contain data, while some also contain data conditions. For the purposes of this specification, the term "data store" will be used to refer to any store which contains some combination of data and data conditions. Thus, all four stores listed in the Data Requirements Dictionary part II will be referred to as "data stores".

Global Data Store Variable

A global data store variable is any variable listed in any of the four global data stores in Chapter 6, namely GUIDANCE_STATE data store (Table 6.1), EXTERNAL data store (Table 6.2), SENSOR_OUTPUT data store (Table 6.3), or RUN_PARAMETERS data store (Table 6.4).

History Variable

Within this specification, a particular array, hereafter referred to as a "history variable" is one which contains a time history dimension; that is, it contains values for the current frame as well as for previous frames. The history variables are the following:

\[
\begin{align*}
&A_{\text{ACCELERATION}} (1:3,0:4) \\
&A_{\text{STATUS}} (1:3,0:3) \\
&A_{\text{AR ALTITUDE}} (0:4) \\
&A_{\text{AR STATUS}} (0:4) \\
&G_{\text{ROTATION}} (1:3,0:4) \\
&G_{\text{P ALTITUDE}} (0:4) \\
&G_{\text{P ATTITUDE}} (1:3,1:3,0:4) \\
&G_{\text{P VELOCITY}} (1:3,0:4) \\
&K_{\text{ALT}} (0:4) \\
&K_{\text{MATRIX}} (1:3,1:3,0:4) \\
&TDLR_{\text{VELOCITY}} (1:3,0:4)
\end{align*}
\]

In each case, the last dimension is the time dimension. The first subscript in a time history dimension is always declared to be zero. The time dimension contains a set of scalars, vectors, or arrays, depending on whether the total number of dimensions is one, two, or three, respectively. Let the term "object" denote a scalar, vector, or array, as appropriate for the particular variable. Each of these variables contains either four or five objects, depending on whether the last dimension is declared to be 0:3 or 0:4 respectively. The variable A_STATUS contains four objects, while each of the other time history variables contains five objects.

Each of the variables listed contains a most recent object and either three or four previous objects. The object with a time subscript of zero is the most recent object; the object with a time subscript of one is the object which is one frame older; the object with a time subscript of two is the object which is two frames older, etc.; the object with the largest time subscript (three or four) is the oldest object.
CONVENTIONS

FORTRAN Convention

This specification was written with the assumption that the implementation would be coded in FORTRAN. If the development language used is something other than FORTRAN, the programmer must investigate the possibility of differences between FORTRAN and the development language chosen.

REQUIREMENTS

Order of Processing

Within each functional unit in Chapter 5, the processing steps are given in a particular order. If the implementation uses the same order as that given in the specification, then correct results should be obtained; however, the programmer is free to use a different order as long as the change in order does not affect the outputs.

Calls to GCS_SIM_RENDEZVOUS

There must be a call to GCS_SIM_RENDEZVOUS prior to the execution of each subframe. See Chapter 2 and Appendix B for discussions regarding GCS_SIM_RENDEZVOUS.

Control Signals

The control signals listed in Table 6.5 in Part III of the Data Requirements Dictionary may be implemented by the programmer in any form desired, or they may be completely ignored and the control of the program may be conducted through other means.

Number Representations

When variables are given in sign-magnitude or other unusual formats, conversion or manipulation may be necessary.

Conversion of Units

It is the responsibility of the programmer to be sure that any implied conversion of units is performed.

Global Data Store Organization

Part II of the Data Requirements Dictionary contains descriptions of four required data stores. Each of these data stores is to be located in a separate, globally accessible data region. The division of the global data stores into four separate regions illustrates the fact these regions have a direct mapping to a specific implementation of GCS on hardware components of an actual lander. (See Figure B.1).

If the implementation is being written in FORTRAN, four labeled common blocks should be declared with the labels GUIDANCE_STATE, EXTERNAL, SENSOR_OUTPUT, and RUN_PARAMETERS, respectively (See Tables 6.1, 6.2, 6.3, and 6.4). The variables declared in each labeled common block must be in the same order as those in the corresponding table.
Use of Variables That Are Not in the Global Data Stores

A programmer may use variables in addition to the global data store variables; however, if the value of such a variable is dependent upon the values of any global data store variable(s), then the programmer should only use the value of such a variable in the same subframe of the same frame in which it was calculated.

