2008

Development of a design engineering in-situ sensor payload optimization tool

Keith Ronald Schreck
San Jose State University

Follow this and additional works at: http://scholarworks.sjsu.edu/etd_theses

Recommended Citation

This Thesis is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Theses by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.
DEVELOPMENT OF A DESIGN ENGINEERING IN-SITU SENSOR PAYLOAD OPTIMIZATION TOOL

A Thesis

Presented to

The Faculty of the Department of Mechanical and Aerospace Engineering

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Keith Ronald Schreck

May 2008
INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI®

UMI Microform 1458142
Copyright 2008 by ProQuest LLC.
All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest LLC
789 E. Eisenhower Parkway
PO Box 1346
Ann Arbor, MI 48106-1346
San Jose State University

The undersigned Faculty Committee Approves

Development of a Design Engineering In-Situ Sensor Payload Optimization Tool

of

Keith Ronald Schreck

APPROVED FOR THE DEPARTMENT OF MECHANICAL AND AEROSPACE ENGINEERING

Dr. Periklis Papadopoulos, Committee Chair  4/1/08

Dr. Nikos Mourtos, Committee Member  4/1/08

Mr. Nikola Djordjevic, Committee Member  4/6/08

Lockheed Martin Space Systems Company

APPROVED FOR THE UNIVERSITY

Pheo I. Williamson  6/17/08
ABSTRACT

DEVELOPMENT OF A DESIGN ENGINEERING IN-SITU SENSOR PAYLOAD OPTIMIZATION TOOL

By Keith R. Schreck

Determination of instrumentation for space science missions is an involved, complex procedure. Components are chosen to meet mission science requirements, creating an initial sensor package design. Design reviews iterate on the initial concept and options are evaluated until a final design solution is determined. Trade studies are traditionally performed at the component level. A final design solution is determined at the end of this often lengthy process. The analysis performed in this work looks at mission requirements and generates a mission sensor package using design engineering relations. Given a set of data to be returned from a science mission, a sensor package that meets mission requirements can be generated for a design solution. A tool for in-situ measurements is developed using systems engineering design relations to deliver a sensor payload configuration.
ACKNOWLEDGEMENTS

I would like to acknowledge Dr. Periklis Papadopoulos, Dr. Nikos Mourtos, and Mr. Nikola Djordjevic for their guidance and support in completing this thesis. I enjoyed the courses I had the opportunity to take under their tutelage, and enjoyed the diverse nature of material that was presented in their courses. I would like to thank them for their direction as well as their time dedicated to the completion of this work.

Additionally, I would like to thank those students in the AE 110 Space Systems Engineering course, – Michelle, Nick, Dai, Freddy, Kenny, Jad, John, Hingloi, and Ramon who aided in the research of commercially available products for the different sensor modules.

Finally, I would like to thank my friends and family, who have provided unwavering support of me in all my academic pursuits, and who have encouraged me through all my education.

Thank you all.
# TABLE OF CONTENTS

LIST OF TABLES........................................................................................................... x

LIST OF FIGURES....................................................................................................... xiii

I. INTRODUCTION................................................................................................. 1

II. METHODOLOGY................................................................................................. 2

III. ISSPO TOOL SUBROUTINES........................................................................... 4

   III.A ISSPO Program......................................................................................... 4

   III.B Input File................................................................................................. 6

   III.C Constants............................................................................................... 7

   III.D Data Type Verifier................................................................................. 7

   III.E Planetary Database............................................................................... 8

   III.F Sensor Databases.................................................................................. 11

      III.F.1 AC (Magnetic) Field....................................................................... 12

      III.F.2 Accelerometer................................................................................. 13

      III.F.3 Acoustics......................................................................................... 15

      III.F.4 Density............................................................................................ 17

      III.F.5 GCMS............................................................................................. 19

      III.F.6 Humidity......................................................................................... 24

      III.F.7 Inclinometers................................................................................ 25

      III.F.8 Nephelometer............................................................................... 27

      III.F.9 Optics.............................................................................................. 30

      III.F.10 Pressure........................................................................................ 34
IV.F.3 Temperature Sensor THP.........................80
IV.F.4 Acoustic Sensor API..........................82
IV.F.5 Permittivity PER...............................84
IV.F.6 Density DEN..................................85
IV.F.7 Refractive Index REF..........................86

IV.G ISSPO Huygens Summary........................89

V. VENUS ATMOSPHERIC PROPERTIES MISSION.........90

VI. SUMMARY..............................................95

VII. FUTURE WORK.........................................97

REFERENCES...............................................99

APPENDICIES

APPENDIX A: N2 Systems Diagram........................107

APPENDIX B: Huygens Probe Benchmark Results..........111

APPENDIX C: Venus Atmospheric Properties Mission Design........145

APPENDIX D: ISSPO Program Module Code..................158

D1. ISSPO Main Program..................................158
D2. Constants.............................................184
D3. Data Type Verifier..................................186
D4. Planetary Database..................................192
D5. AC Field Sensors...................................208
D6. Accelerometer Sensors...............................216
43. ISSPO HASI ACOUSTIC SENSOR INPUT FILE FORMAT.............72
44. ISSPO HASI ACOUSTIC SENSOR CONFIGURATION.................72
45. ISSPO HASI DSP COMPONENT INPUT FILE FORMAT.............74
46. ISSPO HASI DSP COMPONENT CONFIGURATION....................74
47. HUYGENS SURFACE SCIENCE PACKAGE OBJECTIVES..............75
48. ISSPO – HUYGENS SSP COMPONENT COMPARISON....................77
49. ISSPO SSP ACCELEROMETER INPUT FILE FORMAT.............78
50. ISSPO SSP ACCELEROMETER CONFIGURATION.......................78
51. ISSPO SSP TILT SENSOR INPUT FILE FORMAT....................79
52. ISSPO SSP TILT SENSOR SUMMARY CONFIGURATION..............79
53. ISSPO SSP THP SENSOR INPUT FILE FORMAT.....................81
54. ISSPO SSP THP SENSOR SUMMARY CONFIGURATION..............82
55. ISSPO SSP ACOUSTIC PROPERTIES INSTRUMENT SPEED OF
    SOUND INPUT FILE FORMAT........................................83
56. ISSPO SSP ACOUSTIC SENSOR CONFIGURATION.....................84
57. ISSPO SSP REFRACTOMETER SUMMARY CONFIGURATION..........88
58. VENUS ATMOSPHERIC MISSION OBJECTIVES......................92
59. VENUS ATMOSPHERE CONCEPT MISSION ISSPO INPUT
    FORMAT........................................................................93
60. ISSPO VENUS ATMOSPHERE MISSION SENSOR PACKAGE
    SUMMARY RESULTS........................................................94
LIST OF FIGURES

1. ISSPO Tool Flowchart Diagram .................................................. 5
2. Gas Chromatograph Reference Diagram ....................................... 20
3. Mass Spectrometer Reference Diagram ................................-------- 21
4. Simplified Nephelometer Reference Diagram .................................. 28
5. Schematic Diagram of Huygens Probe Instrument Layout ............... 52
6. Schematic Diagram of CCD Array Image Layout and Binning .......... 56
7. Schematic Diagram of Huygens GCMS Sensor ............................... 62
8. Schematic Diagram of Huygens HASI Sensor Package ..................... 64
9. Schematic Diagram of Huygens SSP Top Hat Sensor Cavity .............. 76
10. Schematic Diagram of Huygens SSP Component Layout ................. 77
11. Schematic Diagram of Huygens SSP THP Sensor ........................... 80
12. Schematic Diagram of Huygens SSP Permittivity Sensor Design ....... 85
13. Schematic Diagram of a Refractometer Prism ............................... 87
   A-1. ISSPO N² Systems Diagram .................................................. 109
   A-2. Atmospheric Suite N² Diagram ............................................. 110
I. Introduction

The driving interest behind the development of this program comes from material presented during a short course on In-Situ Instruments for Planetary Probes and Aerial Platforms hosted as part of the 4th International Planetary Probe Workshop [1]. Attendees were given a mass and power budget for a planetary probe mission that used an aerial platform, and were tasked to develop a sensor package that would meet the mission requirement and fit within the constraints. Long before a mission is launched, during the initial planning stages, a series of studies are conducted to create a sensor package custom tailored to meet the mission requirements. Components are chosen to survive the operating environment and meet mission requirements.

Design of the sensor payload package for any mission addresses several issues. The final optimal payload configuration is a result of individual case studies and design engineering studies. The scope of the tool, is limited to optimization techniques within the sensor payload, however, a higher system level criteria may impact the component level design, resulting in a different component selection. At some point, a human decision is still included in the design process, as final selection between competing elements is made. The decision to use one component over another can arise from many factors – functionality, heritage, Technology Readiness Level (TRL) [2], etc. The objective of this work is to combine all of the selection techniques for mission hardware into a single tool that can be used to generate a preliminary sensor package configuration.
II. Methodology

The component selection algorithms implemented in this tool trades sensor component characteristics (operational parameters such as range, performance, weight, accuracy, etc.) to arrive at an optimum component choice, based on a set of mission sensor requirements (e.g., planetary atmospheric data collection). Initial configurations are developed using top level mission requirements. As the solution for each sensor type progresses, the properties of the sensor are evaluated at finer levels of analysis. If a component no longer satisfies a requirement it is eliminated from analysis. If no suitable solution can be determined, a work-around strategy must be made to find a way to modify existing hardware to satisfy the mission requirements, either via a custom built specialized sensor, modifying a commercially available component to allow it to meet the requirement, or by making a modification to the mission requirement. The end result of this design tool for each type of mission science data is a unique commercially available sensor component.

A database of commercially available components is developed for each type of sensor. The down-selection process will employ several methods to eliminate incompatible sensors. Primary selection methods are based on the operational range of the sensor type (e.g., temperature range for Thermocouples, atmospheric gasses for mass spectrometers). Special consideration is given to heritage system components, to further select from multiple sensors that operate over similar operational ranges. Use of heritage materials implies a high level of technological development behind the sensor. The Technology Readiness Level (TRL) employed in a sensor's design, relates the
development level and risk associated with the hardware. While technology with a higher TRL is desirable, there are other advantages to a lower TRL device. A given device could be at a high TRL, but be a heavy component or involve a complicated mechanism. A similar device could have a lower TRL, but be significantly lighter. However, the lower TRL device has an increased level of inherent risk in its use, compared to a more mature design. Multi-role components can also be evaluated for their useful properties. These types of units complete the tasks assigned to multiple sensors with the benefit of a single unit capable of recording several data types. The use of these selection factors will allow for the determination of a component that will meet the mission sensor requirements.

As each component is selected for the sensor package, additional interactions between sensors will come into play. Design constraints may limit the use of certain types of sensors. Once a preliminary sensor design package has been completed, interactions between sensors at a system level may determine if any components are incompatible with other sensors. If this condition occurs, individual sensor requirements will have be modified and evaluated via another iteration with all the sensors until the sensors are compatible with each other and a final design solutions exists, or the program will determines that there is no commercially existing solution that meets the mission requirements. In this event a suitable solution sensor design will be a close result to the final sensor that will have to be custom modified.
III. ISSPO Tool Subroutines

The ISSPO Program is comprised of multiple subroutines being called from the main program. Each subroutine is called and returns specific pieces of data back to the main program. Sensor type subroutines are developed as a self contained model only requiring inputs from the main program to select the correct component or, when necessary, obtain data from another module to select the component. The subroutine for each sensor type is only called if a corresponding type of data is requested in the main program (e.g., ACCELERATION for acceleration data, OPTICS for imaging data, GAS ANALYZER for gas properties). This modular development allows for the program to include all sensor types, yet reduce running time to only relevant sensor types. Many planetary science missions feature a basic atmospheric properties sensor pack that monitor temperature, pressure, density, etc. Within the ISSPO tool these atmospheric sensors have been coupled into a single input option selection that calls all the individual sensors automatically. The function and properties of each of the ISSPO Tool subroutines is discussed here. Descriptions here are not meant as an exhaustive description of the design flow through each module, but to detail the key elements of the modules, limitations, design logic and the data used.

A. ISSPO Program

The ISSPO routine is the primary program call entered at the MatLAB command prompt. From here, the input data is loaded to the main program and the sensor payload is configured from the sensor modules called for in this program. Each module contains
its own set of variables needed by the program. At the end of each module, the relevant data is written to a ‘.mat’ binary data file. In the main ISSPO program, the data file is loaded into memory and the data is made available for all subroutines to use. A summary program flowchart in Figure 1 outlines the operation of the ISSPO Tool.