Use of Tables

Some tables have the heading "CURRENT STATE" and "ACTIONS". If the actual state of the variables appears under the "CURRENT STATE" section in the table, then the actions listed in the same line are to be performed. If the actions in one line of the table are performed, then none of the actions in any other line of the table should be performed in the same subframe. If the actual current state is not represented in any line under the "CURRENT STATE" section of the table, then no action is to be taken.

Rotation of History Variables

In Chapter 5, in certain functional units, an instruction is given to "rotate" specific variables. Table 1.1 illustrates what is meant by rotation. The table is given for a variable with a time dimension of 0-4. For a variable with a time dimension of 0-3, the last line of the table should be ignored. Note that after the variable has been rotated, the new or current object is calculated and placed into the zeroth time history position.

Table 1.1 ROTATION OF VARIABLES

<table>
<thead>
<tr>
<th>TIME HISTORY SUBSCRIPT</th>
<th>VALUES BEFORE ROTATION</th>
<th>VALUES AFTER ROTATION</th>
<th>VALUE AFTER CALCULATIONS FOR CURRENT FRAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>O_{n-1}</td>
<td>X</td>
<td>O_n</td>
</tr>
<tr>
<td>1</td>
<td>O_{n-2}</td>
<td>O_{n-1}</td>
<td>O_{n-1}</td>
</tr>
<tr>
<td>2</td>
<td>O_{n-3}</td>
<td>O_{n-2}</td>
<td>O_{n-2}</td>
</tr>
<tr>
<td>3</td>
<td>O_{n-4}</td>
<td>O_{n-3}</td>
<td>O_{n-3}</td>
</tr>
<tr>
<td>4</td>
<td>O_{n-5}</td>
<td>O_{n-4}</td>
<td>O_{n-4}</td>
</tr>
</tbody>
</table>

Note: O_i denotes object that was calculated in frame i

n = current frame number

X = denotes that any value is acceptable

Precision

All calculations involving floating point variables should be done with precision equivalent to that of FORTRAN D-floating (REAL*8).
EXCEPTION HANDLING

During the execution of a computer program, exception conditions may sometimes occur. The implementation should anticipate or detect certain types of exception conditions and take specific actions. The relevant exception conditions and the actions to be taken are listed below.

Exception Conditions

**DIVIDE BY ZERO**
A division is performed, but the divisor is equal to zero.

**NEGATIVE SQUARE ROOT**
A square root is taken, but the argument for the square root is negative.

**UPPER OR LOWER LIMIT EXCEEDED**
The current value for a data element exceeds its upper or lower limit as specified in the range section in the Data Requirements Dictionary Part I.

Only certain data elements under certain conditions are to be checked for limits exceeded. The criteria for which elements are to be checked, in what context they are to be checked, and when they must be checked is as follows:

Which data elements:

A particular data element is to be checked for limits exceeded only if it is of data type REAL*8, and is in either of the two global data stores GUIDANCE_STATE or SENSOR_OUTPUT.

Context for check:

A data element is to be checked only when it is being used as an input. Rotation of a data element is not considered to be a use as an input for the purposes of limit checking. If the data element is a vector or array, then each element in the vector or array that is being used as input must be checked, including history values. It is not necessary for the functional unit CP to check any of its input data elements for limit exceeded.

When data element must be checked:

When an input data element is to be used or processed in a given subframe, then it must be checked sometime within that same subframe before it is used. If the data element is also being updated or changed in the same subframe before it is being used as an input, then it must be checked sometime between the time it is updated and the time it is used.
Action to be Taken for Each Specified Exception Condition

Write the appropriate output as specified below to the FORTRAN Logical Unit Number 6 and then continue. In the case of UPPER/LOWER LIMIT EXCEEDED, do not modify the data element. Note that to "continue" implies that the divide will be executed, or the square root will be taken, or the data element with exceeded limit will be used.