![ISSPO Program Flowchart](image)

Figure 1 ISSPO Tool Flowchart Diagram
The development and design of the ISSPO program is keyed toward a simple, minimal design. To save time in execution of multiple runs, all the program input variables are preloaded into a case data file with the MatLAB `.m` file extension. This allows the file to be evaluated as a program and it loads all the relevant case data. When ISSPO is executed from MatLAB, a brief introduction to the program is displayed and the user is asked to enter the name of the data file that contains the design information for that case. A logic check for the file is made and continues the program. If the file doesn’t exist, an error message is printed to the user and the program ends. Assuming the file exists and is set up correctly, the ISSPO program creates a case directory in a working directory folder. From there a program directory file folder is made and copies of the main program files are copied into the working directory folder. The main program then runs a planetary database program to obtain reference data values for the intended mission location; such as bulk parameters, orbital properties, and atmospheric data. Specific modules are executed for each sensor type depending on the required mission data.

**B. Input File**

The input file is not, in reality, a program executed as part of the ISSPO program. Instead it is intended to serve as a file to load the basic requirements of the sensor system design into the main program. The main benefit here is to quickly load all of the program inputs from a single file, saving the user from having to enter the data from scratch each time the program is executed. Each time a design change is made, only the variable has
to be updated in the input file. A default set of case inputs and required input data is included in the CaseData MatLAB file.

C. Constants

The constants subroutine is as straightforward as it sounds. Constant data values or conversion factors for different units are stored within two different data sets. These sets correspond to the chosen unit system. Data values and labels are stored for use by the various programs invoked from the main ISSPO program. All of the constants’ data properties are written to a MatLAB ‘.mat’ binary data file and loaded into memory after the program is executed.

D. Data Type Verifier

The Data Type Verifier program is a simple subroutine used to verify inputs. The type of sensor data to be collected by the mission is stored within a cell data array in the input file program. The Input File sensor data array is compared to a list of allowed data types in the ISSPO Program. This verifies the data entered for any misspellings or unknown data types. If the program finds a data type that is not within the program’s database, it prints out the unknown data type to screen, prints an error message, and terminates the program. Additional logic checks are performed on the input file for the different input sensor for any required additional information needed to select the final sensor. Required information varies with each sensor type and is discussed in each section.
E. Planetary Database

The primary purpose of this module is to provide data to the main ISSPO program on the planetary environment that the mission would encounter. Planetary data is crucial in the decision process of sensors for the mission. A database of all the planetary values is generated based on the object relevant parameters, and is stored in a data file loaded by the main ISSPO program to be used in selection of the sensor components. Intended as a planetary object database to determine the environment for a mission, the database provides enough information on planetary bodies that it can be of use to any interested researcher. To aid the common user, the program is sufficiently developed to be used as a component in the ISSPO program or serve as a stand-alone model with its own interactive command structure. From within ISSPO, the planetary database program is called to load the parameters into the main program.

The program consists of a database of planetary bulk parameters (e.g., mass, volume, radius, gravity), orbital parameters (e.g., period, velocity, orbital inclination) and atmospheric properties. Composition of the atmosphere includes major components by percent and trace element composition by particle concentration. These parameters are used to aid further along in the design process in selecting other components.

The planetary database currently contains the full set of properties of 27 celestial objects including the sun, the eight major planets, the moon, the recently reclassified minor planet Pluto, and the eight largest moons of Jupiter and Saturn. A sparse set of data is currently available from NASA’s Planetary Database for Jupiter’s and Saturn’s moons. With increased interest in these moons, the currently available data for these
objects is included in the database with temporary placeholder values for the unknown properties. The data for these bodies is taken only from NASA's Planetary Database [3] to represent a consistent planetary dataset. Planetary values are available from many sources online and in print; however, there are differences in the data that vary from source to source based on the measurements. Additional objects can be added to the database by simply including the additional parameters, no significant program modification is needed to handle the larger data array. Two versions of the database values exist within the planetary database program, one containing SI metric units, the other British Imperial units. With the planned upcoming missions to the asteroid belt and the outer planets, object data for the Saturn and Jovian moons can be added to the database. All of the data for the body is not needed to update the database. Placeholder values can be entered into the array based on analytical relationships or estimates until verified data values can be obtained.

While the main planetary data is stored in one database, the atmospheric properties are split into two different data arrays each using a different data structure. The first consists of the major atmospheric components and is stored in pairs of column elements. For each added element, two columns of data are needed to be accurately read into the program. The first location is a string value containing the element of the atmosphere (e.g., N, O, CO\textsubscript{2}) and the second is the numerical percent composition of the planet’s atmosphere (e.g., Earth ‘N\textsubscript{2}’ 78.084). Data values for the atmospheric compositions are determined from NASA’s Planetary Database and represent average values in the atmosphere. Certain planets exhibit large variations in the concentration of
compounds in the atmosphere over the course of the year, thus average values are used in
the database. Additionally small portions of the atmosphere are comprised of trace
elements or due to the uncertainty in the primary composition the total composition of the
atmosphere may not equal 100 percent.

For each planet, there are additional trace elements present in the atmosphere, but
not in sufficient amounts to be measured as a primary component in the atmosphere.
Trace components are stored in a secondary data array using a trinary column format.
Each trace component requires three columnar elements to describe its composition. The
first column is allotted to a data string containing the name of the gas, the same as the
major atmospheric component array. The second and third columns hold the number of
particles and the concentration, respectively. For example, on Earth the primary trace gas
is displayed as ‘Ar’ for argon in the first column, 9340 in the second column, and ‘ppm’
in the third indicating the concentration of argon gas in Earth’s atmosphere to be 9340
parts per million. Concentration amounts for the trace elements for this example are
expressed as a function of the number of particles in the atmosphere, but can also as
easily be expressed as a number per unit volume of the atmosphere (e.g., ppccm - parts
per cubic centimeter). The Planetary Database program does not actually place any
restriction on the concentration unit type in the array. This allows for future corrections
to the database to update the concentrations, or if the concentration is updated, to reflect a
different unit basis.

For any of the atmospheric elements, either major or trace, the component is
shown via its chemical formula, but could be rearranged to display the component’s
chemical or common name. This was merely done to save space in the database for storage, and to simplify the program output when displaying the components. Although no fixed database size is allocated for the atmospheric properties, an additional column space has been added to the major and trace gas arrays to allow for the addition of newly determined atmospheric components and also serves as a limit in the program when displaying data to the screen or writing the results to a summary file to be loaded later in the main ISSPO program.

F. Sensor Databases

The following sections detail the sensor types contained in the ISSPO Tool. Each database was developed in a similar manner. Inputs from the main ISSPO program are loaded into each sensor database file along with any requirements the sensor must be able to meet, based on the planet the mission is going to. Data arrays contain the sensor properties in a two-dimensional array format. Sensors are listed in row format, with each column designated to a property. The number of data arrays in each sensor database varies based on the number of available properties for each sensor type. The data arrays contain both numerical values and textual properties, such as power requirements of alternating or direct current, or comments on the use of the sensor in certain environments or other restrictions. The final selected sensor properties are recorded to a MatLAB ‘.mat’ data file and saved for use by the main ISSPO program.
1. AC (Magnetic) Field

The AC_FIELD_SENSORS program loads information on magnetic field sensors used to map an object’s magnetic field. The magnetic fields on several planets are understood, but others are too small to maintain a magnetic field. The strength of the field helps to answer questions about the nature of the planet’s history and if it were ever able to support life. A planet’s magnetic field helps to protect the surface from the sun’s harmful radiation and can absorb and block high energy cosmic radiation. By mapping the object’s magnetic field, the levels of surface radiation can be determined. Missions to the Asteroid Belt, smaller moons, and comets reveal important information about the nature of the object’s core. Additionally, the shape of the field around small objects needs to be understood to safely land small probes and how the probe will react to the object’s field.

Magnetic field sensor data [4, 5, 6] is recorded into several data arrays. Properties that have common units in both unit systems are recorded into a single data array using rows for each sensor, and columns for each separate property. Sensor properties that depend on the chosen unit system are loaded into separate data arrays and concatenated onto the common property arrays. A final array containing comments on each sensor adds any relevant data about the sensor into the database, which can be searched for specific keywords to aid in the selection process for a compatible sensor. A planet’s magnetic field strength is tracked in the PLANETARY_DATABASE program and the existence or strength of the field is input to the magnetic field sensor program, along with planetary atmospheric properties that aid in the selection of a sensor based on the
environment the sensor is subject to. The properties shown in Table 1 are recorded for each sensor and stored within the database.

Table 1  AC (Magnetic) Field Sensor Data recorded in ISSPO Tool.

<table>
<thead>
<tr>
<th>AC Field Sensor Database Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Common Sensor Properties</strong></td>
</tr>
<tr>
<td>Sensor Type</td>
</tr>
<tr>
<td>Low Magnetic Field Limit</td>
</tr>
<tr>
<td>Operational Frequency Range</td>
</tr>
<tr>
<td>Sampling Time Interval</td>
</tr>
<tr>
<td>Voltage Requirements</td>
</tr>
<tr>
<td><strong>Unit System Specific Properties</strong></td>
</tr>
<tr>
<td>Sensor Operational Temperature Range</td>
</tr>
<tr>
<td>Electronics Operational Temperature Range</td>
</tr>
<tr>
<td>Sensor Dimensions</td>
</tr>
<tr>
<td>Sensor Weight</td>
</tr>
<tr>
<td>Sensor Comments</td>
</tr>
</tbody>
</table>

2. Accelerometers

The ACCELEROMETER_SENSORS program loads database properties for acceleration sensors into the ISSPO Program. Accelerometers are used to map atmospheric profiles during planetary entry. Acceleration profile data can be used to determine the mass of the science mission object. There is a wide variety of available accelerometers for a number of uses. Accelerometer data is also a key parameter used to determine sequence timings during planetary entry. Events during planetary entry are triggered by accelerometer data, such as parachute deployment, and retro rocket firings to slow landers to safely arrive on the surface. Often multiple sensors are used at different
locations to obtain differential accelerations on the spacecraft to keep it stable, and act as redundant systems to offset any erroneous data recorded by a single accelerometer, when used as event triggers. Sensors [7, 8] are available with a wide range of performance, from high acceleration limits at low fidelity to small acceleration limits at high fidelity. Common properties for each sensor are recorded into a single data and appended to by data from a secondary data array holding values specific to the chosen unit system. Many of the available sensors can be ordered to meet several different acceleration ratings with different sensitivity. For these sensors the highest available rating was input into the database with a comment in the data that it is available at other calibrations. The different calibrations for a single sensor don’t affect the dimensions or mass of the sensor.

The selection of an accelerometer is guided in part by the nature of the mission’s objectives. Mission with low acceleration tolerances will usually require a high sensitivity. In contrast missions requiring high accelerations, a lower sensitivity is available. A required input to the ISSPO program when acceleration data is required is the type of accelerometer data required, whether for a mission with soft accelerations, high accelerations or extreme acceleration profiles like NASA’s Deep Impact. The mission profile type is entered into the second column of the sensor data variable as ‘SOFT’, ‘MEDIUM’, ‘HIGH’, ‘IMPACT’, or ‘BALLISTIC’ in the input file. Data on the power supply type to the unit is entered into the third column. A summary of the recorded values in the database is shown in Table 2. Proper input file format for acceleration sensors is shown in Table 3.
Table 2 Accelerometer Sensor Data recorded in ISSPO Tool.

<table>
<thead>
<tr>
<th>Accelerometer Database Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Sensor Properties</td>
</tr>
<tr>
<td>Sensor Type</td>
</tr>
<tr>
<td>Sensitivity</td>
</tr>
<tr>
<td>Linearity</td>
</tr>
<tr>
<td>Shock Limits</td>
</tr>
<tr>
<td>Current Range</td>
</tr>
<tr>
<td>Dynamic Range</td>
</tr>
<tr>
<td>Operational Frequency Range</td>
</tr>
<tr>
<td>Resonance Frequency</td>
</tr>
<tr>
<td>Circuit Configuration</td>
</tr>
<tr>
<td>Voltage Requirements</td>
</tr>
<tr>
<td>Electrical Signal Noise</td>
</tr>
<tr>
<td>Unit System Specific Properties</td>
</tr>
<tr>
<td>Sensor Temperature Range</td>
</tr>
<tr>
<td>Sensor Weight</td>
</tr>
<tr>
<td>Sensor Dimensions</td>
</tr>
<tr>
<td>Temperature Sensitivity Correction</td>
</tr>
</tbody>
</table>

Table 3 Acceleration Sensor Input File Format

<table>
<thead>
<tr>
<th>1st Column</th>
<th>2nd Column</th>
<th>3rd Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘ACCELERATION’</td>
<td>‘LOW’</td>
<td>‘Voltage’</td>
</tr>
<tr>
<td>‘MEDIUM’</td>
<td>‘Const Current’</td>
<td></td>
</tr>
<tr>
<td>‘HIGH’</td>
<td>‘Self Generating’</td>
<td></td>
</tr>
<tr>
<td>‘IMPACT’</td>
<td></td>
<td></td>
</tr>
<tr>
<td>‘BALLISTIC’</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Acoustic Sensors

The ACOUSTIC_SENSORS program loads acoustic sensor data into the ISSPO program. Acoustic sensors [9, 10] have several different functions and can monitor several different aspects of the mission. A microphone-based acoustic sensor can be used to listen to the sounds of the mission location. They are also useful for monitoring vibration on components onboard the vehicle. Moving parts on the vehicle can send information in their vibration and be used to send back information on the health of the
part. Flow detection-type acoustic sensors can be used to monitor blockages in pipes in soil or air sampling. An acoustic sensor can also be used to measure particle velocity and sound pressure in three orthogonal directions. Recorded database properties for the acoustic sensors are shown in Table 4.

<table>
<thead>
<tr>
<th>Acoustic Sensor Database Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Common Sensor Properties</strong></td>
</tr>
<tr>
<td>Sensor Type</td>
</tr>
<tr>
<td>Frequency Range</td>
</tr>
<tr>
<td>Resonance</td>
</tr>
<tr>
<td>Clipping Limit</td>
</tr>
<tr>
<td>Current Range</td>
</tr>
<tr>
<td><strong>Unit System Specific Properties</strong></td>
</tr>
<tr>
<td>Operational Temperature Range</td>
</tr>
<tr>
<td>Storage Temperature Range</td>
</tr>
<tr>
<td>Temperature Coefficient</td>
</tr>
<tr>
<td>Sensor Dimensions</td>
</tr>
</tbody>
</table>

Due to the wide array of different uses and configurations for acoustic sensors, there is a smaller amount of consistent property data amongst different models. Since acoustic sensors are capable of operating in different modes additional input is required to determine the specific use of the sensor in that case. The type of operation is entered into the input file in the second column position as either, ‘SENSOR’, ‘ARRAY’, or ‘VELOCITY’. A sample of the input file format is shown in Table 5.

<table>
<thead>
<tr>
<th>Table 5 Acoustic Sensor Input File Format</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1st Column</strong></td>
</tr>
<tr>
<td>‘ACOUSTICS’</td>
</tr>
<tr>
<td>‘ARRAY’</td>
</tr>
</tbody>
</table>
4. Density Sensors

The DENSITY_SENSORS program loads and selects the correct density sensor needed to survive the environment. This module is called when the atmospheric data suite option is chosen from the sensor data array. Inputs to this program are the chosen unit system and the planetary bodies’ surface atmospheric density. Using a general definition from Wikipedia [11], the densitometer is basically a light source aimed at a photoelectric cell, which determines the density of the sample from differences in the readings. Understanding of the atmospheric profile provides valuable data into seasonal fluctuations in the atmospheric properties. By understanding the seasonal fluctuations, future missions can predict how the atmospheric profile will affect the trajectory of an entry probe or lander system.

There are a variety of units with different applications [12], material, features, construction, and other specifications. Recorded data values for the sensors are specific to the chosen unit system and loaded upon program execution. Comments included with the data array provide additional information on the use and applications of the densitometer. Due to larger sizes, density sensors can tend to be heavy and consume a higher portion of a mission’s science equipment budget. As an alternate solution, given the object’s atmospheric composition, atmospheric density can be determined from other known planetary properties via the Ideal Gas Equation in Equation 1.

\[ \rho = \frac{P}{RT} \]  

(1)
Planetary Atmospheric properties are assigned in the PLANETARY_DATA-BASE subroutine. A logic check within the program determines if an atmosphere exists, and will calculate an approximate density based on the Ideal Gas Equation and override a zero input value for the atmospheric density, if it is unknown. The database contains gas constants for different gasses, and the value used is based on the gas constant for the primary constituent in the atmosphere. This incorporates an element of error into the planetary atmosphere density calculation that varies based on the number of known constituents in the atmosphere. The planetary atmospheric density value is fed into the density sensor program to determine a sensor capable of operating in the atmosphere. Database properties for the density sensors are shown in Table 6.

<table>
<thead>
<tr>
<th>Density Sensor Database Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit System Specific Properties</td>
</tr>
<tr>
<td>Sensor Type</td>
</tr>
<tr>
<td>Accuracy</td>
</tr>
<tr>
<td>Temperature Range</td>
</tr>
<tr>
<td>Weight</td>
</tr>
<tr>
<td>Voltage Requirements</td>
</tr>
<tr>
<td>Vibration Limits</td>
</tr>
<tr>
<td>Viscosity</td>
</tr>
</tbody>
</table>

In development of mission plane for the outer planets, there is a fair amount of uncertainty of the exact nature of the planetary surface material, whether the vehicle will land on solid ground, a frozen slush, or a vast ocean of liquid. To aid the ISSPO Tool in selection of a sensor suited to the particular application, knowledge of the operating
environment is required. Sensors are chosen based on sampling gas density, i.e., during atmospheric entry, or liquid, if the sensor is to take samples on the ground. By default the density sensor module is called as part of the "ATMOSPHERIC" sensor package suite. To modify the type of sensor selected for a particular application, the ISSPO tool needs to know which sensor package to associate the modification with. To modify density sensors, enter the term "DENSITY" into any column following the ATMOSPHERIC sensor package call. In the following column, enter the type of density sensor required, either "GAS," or "LIQUID." The program will err off if it detects the "DENSITY" modifier flag in the input variable array without a sensing medium type. If the "DENSITY" flag is not entered into the ATMOSPHERIC sensor package call, the program defaults to "GAS" sensing type and continues the program. A sample input format for a density sensor is shown in Table 7.

<table>
<thead>
<tr>
<th>1st Column</th>
<th>ANY Subsequent Column</th>
<th>Next Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ATMOSPHERIC&quot;</td>
<td>&quot;DENSITY&quot;</td>
<td>&quot;GAS&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;LIQUID&quot;</td>
</tr>
<tr>
<td>&quot;ATMOSPHERIC&quot;</td>
<td></td>
<td>&quot;GAS&quot;</td>
</tr>
</tbody>
</table>

5. **GCMS Sensors**

The GCMS_SENSORS program loads Gas Chromatograph Mass Spectrometer (GCMS) sensor data into the ISSPO program. These units are typically a combination of two different sensor components: Gas Chromatograph and Mass Spectrometers. These
two components work together to perform similar tasks but operate via different methods. Both components determine the chemical makeup of a gas or solid sample by breaking up the samples into its smallest components and determining the amounts of the different chemicals to known test samples. These units are calibrated periodically to verify that consistent results are obtained. By analyzing the output of these devices the composition of the original sample can be precisely determined. In Gas Chromatography [13], an unknown gas sample is carried into a sampling chamber along with an inert carrier gas and heated. As the gas is heated the different components begin to ionize into its simplest elements. The ionized particles are tracked as they leave the sampling chamber and recorded for the amount of time within the oven and their exit temperature as these can be used to determine the composition of the sample.

![Diagram of Gas Chromatograph](http://en.wikipedia.org/wiki/Gas_chromatography)

In Mass Spectrometry [14], the object under analysis is often a solid sample instead of a gas. Here an ion source is used to bombard the surface of the sample and ionizes the sample. The beam of the resultant out-gassed particles is then sent through a magnetic field to deflect the particle stream. A detector at the far end measures the
amount of the beam deflection and correlates the amount of deflection with a specific element. Lighter elements experience a greater amount of deflection in the magnetic field. A charge to mass ratio is then determined by the detector and plotted to show the intensity on the detector over different charge to mass ratios. An operational diagram of a mass spectrometer is shown in Figure 3.

![Mass Spectrometer Reference Diagram](http://en.wikipedia.org/wiki/Mass_spectrometry)

Figure 3 Mass Spectrometer Reference Diagram [14]


Intensity signal spikes indicate the presence of different chemical elements. For each known compound a unique series of peaks is generated. The intensity data is recorded and transmitted back to scientists in a laboratory that compare the data to known compounds to determine the sample's original components and concentration. Use of these two sensors in combination allow for the determination of both atmospheric and ground samples.
The use of a GCMS sensor system in space missions is still fairly limited, so the reliability of space rated units is fairly limited. A unit's volume, mass, and power are all at premiums on any space mission, so efforts are made to reduce these parameters when selecting components. These types of units are often mission specifically built to meet mission's requirements. A large number of different models exist, but are primarily used in laboratory setups, and thus not as restricted by mass, volume and power requirements. The data for these sensors [15, 16] stored in the ISSPO database records some of the key selections parameters for a viable unit and records additional information on the performance capabilities of the sensors in the comments section.

Two different operational types exist for these devices. The select the correct operational type within the ISSPO program additional information is required in the case input file. Their operation and specifications are tied the manner under which they report the results of the analysis. One type displays the wavelengths and intensity of the of the spectral emission band by filtering the samples gasses. By observing the wavelength and intensity of the samples spectral peaks, the exact chemical composition of the unknown sample can be determined by comparing the results against known gas samples. The other type uses a charge to mass ratio to determine the samples composition. To select the different types of sensor to use enter "WAVELENGTH," or "MASS-CHARGE" in the 2nd column position in the case input file. A sample of the input format is shown in Table 8. If the mass charge option is used, an additional range value of "LOW", "MEDIUM", or "HIGH" is required in the third column position in the input file.
Table 8 GCMS Sensor Input File Format

<table>
<thead>
<tr>
<th>1st Column</th>
<th>2nd Column</th>
<th>3rd Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;GAS ANALYSIS&quot;</td>
<td>&quot;MASS-CHARGE&quot;</td>
<td>&quot;LOW&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;MEDIUM&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;HIGH&quot;</td>
</tr>
<tr>
<td>&quot;GAS ANALYSIS&quot;</td>
<td>&quot;WAVELENGTH&quot;</td>
<td></td>
</tr>
</tbody>
</table>

The chemical composition of the planetary atmosphere is loaded via the PLANETARY_DATABASE program and input to the GCMS SENSOR program. Each component of the atmosphere emits absorption spectrum lines at distinct wavelengths. The strong signal peaks for each component of the atmosphere are loaded into a data array and used to determine the minimum and maximum wavelengths for the components in the atmosphere. These values are then used to select a sensor whose operational spectral range covers all the components in the atmosphere. Units based on detection of wavelength spectral peaks allow for the detection of complex molecules beyond simple elements, and offer the ability to determine the presence of tholins or organic components within a planetary atmosphere. Recorded database parameters are shown in Table 9.

Table 9 GCMS - Wavelength Sensor Data recorded in ISSPO Tool.

<table>
<thead>
<tr>
<th>GCMS Sensor Database Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit System Specific Properties</td>
</tr>
<tr>
<td>Sensor Type</td>
</tr>
<tr>
<td>Spectral Width</td>
</tr>
<tr>
<td>Scan Time</td>
</tr>
<tr>
<td>Weight</td>
</tr>
<tr>
<td>Voltage</td>
</tr>
</tbody>
</table>
Selection of the "MASS-CHARGE" option loads sensor properties from a different database and contain different properties. A separate set of selection criteria is used to determine the correct sensor. Use of mass charge based analyzers are best suited towards missions monitoring basic atmospheric elements, and are less well suited to analyze the high molecular weight of organic compounds in a planetary atmosphere. Database properties for these configurations are shown in Table 10.