Output to be Generated for Each Exception Condition

The first line of the exception message should appear as follows:
"%EXCEPTIONAL-CONDITION-GCS-"<insert specific condition here>

where the specific condition is one of the following:
"DIVIDE_BY_ZERO"
"NEGATIVE_SQUARE_ROOT"
"LOWER_LIMIT_EXCEEDED"
"UPPER_LIMIT_EXCEEDED"

The second line of the exception message should contain the name of the functional unit where the exception condition occurred (i.e. AECLP, ASP, etc.), the name of the actual subroutine where the exception condition occurred, and the current value of the frame counter. Implementations that are coded in FORTRAN should use the following FORTRAN format statement:

FORMAT (x, a6, x, a32, x, i4)

A third line of the exception message containing information that is specific to the individual error type may be required as specified below.

Divide By Zero
No additional output necessary.

Negative Square Root
Display the value of the argument to the square root operation.
Use FORTRAN format statement FORMAT (x, e23.14).

Lower Limit Exceeded
Display the name of the data element in question and the value of the data element.
Use FORTRAN format statement FORMAT (x, a32, e23.14) for type real elements.

Upper Limit Exceeded
Display the name of the data element in question and the value of the data element.
Use FORTRAN format statement FORMAT (x, a32, e23.14) for type real elements.
2. LEVELS 0 AND 1 SPECIFICATION
LEVEL 0 SPECIFICATION

The GCS will provide an interface between the sensors (rate of descent, attitude, etc.) and the engines (roll and axial). The purpose of the GCS is to keep the vehicle descending along the predetermined velocity-altitude contour which has been chosen to conserve enough fuel to effect a safe attitude and touch down.

The GCS effects this control by:

- processing the following sensor information:
  - acceleration data from the three accelerometers -- one for each vehicle axis,
  - range rate data from four splayed doppler radar beams,
  - altitude data from one altimeter radar,
  - temperature data from a solid-state temperature sensor and a thermocouple pair temperature sensor,
  - rates of rotation from three strapped-down gyroscopes -- one for each vehicle axis, and
  - sensing of touch down by the touch down sensor.
- determining the appropriate commands for the axial and roll engines and the chute release mechanism and issuing them to keep the vehicle on a predetermined velocity-altitude contour.

The GCS also transmits telemetry data and synchronizes through a rendezvous routine (GCS_SIM_RENDEZVOUS) with GCS_SIM [10], the simulator and controller.

Note that implementations of the GCS developed from this specification may be executed singly or in parallel. Consequently, only specific system services can be used in an implementation. In particular, a rendezvous routine will be provided and should be invoked, as specified in the implementation notes in Appendix B. In addition, FORTRAN Intrinsic Functions may be used. Other system services and library routines are explicitly excluded from use by the programmer.

Figures 2.2 through 2.5, 3.1, 3.2, and 4.1 through 4.4, and Tables 2.1, 3.1, 4.1, and 4.2 follow Hatley's extension to Structured Analysis (see Appendix A), with the following exceptions and assumptions.

Exceptions:

1. Any data store may appear at more than one level because the processes specified do not communicate directly but only through data stores.
2. Any unlabeled flow between a process and a data store may not necessarily carry all the information in the data store (the actual flow content is defined by the process specification and the Data Requirements Dictionary Part II).
Assumptions:

1. The initial value for control signals is assumed to be "FALSE".
2. In a process activation table (PAT), an empty process cell indicates the process is deactivated.
3. In a PAT, an empty output cell indicates the control signal value remains unchanged.
4. In a PAT, output control signals receive values before any processes are activated and therefore may delay the activation of processes by deactivating their parent process.

An example of assumption 4 is Table 3.1 where setting RENDEZVOUS to "TRUE" delays the activation of the processes of which RUN_GCS is composed until GCS_SIM sets RENDEZVOUS to "FALSE".
Figure 2.1 STRUCTURE OF THE GCS SPECIFICATION
Figure 2.2 DATA CONTEXT DIAGRAM: LANDER
Figure 2.3 CONTROL CONTEXT DIAGRAM: LANDER
Figure 2.4 DATA FLOW DIAGRAM (DFD) 0: GCS