<table>
<thead>
<tr>
<th>GCMS Sensor Database Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unit System Specific Properties</strong></td>
</tr>
<tr>
<td>Sensor Type</td>
</tr>
<tr>
<td>Mass Filter Material</td>
</tr>
<tr>
<td>Dynamic Scan Range</td>
</tr>
<tr>
<td>Sensitivity</td>
</tr>
<tr>
<td>Number of Ion Sources</td>
</tr>
<tr>
<td>Filament Material</td>
</tr>
<tr>
<td>Electron Energy</td>
</tr>
<tr>
<td>Focus Voltage</td>
</tr>
<tr>
<td>Dimensions</td>
</tr>
<tr>
<td>Input Voltage</td>
</tr>
<tr>
<td>Current</td>
</tr>
<tr>
<td>Power Requirements – Min/Max/Mean</td>
</tr>
</tbody>
</table>

6. **Humidity Sensors**

The HUMIDITY_SENSORS program loads humidity sensor data into the ISSPO program. This module is called when the atmospheric data suite option is chosen from the sensor data array. Humidity sensors [17] track the amount of water vapor in the
planets atmosphere. Several of the other sensor types have constraints on the relative humidity that they will operate in. For most of the known objects in our solar system, the relative humidity of the atmosphere is a minor concern as only Earth has high water vapor content in the atmosphere. Since many planetary objects do not contain water vapor in any great quantity, inclusion of this sensor may not be required to obtain a better model of the planet’s atmospheric properties. However, humidity sensors are small and lightweight, so it would not use much of the mission’s mass, volume, and power budget to include a sensor in the design package. For planetary objects that do not contain any known amounts of water vapor in the atmosphere, the inclusion of humidity sensors is not required and the sensor design is overridden to zero values. Humidity sensor data arrays are created based on the chosen unit system, and the recorded properties are shown in Table 11.

Table 11 Humidity Sensor Data recorded in ISSPO Tool.

<table>
<thead>
<tr>
<th>Humidity Sensor Database Properties</th>
<th>Unit System Specific Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor Type</td>
<td>Relative Humidity Range</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>Hysteresis</td>
</tr>
<tr>
<td>Response Time</td>
<td>Stability</td>
</tr>
<tr>
<td>Temperature Range</td>
<td>Frequency Range</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Weight</td>
</tr>
</tbody>
</table>

7. **Inclinometer Sensors**

The INCLINOMETER_SENSORS program loads inclinometer sensor data into the ISSPO program. This sensor allows for the determination of tilt angles from the
vertical or horizontal plane. Tilt sensors can provide valuable information about planetary surface terrain and orientation to lander or rover systems. Rovers need information about the local terrain conditions to determine if it is able to maneuver around steep grade terrains. Use of tilt sensors can provide local terrain information and determine if the rover has sufficient power to climb out of craters or on steep slopes traversing over hilly terrain. Sensors selection is driven in part by the application and the expected operational range the sensor is expected to encounter. Selection of a sensor for an entry probe would differ from that for a lander vehicle. During planetary entry the probe can tumble through high angles or orientations, whereas a lander vehicle will travel over fairly level terrain and is not required to be able to tilt to large angles.

The sensor data recorded [18, 19, 20] comes form many industrial use tilt sensors. There are a wide variety of configurations available with multiple options. Additional features and unit specifications are included in the database. Units are available with multi axis detection abilities, various voltage requirements, and computer data interfaces. Desired sensor ranges for the chosen sensor mission are entered as an additional input into the second column of the sensor data array in the input file as either “LOW,” “MEDIUM,” or “HIGH.” Inclinometer sensor data arrays are created based on the chosen unit system, and the recorded properties are shown in Table 12. A sample of the input file format for inclinometers is shown in Table 13.
Table 12 Inclinometer Sensor Data recorded in ISSPO Tool.

<table>
<thead>
<tr>
<th>Inclinometer Sensor Database Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unit System Specific Properties</strong></td>
</tr>
<tr>
<td>Sensor Type</td>
</tr>
<tr>
<td>Output Voltage Range</td>
</tr>
<tr>
<td>Sensitivity</td>
</tr>
<tr>
<td>Input Voltage</td>
</tr>
<tr>
<td>Bandwidth</td>
</tr>
<tr>
<td>Storage Temperatures</td>
</tr>
<tr>
<td>Weight</td>
</tr>
</tbody>
</table>

Table 13 Inclinometer Sensor Input File Format

<table>
<thead>
<tr>
<th>1st Column</th>
<th>2nd Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;INCLINATION&quot;</td>
<td>&quot;LOW&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;MEDIUM&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;HIGH&quot;</td>
</tr>
</tbody>
</table>

8. **Nephelometer Sensor**

The NEPHELOMETER_SENSORS program loads nephelometer sensor data into the ISSPO program. This sensor allows for the measurement of suspended particles in a gas. These make it possible to obtain atmospheric data such as visibility and particle size in the atmosphere. There are few commercially available nephelometers [21, 22]. Most space missions that would require one onboard are likely to be custom designed for that particular mission. Size and power requirements of commercially available nephelometers make them impractical for use in space exploration missions. Smaller, more power efficient sensors are producible, but are usually custom designed on an ‘as needed’ basis.
The sensor models included in the database are based on an integrating nephelometer design as shown in the simplified operating schematic in Figure 4. A sample chamber is filled with an atmospheric sample and analyzed over a time period while the particles in the chamber are counted. At the end of the integration cycle, the sample is expelled out of the chamber and is ready to begin again. While the sample is being recorded, light is used to illuminate the sample while a series of filters determines the amount of light blocked by the sample. Certain models can offer the advantage of additional atmospheric sensor data recorded during the sample. The units include pressure, temperature, and humidity sensors within the test chamber. However, these units incorporate a complex design and contain many moving parts that increase risk involved with its operation due to failure or clog. The design incorporates inlet and exit valves and an internal fan to maintain the particles suspended in the air. Additionally, the built-in pressure, temperature, humidity sensors have reduced operational ranges to those of individual sensors. The possibility exists to upgrade these internal sensors and re-qualify its capability to satisfy the mission requirements.

Figure 4. Simplified Nephelometer Reference Diagram
Due to the large mass, volume and power requirements of nephelometer sensors, several other design considerations must be taken into account. During the design phase, it must be determined whether there is enough budget in the design to accommodate this sensor package. For smaller spacecraft, the mission planners may have to consider the workload time for this sensor, as the electrical power system may not be able to power all the components all the time. Several other sensors may have to be shutdown for a period of time to divert enough power to the nephelometer for it to function.

Commercially available nephelometers are designed as stand-alone units, with an integrated power supply unit. These units can be removed to reduce the sensors weight volume, and power requirements. The unit can then be directly connected to the spacecraft’s power distribution system. Nephelometer sensor properties tracked in the database are shown in Table 14.

Table 14 Nephelometer Sensor Data recorded in ISSPO Tool.

<table>
<thead>
<tr>
<th>Nephelometer Sensor Database Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nephelometer Properties</td>
</tr>
<tr>
<td>Sensor Type</td>
</tr>
<tr>
<td>Sensor Bandwidth</td>
</tr>
<tr>
<td>Power Requirements</td>
</tr>
<tr>
<td>Dimensions</td>
</tr>
<tr>
<td>Sensor drift</td>
</tr>
<tr>
<td>Sample Flow Rates</td>
</tr>
<tr>
<td>Relative Humidity Limits</td>
</tr>
<tr>
<td>Power Supply Properties</td>
</tr>
<tr>
<td>Weight</td>
</tr>
<tr>
<td>Voltage</td>
</tr>
</tbody>
</table>
9. **Optic Sensors**

The IMAGING_SENSORS program loads and selects the correct optical imaging sensor data into the main ISSPO program. Optical imaging cameras are a staple component of nearly all planetary missions. The ability to record images from another planet and relay back data has truly given us insight to what occurs on other worlds. Imagers have become so crucial, that often multiple sensors are used to take a variety of different image formats. Optical imagers are not only used to relay images back to earth but also to “see” obstacles in its path and be able to determine how to maneuver around objects by analyzing the image data. Optical imagers can provide a wealth of information beyond the visible image. Given a reference object length in an image, the size of geographic formations can be determined. Observing geologic formations can provide insight into the past history of the planet and indicate how the planet environment is changing over time. Recording variations in terrain over seasonal periods provide additional valuable information on environmental cycles on a planet.

Imaging sensor units have a wide variety of capabilities and operating ranges. Use of different lens types allow for wide field images, panoramic, fish eye, magnification, etc. Commonly used imagers feature a black and white format, but the operating range can be extended to other regions of the spectrum via the use of color filters to record images in each color spectrum later be combined to create a single false color image. Use of additional filters can extend the imagers range to include the ultraviolet, infrared and x-ray regions of the electromagnetic spectrum. These extended
capabilities make use to imaging systems a highly valued component for many planetary missions.

Selection of optical system components can also have profound effects on the design of the overall system. Digital imaging systems typically use either a CCD or a CMOS sensor to detect an image. Recently imaging sensor array sizes have grown significantly and deliver several mega pixel image resolution. A tradeoff here is made between the camera resolution size and the data relay system used on the spacecraft. For a fixed rate data communications system, the designer can choose a high resolution camera that will fill up the available data bandwidth quickly, with a limited number of images. Alternatively, a low resolution array will only be able to image a small area at a time however; the system would be able to handle the bandwidth of all the small images. A similar situation arises in sizing on board memory for the mission, to store a number of high resolution images requires a large amount of memory, which must be factored into the system design and will use up portions of the spacecraft’s mass, volume and power budget. Each of these factors must be taken into account in the overall system design.

A common practice in including imagers in mission payloads is the use of multiple imagers, with several different types. On board the Mars Exploration Rovers, is a suite of optical imaging camera types [23] that aide in the collection of science data and help maneuver the rovers around obstacles. Five different imaging camera systems are used for a variety of purposes. The PANCAM array consists of two panoramic cameras situated atop the mast and is capable of recording 360 degree azimuth and +/- 90 degree elevation range. A pair of NAVCAMS sits atop the mast next to the PAMCAM and
provides a wide field of view to aid in driving the rovers. A micro-imager camera is mounted at the tip of the extendible arm and allows for close up images to be taken of the geology and terrain surrounding the rover. Hazard cameras are mounted on the front and rear of the rover under the solar panels with a view looking at the front and rear wheels. These lenses aid in steering the rovers around small obstacles near the wheels. A SUNCAM CCD array is used to provide an accurate orientation measurement of the rover’s position on Mars relative to Earth to indicate the direction to aim the high gain antennae to obtain the best signal to Earth. Most of these cameras are custom built units that were required to operate in a harsh environment. Several sensors on the exterior surface are comprised solely of a CCD imaging array and a optical lens to resolve the image. Each of these sensors provides additional functions to the rover and was selected carefully to meet the requirements of its function.

Selection of the appropriate CCD array to capture the image is only a part of sensor. The array must be coupled to an appropriate camera lens to obtain the proper image type. Data on sensors [24, 25, 26] entered into the ISSPO database is based on a CCD array, or complete system with array and camera lens. Selection of the optical sensor option in the input file requires three additional columns of input in the same sensor row. The second column represents the type of system desired, and is entered either as “CAMERA,” or “ARRAY,” or “LINEAR.” The third column corresponds to the desired resolution of the sensor as either “LOW,” “MEDIUM,” or “HIGH.” Additionally required by the program is the desired type of image to be taken. There are many different types of image range available that extend beyond the visual spectrum.
Additional input in the fourth column is the desired type of image to be returned. Valid inputs to the optical sensor program are “X-RAY,” “VISUAL,” “UV,” “NIR,” and “MICRO.” These terms correspond to the x-ray, visual, ultraviolet, near-infrared, and microscopic images. The desired image type for the optical sensor can allow for multiple image type sensor configurations. Enter the required image type into additional columns on the optical sensor row within the sensor data variable in the input file to configure the correct sensor. Since most of the sensor elements are common between the use of a CCD array and a complete camera system, a single database exists within the program. Optical imaging sensor properties tracked in the database are shown in Table 15. A sample of the input format required for the optical sensor program is shown in Table 16.

### Table 15 Optical Imaging Sensor Database Properties

<table>
<thead>
<tr>
<th>Sensor Type</th>
<th>Optical Range Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCD Array Dimensions</td>
<td>Pixel Size</td>
</tr>
<tr>
<td>Image Area</td>
<td>Signal Read Noise</td>
</tr>
<tr>
<td>Full Well Capacity</td>
<td>Gain</td>
</tr>
<tr>
<td>Linearity</td>
<td>ADC Dynamic Range</td>
</tr>
<tr>
<td>Readout Rates</td>
<td>Readout Time</td>
</tr>
<tr>
<td>Frame Rate</td>
<td>Dimensions</td>
</tr>
<tr>
<td>Operating Temperature</td>
<td>Cooling Temperature</td>
</tr>
<tr>
<td>Cooling Method</td>
<td>ADC Dynamic Range</td>
</tr>
<tr>
<td>Weight</td>
<td></td>
</tr>
</tbody>
</table>

### Table 16 Optical Sensor Input File Format

<table>
<thead>
<tr>
<th>1st Column</th>
<th>2nd Column</th>
<th>3rd Column</th>
<th>&gt;4th Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>“OPTICS”</td>
<td>“CAMERA”</td>
<td>“LOW”</td>
<td>“X-RAY”</td>
</tr>
<tr>
<td>“ARRAY”</td>
<td>“MEDIUM”</td>
<td>“HIGH”</td>
<td>“VISUAL”</td>
</tr>
<tr>
<td></td>
<td>“UV”</td>
<td></td>
<td>“NIR”</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>“MICRO”</td>
</tr>
</tbody>
</table>

33
10. *Pressure Sensors*

The PRESSURE_SENSORS program loads and selects the correct pressure sensor needed to survive the environment. This module is called when the atmospheric data suite option is chosen from the sensor data array. Inputs to this program are the chosen unit system and the planetary body's surface pressure. There exists a myriad of pressure sensors with different calibrations, functions and measurement types [27, 28, 29, 30, 31]. The selection of hardware is highly driven by the application. Pressure sensors can operate in several different types – vacuum, gage, or differential. They can be used to monitor atmospheric pressure or used as a trigger switch to activate a system once a pressure limit is reached. Several other driving factors in the selection of components include the operational and storage temperatures, maintenance of any moving parts or routine oiling, and analog or digital signal outputs. Operational properties for the pressure sensor database are compiled from several different suppliers and incorporate a spectrum of different sensor technologies.

Sensors are chosen to represent a wide variety of configurations, instead of all calibrations of a single sensor range to generate a broad database. Data arrays are generated for each unit system and include a comment array containing additional information on use, limitations, maintenance, etc. for each sensor. Several pressure sensors have multi-function capability with built in temperature sensors used in special applications. Models with this capability include the thermocouple temperature limits in the database. Operational performance range for the optimal pressure sensor is based on the known planetary surface pressure defined within the planetary database program.
Additional information on the planetary environment is used to reduce the number of sensors available in the database to those meeting the environmental specifications. Recorded data properties for each type of pressure sensor are shown in Table 17.

<table>
<thead>
<tr>
<th>Table 17 Pressure Sensor Data recorded in ISSPO Tool.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pressure Sensor Database Properties</strong></td>
</tr>
<tr>
<td>Sensor Type</td>
</tr>
<tr>
<td>Stability</td>
</tr>
<tr>
<td>Burst Pressure Rating</td>
</tr>
<tr>
<td>Number of Life Cycles</td>
</tr>
<tr>
<td>Shock Limits</td>
</tr>
<tr>
<td>Operating Frequency Range</td>
</tr>
<tr>
<td>Operating Temperature</td>
</tr>
<tr>
<td>Voltage Requirements</td>
</tr>
<tr>
<td>Sensor Material Construction</td>
</tr>
<tr>
<td>Weight</td>
</tr>
</tbody>
</table>

11. Radiation Sensors

The RADIATION_SENSORS program loads and selects the correct radiation sensor into the main ISSPO program. This module is called when information on the planets radiation exposure is desired. Radiation effects pose several significant hazards to science missions; first and foremost, it’s deleterious effect on biological organisms, and second, its effects on science hardware. Large portions of the sun’s incident radiations is absorbed and blocked by the earth’s magnetic field and ozone layer. This effect protects us from the sun’s output of radiation. On other planets without a planetary magnetic field the surface environment is subject to the full effect of the sun’s radiation and any incident cosmic radiation. Certain types of radiation can be effectively blocked.
by various mediums. Mapping of a planet's radiation exposure will help determine methods to safely and effectively block harmful radiation, and allow for safe human exploration. Radiation has also negative affect flight hardware. Stray high energy particles can interfere with electronic systems on spacecraft, and without control to handle these cases, can render the equipment unusable. During the Gravity Probe B science experiment to prove Einstein's theory of curved space-time, [32] radiation events caused by high energy photons from solar coronal ejections interfered with the delicate electronics onboard the satellite. This resulted in computer "safemode" holds to the mission timeline, to clear out any errors in the flight computer. Safely handling radiation events, and understanding the occurrence of radiation events can allow for expanded science opportunities.

Radiation sensors [33, 34, 35, 36, 37, 38, 39, 40] operate over several different ranges and feature the ability to track different event types. Detectors can track and record single high energy protons and electrons, as well as larger alpha and beta particle radiation forms. Sensors also detect high energy photons existing as gamma and x-ray radiation which is lethal to humans in high doses, or over prolonged periods of exposure. A preliminary knowledge of the expected forms of radiation that will be encountered is required to determine the correct type of detector required for the mission. Radiation sensors can be selected by specific types or general detectors can be used to measure the energy levels of the different particles and the type of particle can be determined later. Selection of the radiation sensor option within the sensor data variable array requires additional program inputs. The different radiation types are entered into the second
column location of the radiation sensor row. Allowable types include “Charged Particle,” “Alpha,” “Beta,” “Gamma,” and “X-Ray.” Multiple types of radiation can be entered into the sensor array with each additional radiation type being entered into the next column location. A sample input file format for radiation sensors in shown in Table 18. Recorded data properties in the radiation sensor program for each type of sensor are shown in Table 19.

<table>
<thead>
<tr>
<th>Table 18 Radiation Sensor Input File Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>1&lt;sup&gt;st&lt;/sup&gt; Column</td>
</tr>
<tr>
<td>“RADIATION”</td>
</tr>
<tr>
<td>“Alpha”</td>
</tr>
<tr>
<td>“Gamma”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 19 Radiation Sensor Data recorded in ISSPO Tool.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radiation Sensor Database Properties</td>
</tr>
<tr>
<td>Sensor Type</td>
</tr>
<tr>
<td>Proton Energy Range</td>
</tr>
<tr>
<td>Electron Energy Range</td>
</tr>
<tr>
<td>Alpha Particle Energy Range</td>
</tr>
<tr>
<td>Beta Particle Energy Range</td>
</tr>
<tr>
<td>Gamma Particle Energy Range</td>
</tr>
<tr>
<td>X-Ray Energy Range</td>
</tr>
<tr>
<td>Particle Count Rate</td>
</tr>
<tr>
<td>Weight</td>
</tr>
<tr>
<td>Power Requirements</td>
</tr>
</tbody>
</table>
12. **Refraction Sensors**

The REFRACTTION_SENSORS program loads and selects the correct refraction sensor into the main ISSPO program. This module is called when information on the nature of a planetary body’s sample refractive index is being determined. Refractometers measure the optical refraction properties of a sample and can determine an unknown samples composition, the purity of a sample, or the concentration of different substances. Samples can be either liquids or gasses. Selection of the sensor will depend on the nature of the data to be returned and known information about the location where the sensor will be operating.

Refractometers operate through the use of a light source and a prism. Prisms can be made of a number of optically pure materials. Photodetectors within the sensor record a “sensed” light level. Given the known refractive properties of the prism the composition of the original sample material can be determined, from the observed optical properties. The refractive index measured, is a measure of angle of light deflection as it passes through the prism, and the detectors register the transition between the dark / light line.

Selection of refractometer sensors is based on the environmental conditions the sensor must be able to survive during the operation mission phase. Database [41, 42] elements consist of industrial sensors and flight models representing commercially available optical designs used to determine concentration or purity, of gasses, liquids, or trans-lucent solids. Database properties tracked in the sensor module are shown in Table 20.
Table 20 Refraction Sensor Data recorded in ISSPO Tool.

<table>
<thead>
<tr>
<th>Refraction Sensor Database Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor Type</td>
</tr>
<tr>
<td>Resolution</td>
</tr>
<tr>
<td>Accuracy</td>
</tr>
<tr>
<td>Voltage</td>
</tr>
<tr>
<td>Power</td>
</tr>
<tr>
<td>Operating Temperatures</td>
</tr>
</tbody>
</table>

13. **Temperature Sensors**

The TEMP_SENSORS program loads and selects the correct temperature sensor needed to survive the environment. This module is called when the atmospheric data suite option is chosen from the sensor data array. Inputs to this program are the chosen unit system, the planetary body's surface temperature, and the detection type. Temperature sensors operate primarily over two different methods, either a voltage change or a resistance measurement over the operational temperature range. The operational sensor type is fed to the TEMP_SENSORS program. Inputs for the sensor type are "VOLTAGE" or "RESISTANCE". The ISSPO program defaults to use "VOLTAGE" if the temperature term and a sensor type are not found in the SENSOR_DATA variable in the same row as the atmosphere option. A sample input for the temperature sensor option is shown in Table 21.
Standardized sets of temperature sensors exist for most applications. Thermal performance values for different sensor types are consistent amongst different vendors, with minor variations in values at the extreme thermal limits. Performance specifications for thermocouples are regulated by several national standard organizations and, while small differences are observed in data between suppliers, the performance of each type of thermal sensor is assumed to be consistent. The temperature sensor database for voltage based thermocouples, consists of currently available calibrations from Omega sensors [43], and contains a wide variety of data on each thermocouple type that can be used to determine the appropriate temperature sensor. Combinations of Thermocouple material and Insulation material must be chosen to survive the planet’s surface temperature and the effects of the atmosphere. Selection of thermal sensors is based on the operational range of each type and the chosen planetary body’s known surface temperature. Additional information on the sensor entered in the comments section provides additional selection criteria for sensor based on popular calibration ranges or calibration for cryogenic temperature ranges. Insulation material for the thermocouple wires is given a rating for its resistance to different environmental factors. An aggregate score is determined for each material and used to determine the appropriate insulation material.
Resistance type thermocouple sensors [44, 45, 46] track the same thermocouple properties, however, they are bare wire elements and do not use insulating material on the sensor leads. Sensor properties recorded in the database are shown in Table 22.

<table>
<thead>
<tr>
<th>Temperature Sensor Database Properties</th>
<th>Thermocouple Properties</th>
<th>Insulator Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor Type</td>
<td>Cold Temperature Limit</td>
<td>Insulator Name</td>
</tr>
<tr>
<td>Hot Temperature Limit</td>
<td>Error Tolerance Percent (%)</td>
<td>Hot Limit</td>
</tr>
<tr>
<td>Error Tolerance Temperature (deg)</td>
<td>Positive Lead Material</td>
<td>Flexibility</td>
</tr>
<tr>
<td>Negative Lead Material</td>
<td>Minimum EMF Voltage</td>
<td>Resistance to Solvents</td>
</tr>
<tr>
<td>Maximum EMF Voltage</td>
<td>Usage/Restriction Comments</td>
<td>Resistance to Bases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resistance to Humidity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Material Construction: Overall</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Material Construction: Conductors</td>
</tr>
</tbody>
</table>

14. **Wind Velocity**

The WIND_VELOCITY_SENSORS program loads and selects the correct wind velocity sensor needed to record air speed in the object’s environment. This module is called when the atmospheric data suite option is chosen from the sensor data array. Understanding of the atmospheric winds adds a key element to the understanding of the nature of the environment on planetary bodies. Atmospheric wind speeds are a key parameter in landing and launching probes to other planets. To ensure safe launch
conditions, limits are placed on wind speeds to ensure that a launch vehicle can safely ascend through the atmosphere without being pushed off course by high winds. Wind speed measurements also add key data to understanding seasonal variations on other planets. Selection of wind velocity measurement is based on the mission operation.

Sensors within the ISSPO database will function for two different operating regimes. For planetary surface measurements for a lander or rover based mission, an anemometer type measurement may present a viable option. For atmospheric wind profile mapping during an atmospheric descent, or any fast moving observational platform, a Doppler based system would present a better solution. However, this also places a mission requirement that some sort of orbital platform is available. Doppler based systems communicate between a sensor located on the entry vehicle and one attached to an orbiting satellite. A series of frequency pulses between the two components can be used to determine the speed of winds between the two elements as the signals will be Doppler shifted slightly by the wind movement. The operating type of the wind velocity sensor is entered into the SENSOR_DATA variable in the input file. Valid sensor configurations within the ISSPO tool are “ANEMOMETER” or “DOPPLER.” If the wind velocity term is not defined within any columns in the atmospheric sensor option in the input file, ISSPO will default to “ANEMOMETER.” A sample input file format for wind velocity sensors is shown in Table 23.
Recent mission data from the Mars Exploration Rovers (MER), Spirit and Opportunity, have shown the influence that the atmosphere can have on the success of the mission. Originally slated for 90 days, the two rovers have far surpassed their expected operating time. This was due to mission planners accounting for the deposition of dust on the rovers’ solar panels obscuring the sunlight and reducing the amount of power generated. However, seasonal spring winds [47, 48] cleaned some of the dust off the panels and allowed additional power to be generated. During Earth’s summer of 2007, a global dust storm [49] encompassed a large portion of the Martian sky, effectively blocking the rovers’ ability to collect sunlight and generate electrical power to keep the electronics warm and communicate with Earth. Improved understanding of how these dust storms evolved will allow future missions to better predict the formation of these storms and successfully maintain power in the vehicles.

Separate database arrays [50, 51] for anemometer type sensors are created for each unit system and loaded into the main program via the initial function call. Different models of sensors include different recording capabilities for minimum, maximum, and average values over a sampling time interval. The sensors also offer different output formats via digital or analog signals and offer the ability to transfer data to desktop
computers for analysis. Selection of the wind velocity sensor is based on data from the planetary database for the desired body. The selection procedure is bypassed and zeroed out if the planetary object under analysis is devoid of an atmosphere. Sensor properties recorded in the database are shown in Table 24.

<table>
<thead>
<tr>
<th>Sensor Type</th>
<th>Wind Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>Volumetric Flow Rate Ability</td>
</tr>
<tr>
<td>Probe Temperature Range</td>
<td>Instrument Temperature Range</td>
</tr>
<tr>
<td>Power Requirements</td>
<td>Weight</td>
</tr>
<tr>
<td>Electronics Dimensions</td>
<td>Probe Dimensions</td>
</tr>
</tbody>
</table>

Doppler based velocity sensors operate under a set of drastically different specifications. Properties [52, 53] for these sensors are stored in separate arrays based on the chosen unit system. Sensor properties recorded in the database are shown in Table 25.

<table>
<thead>
<tr>
<th>Sensor Type</th>
<th>Output Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Level</td>
<td>Aging Data</td>
</tr>
<tr>
<td>Spectral Purity – Harmonics/Non-Harmonics</td>
<td>Stability</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Stability</td>
</tr>
<tr>
<td>Warm up Time</td>
<td>System Voltage</td>
</tr>
<tr>
<td>Operating Temperature Range</td>
<td>Storage Temperature Range</td>
</tr>
<tr>
<td>Temperature Sensitivity</td>
<td>Orientation Sensitivity</td>
</tr>
<tr>
<td>Pressure Sensitivity</td>
<td>Power Requirements</td>
</tr>
<tr>
<td>Weight</td>
<td>Dimensions</td>
</tr>
</tbody>
</table>
15. Digital Signal Processing

The DIGITAL_SIGNAL_PROCESSING module is executed when the "DATA PROCESSING" option is entered in the SENSOR_DATA array variable in the case input file. This device does not constitute a sensor system that records any sort of data. Instead its function is to interpret and manipulate telemetry data received from other onboard sensors and process the data to determine the next set of actions to be taken. This component is crucial when a mission requires for example, altimetry data to determine when to execute mission events during a mission timeline. The altimetry data must be processed within a very short period of time to determine when events must be performed, such as collecting air samples at specific altitudes, or determining when to activate other components in the vehicle.

Digital signal processing devices (DSP) [54, 55] are essentially microprocessor computers that are highly specialized to their application. These devices are built with special handling capability for data processing and to handle large amounts of processing requests from in-flight hardware. Selection of a DSP component requires some additional about the nature of the mission and the amount and speed of data it needs to be able to process. Specialized versions of DSP’s are tuned to perform specific calculation types with very fact processing times. The number of components connected to the device is also a crucial selection factor as a single input source may not overload the processor for speed, but a high number of inputs may overload the sensors processing capacity / bandwidth and either bog down the processor to a point where it cannot
process the data quickly enough to meet mission time events, or signals may get crossed between different computational cycles and possible data corruption could result.

With information on the nature and amount of data the processor will be receiving the selection of correct DSP unit can be performed. Selection of the “DATA ANALYSIS” option within the SENSOR_DATA variable in the input file requires two additional program inputs to aid in sizing the correct sensor. The second column positions relates to the desired operational computational speed of the processor. Acceptable input values for this column are “LOW,” “MEDIUM,” and “HIGH.” The third column position requires a numeric value for the number of devices connected to the processor. A sample input for the digital signal processing option is shown in Table 26. Component properties tracked within the ISSPO database are shown in Table 27.

<table>
<thead>
<tr>
<th>1st Column</th>
<th>2nd Column</th>
<th>3rd Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>“DATA ANALYSIS”</td>
<td>“LOW”</td>
<td>Number of devices</td>
</tr>
<tr>
<td></td>
<td>“MEDIUM”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“HIGH”</td>
<td></td>
</tr>
</tbody>
</table>

Table 26 Digital Signal Processing Input File Format

<table>
<thead>
<tr>
<th>Digital Signal Processor Database Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor Type</td>
</tr>
<tr>
<td>CPU Cores</td>
</tr>
<tr>
<td>Program Memory</td>
</tr>
<tr>
<td>Cycle Time</td>
</tr>
<tr>
<td>System Voltage</td>
</tr>
<tr>
<td>Storage Temperature Range</td>
</tr>
<tr>
<td>Weight</td>
</tr>
<tr>
<td>Current</td>
</tr>
<tr>
<td>CPU Clock Speed</td>
</tr>
<tr>
<td>Data Memory</td>
</tr>
<tr>
<td>Computational Units</td>
</tr>
<tr>
<td>External Interrupts</td>
</tr>
<tr>
<td>Operating Temperature Range</td>
</tr>
<tr>
<td>CPU Power Requirement</td>
</tr>
<tr>
<td>Voltage</td>
</tr>
<tr>
<td>Dimensions</td>
</tr>
</tbody>
</table>
G. Error Program

The ERROR_PRG module can be executed from any point within the ISSPO Program. It is designed to handle any program errors, which may occur, in a clean manner. When an error is encountered within the program, the error program is called with an error number. Each error number is unique and corresponds to a single program fault source. This format is used to simplify the cause of the error when executing the program, instead of simply terminating the program. Error codes are stored as integer numbers and correspond to a single action in the error program.

When an error is recorded anywhere in the program, the error program is executed and sent an error code. The error program then prints a message to the MatLAB command screen with the error number that occurred, a description of the error or possible cause, and a possible solution to resolve the error. Depending on the type of error that occurred, the program may simply terminate, or an alternate solution displayed. Program error codes are organized into several different categories, based on the nature of the error that occurred: a program structure error, invalid variable inputs entered into the program, sensor program errors, files not created, etc. Occasionally an unexpected error may be encountered that will trigger a MatLAB internal compiler error. This is usually the result of a syntax error within the code or illegal variable definition, for example, defining the SENSOR_DATA variable as a single value in the first row and defining a sensor in the second row that requires additional information. A detailed list of program errors and possible solutions is included in Appendix B.
H. Mass Properties

The MASS_PROPERTIES subroutine is called as part of the ISSPO program upon completion of all of all the individual sensor cases. Its function is to collect and maintain information on the mass properties and physical dimensions of the final sensor chosen for each sensor type entered in the input file. At the end of each sensor file, a MatLAB "*.mat" binary file is written containing the final sensor data array and unit labels. In the mass properties program the sensor data file is loaded for each sensor in the input file sensor data array, and returns a program errors if the file does not load properly. Within the program, sensor mass and dimensions are extracted from each sensor data array and compiled into a single mass property array. Mass properties and dimensions for the different sensor types are defined within each sensor database file. If the values are unknown a default zero (0) value is stored in the database location. Sensor data is also converted into the chosen unit system base units. If the UNITS variable is set to "SI" this calculates the sensor masses in kilograms (kg) and dimensions in meters (m). If UNITS is equal to "British" sensor masses are set to pounds mass (lbm) and dimensions are set to feet (ft). These units are used to calculate the total sensor package limits and displays the results in units equivalent to the units set for the MASS_LIMIT variable in the user input file. Additionally the sensors dimensions are stored and converted to sensor volumes in the main ISSPO program.
I. Power Properties

The POWER_PROPERTIES subroutine is called as part of the ISSPO program upon completion of the MASS_PROPERTIES program. It functions to collect and maintain information on the power requirements of the final sensor chosen for each sensor type entered in the input file. A MatLAB “.mat” binary file is written containing the final sensor data array and unit labels. In the power properties program, sensor data files are loaded for each sensor in the input file sensor data array. Within the program, sensor power requirements are extracted from the sensor data array. Power consumption data for all different sensor types is not always consistent or available. When available, for each product, power consumption requirements are recorded. When both voltage and current data are available the system power loads are determined from basic relations. In determining power consumption, the largest power requirement for each sensor is used. This represents a worst case design in the amount of power needed to operate the equipment. If neither voltage, nor current data, is available for a sensor the resulting power consumption is set to zero (0) within the POWER_PROPERTIES program. A check within the main program determines if the total power requirements meets the sensor payload power limit entered into the input file. If the total power limit of the system is exceeded, it does not mean that the configuration is invalid, merely that an insufficient power amount is available to operate all the sensors simultaneously. In this case a power load budget must be established to determine when during the mission is data needed from a specific sensor.
The mission design will have to allow for a contingency to operate a single high power sensor at the expense of several low power sensors, not collecting active data continuously. Sensor power data is converted into the correct unit system. In “SI” UNITS the variable is set to calculate power usage in watts (W). If UNITS is equal to “British” sensor power requirements are set to British Thermal Units per hour (BTU’s/hr). These units are used to calculate the total sensor package power limits and displays the results in units’ equivalent to the units set for the POWER_LIMIT variable in the user input file. Total system power consumption levels are returned to the main ISSPO program and compared to the input system to determine if system constraints are met.

IV. Huygens Probe Benchmark Case

As a benchmark case to validate the methodology developed within the ISSPO program, the interplanetary probe mission to Saturn’s Titan moon carried out by the Huygens probe was used. This spacecraft consisted of several complex sensor packages with independent mission objectives. [56, 57, 58] Data on the spacecraft systems and top level packages is obtained from various references to determine which of the hardware components were based on available components, and which were custom designs built to suit a specific science goal for this mission.
Custom designed sensor configurations will always be required when the mission science question is truly unique or the level of fidelity of a commercially developed sensor is not readily available. In this case an available sensor may only need slight modification by the developer to meet the mission objective. The base model design can act as a reference point that can be modified to handle mission requirements that it was not originally designed to meet. In evaluating the component designs for the different sensor packages on the Huygens Lander, there is a combination of commercially available sensors, modified versions, and custom functioning units used.

Each subsystem of the Huygens Planetary Probe was analyzed separately as stand alone design system. Six primary science packages comprised the science payload for the entire mission. The requirements for each package will be described in turn and evaluated using the mass and power limits for each subsystem. A comparison of the results to the published data available will be made. A schematic layout of the six science payloads on the inverted Huygens probe is shown in Figure 5 looking at the bollom of the probe. The Descent Imager Spectral Radiometer (DISR) and Doppler Wind Experiment (DWE) are mounted on the top surface of the spacecraft and not shown in the figure. Despite the large number of instruments contained in the science payload, it represented only a small portion of the total spacecraft mass and volume. The mission profile was for a short lifetime to collect and relay the data thus a number of components has redundant systems or multiple ways to obtain the different data in case a system failed.
A. Aerosol Collector and Pyrolyser (ACP)

The Aerosol Collector and Pyrolyser (ACP) Package is one of the science measurements chartered with mapping the properties of Titan’s Atmosphere. It samples the atmosphere twice during the entry and descent phase of the mission, in the tropopause (160 – 40 km), and in the cloud layer (23 – 17 km) and prepares the samples for analysis by other components on the probe. Samples are passed to the Gas Chromatograph Mass Spectrometer for analysis. The chemical composition of the samples is analyzed looking for specific combinations of aerosols in the atmosphere. A summary of the ACP sensor system is shown in Table 28.
Table 28 Mission Properties of the Aerosol Collector and Pyrolyser Sensor

<table>
<thead>
<tr>
<th>Aerosol Collector and Pyrolyser Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scientific Objectives:</strong></td>
</tr>
<tr>
<td>Chemical composition of photochemical Aerosols – Hydrogen (H), carbon (C), nitrogen (N) and oxygen (O)</td>
</tr>
<tr>
<td>Relative concentrations of the organic condensates inside the lower stratosphere (C2, H2, C2, H, HC3N, HCN)</td>
</tr>
<tr>
<td>Relative concentrations of the organic condensates within the troposphere (mainly CH4, C2H6)</td>
</tr>
<tr>
<td>Non condensable constituents trapped in the collected particles (CO2)</td>
</tr>
<tr>
<td><strong>Instrument Characteristics</strong></td>
</tr>
<tr>
<td>Sampling of the particles (direct impact plus capture by filtration)</td>
</tr>
<tr>
<td>Transfer of the evaporates and pyrolysis products to the GCMS (via the special ACP inlet)</td>
</tr>
<tr>
<td>Analysis (direct MS and GCMS)</td>
</tr>
<tr>
<td>Designed to operate with precise timing</td>
</tr>
<tr>
<td><strong>Payload Properties:</strong></td>
</tr>
<tr>
<td>Mass = 6.18 kg</td>
</tr>
<tr>
<td>Power (typical/peak) = 3 / 85 W</td>
</tr>
<tr>
<td>Energy = 78 Wh</td>
</tr>
<tr>
<td>Data Rate = 128 bits/sec</td>
</tr>
</tbody>
</table>

The ACP on its own does not make any direct samples of the atmosphere. Instead, it prepares samples of the atmosphere for the GCMS. The design of the ACP was a collaboration of multiple research institutes and industry. Use of this type of device is not common in planetary probe missions and no broad database of similar devices exists that are commercially available. Calibration of this package type was excluded from the rest of the sensors built into the model.

B. Descent Imager / Spectral Radiometer (DISR)

The Descent Imager / Spectral Radiometer (DISR) sensor suite onboard the Huygens probe records images as the probe enters the Titan atmosphere. It is responsible
for recording both descent imagery to monitor the surrounding terrain during the descent phase to provide background context, and upward looking data to observe the optical properties of the atmosphere. The Huygens probe spun as it entered Titan’s atmosphere. With the DISR mounted on the side of the probe, the rotation allowed the probe to record smaller strip and area images that were later compiled into a 360 degree image of the terrain under the craft during descent. As the probe descended towards the surface, the imaging pattern of the spacecraft provided continuously updated terrain images. Upon arrival on the surface, the DISR is also responsible for recording images of the planetary surface and visually determine the landing location of the probe. An overview of the DISR instrument properties and objectives is included in Table 29.

Table 29 Mission Properties of the Descent Imager / Spectral Radiometer

<table>
<thead>
<tr>
<th>Descent Imager / Spectral Radiometer Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instrument Overview:</strong></td>
</tr>
<tr>
<td>CCD Detector Array</td>
</tr>
<tr>
<td>Surface Science Lamp</td>
</tr>
<tr>
<td>In-flight Calibration System</td>
</tr>
<tr>
<td>Flexible data collection software</td>
</tr>
<tr>
<td>Upward and downward-looking IR spectrometer</td>
</tr>
<tr>
<td>Upward and downward-looking violet photometer</td>
</tr>
<tr>
<td><strong>Payload Properties:</strong></td>
</tr>
<tr>
<td>Mass = 8.07 kg</td>
</tr>
<tr>
<td>Energy = 42 Wh</td>
</tr>
</tbody>
</table>

Assembling each of these devices as a complete camera package the payload weight budget would have been exceeded. Instead to meet all the optical observation
requirements commercial grade CCD detectors were coupled to multiple camera lenses, each built to meet a specific observation requirement. This configuration allowed for a higher instrument density focusing multiple devices onto a single CCD array. The Side Looking Imager (SLI), Medium Resolution Imager (MRI), and High Resolution Imager (HRI) are connected by fiber optic cable to focus on different sections of the CCD array. This allows it to simultaneously record data from multiple imagers into a single unit. The custom configuration of the lenses does not represent a commercial viable solution for optical data recording. In this case the ISSPO tool was tasked to select a CCD based on the required types and amount of data to be recorded by the probe during the mission timeframe.

The CCD array used on the Huygens Probe is a modified version of a commercially available CCD array rated for aerospace applications developed by Fairchild Imaging. [60] The flight unit carried aboard the Huygens probe utilized a 512 x 256 pixel frame transfer CCD. The flight unit chosen supports optical binning modes in which areas of the CCD array can be designated to separate images. The use of this option allowed for greatly reduced sensor weights instead of carrying a separate camera system for each image type required. An example of the use of software binning and multiple imaging areas on a CCD array is shown in Figure 6.
Evaluating the ISSPO tool for the CCD array used in the DISR sensor package the following input data in Table 30 was used to determine the correct sensor.

Table 30  ISSPO - Huygens DISR Input Data

<table>
<thead>
<tr>
<th>1st Column</th>
<th>2nd Column</th>
<th>3rd Column</th>
<th>&gt;4th Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;OPTICS&quot;</td>
<td>&quot;ARRAY&quot;</td>
<td>&quot;LOW&quot;</td>
<td>&quot;VISUAL&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;UV&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;NIR&quot;</td>
</tr>
</tbody>
</table>

Based on these criteria for a CCD array design, the ISSPO tool determined a similar CCD array configuration to the design flown on the Huygens Probe. A condensed summary of the sensor data is shown in Table 31. Data on the specific CCD array flown on the Huygens probe was not available for a direct comparison of the CCD
design. The selected design has a larger CCD area, and offers the same frame transfer capability of the model flown on Huygens. The model 424 CCD array is a flight proven model developed for NASA's Deep Impact Mission. [61]

<table>
<thead>
<tr>
<th>Table 31 ISSPO System Architecture DISR CCD Array</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sensor Type</strong></td>
</tr>
<tr>
<td><strong>Imaging Spectrum:</strong></td>
</tr>
<tr>
<td><strong>Array Dimensions - Length:</strong></td>
</tr>
<tr>
<td><strong>Width:</strong></td>
</tr>
<tr>
<td><strong>Sensor Pixel Size:</strong></td>
</tr>
<tr>
<td><strong>Imaging Array Size - Length:</strong></td>
</tr>
<tr>
<td><strong>Width:</strong></td>
</tr>
<tr>
<td><strong>Dimensions - Length:</strong></td>
</tr>
<tr>
<td><strong>Width:</strong></td>
</tr>
<tr>
<td><strong>Height:</strong></td>
</tr>
</tbody>
</table>

C. **Doppler Wind Experiment (DWE)**

The Doppler Wind Experiment (DWE) carried aboard the Huygens probe is used to determine wind speeds in the atmosphere as the probe descended towards Titan’s Surface. The DWE consists of two Ultra-Stable Oscillators, with the properties shown in Table 32, one carried aboard the Huygens Probe, while the other remains aboard the Cassini Orbiter. Measured wind velocity is back calculated from Doppler shifts in the transmitted radio signal frequencies between the two receiver transmitters. The experiment had several objectives.
### Table 32 Huygens DWE Sensor Properties

**Doppler Wind Experiment Properties**

**Scientific Objectives:**
- Determine the height profile of Titan zonal wind velocity over the altitude range from 0 - 160 km with an accuracy of ~1 m/s
- Measure Doppler fluctuations to determine the level and spectral index of turbulence and possible wave activity in Titan’s atmosphere
- Measure Doppler and signal level modulation to monitor Probe Descent Dynamics, including its rotation rate and phase, parachute swing and post-impact status

**Physical Properties:**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass (g)</td>
<td>1898</td>
</tr>
<tr>
<td>Dimensions (mm)</td>
<td>170 x 117 x 119</td>
</tr>
<tr>
<td>Radiation shielding (g)</td>
<td>150</td>
</tr>
</tbody>
</table>

**DC power:**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warm-up power (W)</td>
<td>&lt; 18.4</td>
</tr>
<tr>
<td>DC consumption (mA)</td>
<td>&lt; 675</td>
</tr>
<tr>
<td>Energy (Wh)</td>
<td>&lt; 30 min</td>
</tr>
<tr>
<td>System limit (A)</td>
<td>0.7</td>
</tr>
</tbody>
</table>

**Frequency Parameters:**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output frequency (MHz)</td>
<td>10</td>
</tr>
<tr>
<td>Frequency long term drift</td>
<td>1.4 x 10^{-9}</td>
</tr>
<tr>
<td>Allen Variation</td>
<td>3 x 10^{-11}</td>
</tr>
<tr>
<td></td>
<td>6 x 10^{-12}</td>
</tr>
<tr>
<td>τ = 1 s</td>
<td></td>
</tr>
<tr>
<td>τ = 10 s</td>
<td></td>
</tr>
</tbody>
</table>

The input file setup used by the ISSPO program to determine the properties of the sensor to use for the DWE sensor package is shown in Table 33.

### Table 33 ISSPO Huygens DWE Sensor Input File

<table>
<thead>
<tr>
<th>1st Column</th>
<th>2nd Column</th>
<th>3rd Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ATMOSPHERIC&quot;</td>
<td>&quot;WIND VELOCITY&quot;</td>
<td>&quot;DOPPLER&quot;</td>
</tr>
</tbody>
</table>

The use of Doppler based velocity detections [62] involves tracking shifts in radio frequencies between two receivers. To determine the velocity to first order accuracy for a Doppler shift the primary governing equations are:
\[
\Delta f = -\frac{f}{c} \Delta V \tag{2}
\]

\[
\Delta V = (\vec{V}_p - \vec{V}_o) \cdot \vec{R}_{op} \tag{3}
\]

where \( p \) and \( o \) are the probe and orbiter elements, respectively. And \( \vec{R}_{op} \) is a unit vector path from the orbiter to the probe defined as:

\[
\vec{R}_{op} = \frac{\vec{R}_p - \vec{R}_o}{|\vec{R}_p - \vec{R}_o|} \tag{4}
\]

The \( \Delta V \) is selected to be negative at the beginning of the descent phase resulting in a blue shifted (increased) frequency in Equation 2. The resulting velocity equation is then written as a sum of four velocity components as summed and defined below.

\[
\vec{R}_{op} \cdot \vec{V}_p = V_1 + V_2 + V_3 + V_4 \tag{5}
\]

where,

\( V_1 \) defined as zonal wind \( u \) (positive towards East)
\( V_2 \) defined as Titan’s rotation \( \Omega a \cos \lambda \) (co-aligned with \( V_1 \))
\( V_3 \) defined as meridional wind \( v \) (positive toward North)
\( V_4 \) defined as descent velocity \( v_T + v \) vertical wind \( w \) (positive upwards)

Finally the orbiter’s velocity projection onto the radio path direction is:

\[
\vec{R}_{op} \cdot \vec{V}_o = V_5 \tag{6}
\]

Based on the different sensor types in the database and the planetary environmental properties that the sensor must cope with the ISSPO determined a sensor package with similar properties to the actual flight unit. The DWE package [63] built by Daimler-Benz Aerospace (DASA), now EADS Germany used a commercially developed
space-qualified rubidium oscillator built by Ball Efratom Elektronik GmbH. Detailed information on the specific sensor configuration flown is not available and a similar performing model built by Symmetricom [52] uses a militarized Rubidium Oscillator.

There are slight performance differences between the actual flight unit and the ISSPO design solution. The largest differences is the unit is about half the weight of the Huygens model but consumes more power at max input, but better matches the sensor performance during steady state operation. The model has a significantly shorter warm up period resulting in lower total power consumption. Brief summary results on the DWE sensor package are provided in Table 34 with full details of the sensor design and operation provided in Appendix B.

<table>
<thead>
<tr>
<th>ISSPO DWE Sensor System Architecture Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong>: Symmetricom 8130A</td>
</tr>
<tr>
<td><strong>Physical Properties</strong>:</td>
</tr>
<tr>
<td>Mass (g)</td>
</tr>
<tr>
<td>Dimensions (mm)</td>
</tr>
<tr>
<td><strong>DC power</strong>:</td>
</tr>
<tr>
<td>Warm-up power (W)</td>
</tr>
<tr>
<td>DC consumption (mA)</td>
</tr>
<tr>
<td>Energy (Wh)</td>
</tr>
<tr>
<td><strong>Frequency Parameters</strong>:</td>
</tr>
<tr>
<td>Output frequency (MHz)</td>
</tr>
<tr>
<td>Frequency long term drift</td>
</tr>
<tr>
<td>Allen Variation</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
|                                             | τ = 10 s
D. Gas Chromatograph Mass Spectrometer (GCMS)

Use of Gas Chromatograph Mass Spectrometer (GCMS) in planetary missions is a relatively new option. Typically these units have been custom built and designed specifically for a single mission. As such, a database of flight rated configurations is limited. Numerous laboratory models exist but are not built with the same considerations that are given to flight systems. In aerospace missions mass, power, and volume are all at a premium whereas laboratory models do not have these constraints. Additionally flight rated models must be completely automated as there is no possibility to intervene should a problem arise. The design of such a devise is also a combination of multiple separate instruments. The configuration flown on the Huygens probe is a custom configuration due to the payload constraints but based on commercially available individual products. The completed flight unit now represents a flight proven configuration with enough available performance data that it can represent a stand alone product to be used on other future missions with minimal changes for other planetary atmospheres. The GCMS onboard Huygens was built by NASA’s Goddard Space Flight Center (GSFC), [64, 65, 66] and can be made available to reproduce for any future missions needing a similar device. The device is built around the two basic components: a gas chromatograph, and mass spectrometer. The layout of these components and associated hardware resulted in one of the most complex instruments GSFC has ever built as shown in Figure 7.
The science objectives for the GCMS are defined relatively broad [67], and include the following goals: to determine the noble gas abundance, isotopic ratios, and identify the high molecular weight trace organic compounds. Sufficient data from these sources provided enough design information to add the sensor configuration to the GCMS database. The flight unit is comprised of commercially available products including five ion sources attached to the mass spectrometer. The different components were machined to fit and assembled together to create the final flight unit.

The final assembled version houses all the sensor components and associated electrical hardware to sample the Titan atmosphere and analyze the data. Information required for the ISSPO program to determine the appropriate sensor is shown in Table 35.
Table 35 ISSPO GCMS Sensor Input File Format

<table>
<thead>
<tr>
<th>1st Column</th>
<th>2nd Column</th>
<th>3rd Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;GAS ANALYSIS&quot;</td>
<td>&quot;MASS-CHARGE&quot;</td>
<td>&quot;MEDIUM&quot;</td>
</tr>
</tbody>
</table>

The data in Table 36 corresponds to the final published data for this component. The sensing range is defined as "MEDIUM" in comparison to other models with larger or smaller scanning ranges. It is not clear, from the reports whether this range is the maximum sensing range of unit or merely the range of chemical compounds expected to be found within the Titan atmosphere. In total 5634 mass spectra recordings were made during the descent phase with an additional 2692 spectra samples collected on the surface with a range of 2-141 amu’s. [68] Designs within the ISSPO tool can be set to a high range, if the chemical contents of the planetary atmosphere are unknown; however with a better understanding of the compounds under investigation the required sensing range can be reduced.

Table 36 ISSPO GCMS Sensor Configuration

<table>
<thead>
<tr>
<th>ISSPO GCMS System Architecture Design Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model: GCMS - Huygens</td>
</tr>
<tr>
<td>Physical Properties:</td>
</tr>
<tr>
<td>Mass (kg)</td>
</tr>
<tr>
<td>Dimensions (mm)</td>
</tr>
<tr>
<td>DC power:</td>
</tr>
<tr>
<td>Typical Power (W)</td>
</tr>
<tr>
<td>Average Power (W)</td>
</tr>
<tr>
<td>Peak Power (W)</td>
</tr>
<tr>
<td>Sensing Parameters:</td>
</tr>
<tr>
<td>Mass - Charge Range</td>
</tr>
<tr>
<td>Number of Ion Sources</td>
</tr>
<tr>
<td>Ion Source Charge</td>
</tr>
<tr>
<td>Field Range</td>
</tr>
</tbody>
</table>
E. **Huygens Atmosphere Structure Instrument (HASI)**

The Huygens Atmosphere Structure Instrument (HASI) is a key multi-sensor component to mapping the Titan atmosphere. In incorporated multiple atmospheric sensor elements into an overall package that would monitor most atmospheric properties during the planetary entry. Sensors within this package have been categorized into the type of environmental data that is returned. These categories include: acceleration, pressure, temperature, and Permittivity, Wave, and Altimetry. Mission science objectives for the HASI sensor suite are to determine the density, temperature, and pressure through the descent profile to the surface, determine the nature of the surface contact, whether solid or liquid, determine atmospheric electrical conductivity, electric fields and atmospheric lightning, and surface topography by monitoring surface dielectric fields. Within the design of this sensor package a certain amount of redundancy is included; in case a single element failed the secondary unit would continue to provide information.

![Figure 8 Schematic Diagram of Huygens HASI Sensor Package](image-url)
A diagram of the different sensor components contained in the HASI package is shown in Figure 8. A summary of the different science data to be returned, and sensor system components, is detailed in Table 37. The sensors in the PWA subsystem or mounted external to the spacecraft, and shielded from the electric field generated within the probe body by all the electronic components. Two deployable boom arms are released prior to atmospheric entry, which contain the mutual impedance and relaxation probe sensors. These sensors consist of a pair of metal wire coils and flat disks mounted on the deployed boom arms shown in Figure 14.

<table>
<thead>
<tr>
<th>Sensor Package</th>
<th>Sensor Type</th>
<th>Measured Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerometers (ACC)</td>
<td>3-Axis Acc</td>
<td>Atmospheric deceleration, Descent Monitoring, Impact Response</td>
</tr>
<tr>
<td>Pressure Profile Instrument (PPI)</td>
<td>Kiel probe, capacitive gauges</td>
<td>Atmospheric Pressure</td>
</tr>
<tr>
<td>Temperature Sensor</td>
<td>2-Dual Element Platinum Thermometers</td>
<td>Atmospheric Temperature</td>
</tr>
<tr>
<td>Permittivity, Wave &amp; Altimetry (PWA)</td>
<td>Mutual impedance</td>
<td>Atmospheric electric conductivity</td>
</tr>
<tr>
<td></td>
<td>AC field measurement</td>
<td>Wave electric fields &amp; Lightning</td>
</tr>
<tr>
<td></td>
<td>Relaxation probe</td>
<td>Ion conductivity and DC electric field</td>
</tr>
<tr>
<td></td>
<td>Acoustic sensor</td>
<td>Acoustic noise due to turbulence of storms</td>
</tr>
<tr>
<td></td>
<td>Radar signal processing (FFT)</td>
<td>Radar echoes below 60 km altitude</td>
</tr>
</tbody>
</table>

Performance data on the sensitivity of the PWA sensor components is available in several documents [69,70] however, no data on the physical specification (dimensions,
weight, materials, etc.) of the sensors is available. The scope of the ISSPO tool is
determine the necessary sensor components based on mission requirements from
databases of existing commercially available components. The mutual impedance
sensors and relaxation probe design used do not represent a broad enough spectrum of
uses to develop a database of configuration for interplanetary missions. These
components are excluded from the rest of the sensor design analysis.

Information on sensors in this package based on commercially available sensors
was incorporated into the sensor databases and analyzed to determine the selection
reasoning for each component in this sensor package. For each of the separate science
requirements the appropriate sensor configuration is selected by the ISSPO tool out of the
databases based on the information required by the program for each different sensor
type. A comparison of the ISSPO tool results to the known flight units for each
component of the HASI sensor package is shown in Table 38. Variations in the data
between the flight version and the ISSPO determined sensor result based on the required
input data, is discussed in each separate section.

<table>
<thead>
<tr>
<th>HASI Sensor Package Components</th>
<th>Flight Unit</th>
<th>ISSPO Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sensor Package</strong></td>
<td><strong>Flight Unit</strong></td>
<td><strong>ISSPO Results</strong></td>
</tr>
<tr>
<td>Accelerometer – 3 axis servo</td>
<td>Sundstrand QA-2000-030</td>
<td>QA2000-030</td>
</tr>
<tr>
<td>resistive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pressure Profile Instrument</td>
<td>Vaisala - Barocap</td>
<td>Series 48-0025</td>
</tr>
<tr>
<td>Temperature Sensor</td>
<td>dual element platinum resistance thermometers Rosemount Aerospace Inc.</td>
<td>Goodrich Model 0146MD</td>
</tr>
<tr>
<td>PWA – Acoustic Sensor</td>
<td>Kulite CT-190M</td>
<td>Kulite CT-190M</td>
</tr>
<tr>
<td>PWA – Digital Signal Processor(FFT)</td>
<td>Analog Devices ADSP-2100A</td>
<td>ADSP-2100A</td>
</tr>
</tbody>
</table>
1. **Accelerometers**

Two different configurations of accelerometers are carried aboard the HASI sensor package. The selected sensors feature single axis calibration, one featuring a high sensitivity used along the probe primary x-axis to precisely measure the deceleration profile of the probe through the entry phase, the other configuration three single-axis sensors to measure acceleration loads along all three primary axis at reduced resolution. Sundstrand, makers of the three-axis servo accelerometer [71], was acquired by Allied Signal in 1994, and operated for 5 years before being acquired by Honeywell sensors in 1999 [72]. Since, the launch of the Huygens probe additional versions of the sensors have been developed based on the same sensor platform and sensor performance data varies in sources. Due to the nature of the designs of these elements the maximum operational range of these components is shown on datasheets, however, through signal conditioning the maximum range of the models can be reduced with a corresponding increase in sensor resolution. For the piezoresistive accelerometer the design comes from the Endevco 7264 accelerometer line [73], the optional “T” configuration of the design reduces the transverse sensitivity to one percent from the default three percent calibration. The resulting HASI accelerometer sensor design data is shown in Table 39. Accelerometer data was also used to determine the atmospheric density during the planetary descent from Equation 7:

\[
\rho = \frac{-2ma}{C_D AV^2}
\]  

(7)
where $a$ is the magnitude of the acceleration, $m$ is the entry probe mass, $CD$ is the vehicle's drag coefficient, $A$ is the frontal cross sectional area, and $V$ is the velocity of the probe relative to the atmosphere direction.

2. **Pressure Profile Instrument**

The Pressure Profile Instrument’s objective is to map Titan’s atmosphere pressure profile during the descent phase of the mission. Available data [74] on the configuration of the PPI sensor indicates that the silicon capacitive absolute pressure sensor flown is a variant of the Barocap design manufactured by Vaisala Co., in Helsinki Finland. Two types of silicon diaphragm were used, constructed based on different thicknesses to yield different sensitivities to the pressures in different regions of the atmosphere. Searches for specific information on the configuration flown aboard Huygens yielded no successful results. Available configurations on the Vaisala website, do not represent the flight rated model, or cover the same operational range. Without the specific flight configuration
data to incorporate into the database, the planetary environmental data is used to
determine an alternate sensor with similar performance capabilities.

Contrary data [72] on the performance of the pressure sensing range proposes an
upper limit on the sensing range of 1600 mbar. Onboard the spacecraft the pressure
profile instrument was not exposed directly to the atmosphere, but atmospheric samples
were funneled through a Kiel Probe into the instrument. The performance data for the
flight model is compared to the ISSPO sensor configuration in Table 40.

Table 40 ISSPO HASI Pressure Profile Instrument Sensor Configuration

<table>
<thead>
<tr>
<th>ISSPO HASI PPI Sensor Package Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model :</td>
</tr>
<tr>
<td>Barocap – Flight Unit</td>
</tr>
<tr>
<td>Physical Properties:</td>
</tr>
<tr>
<td>Dimensions (mm)</td>
</tr>
<tr>
<td>Sensing Parameters:</td>
</tr>
<tr>
<td>Pressure Range (g’s)</td>
</tr>
<tr>
<td>Accuracy</td>
</tr>
</tbody>
</table>

3. Temperature Sensors

The flight unit temperature sensors carried as part of the HASI sensor package
features a coarse and fine sensor design. The design features a platinum resistance
thermometer mounted onto a platinum-rhodium truss frame that is mounted on a small
STUB boom that extends past the spacecraft boundary layer, shown in Figure 15, during
entry to measure free stream temperatures. The original flight units were produced by
Rosemount Aerospace, Inc. based in Minnesota, U.S.A. Since the qualification effort and
launch of the Cassini-Huygens mission, they were acquired by Goodrich Sensor Systems [75], and data on the specific model is no longer available.

Based on the design of the TEM structure the measured temperature is not the actual ambient temperature and must be corrected for dynamic effects while moving in the atmosphere. The relationship between measured and corrected temperature [76] is given by Equation 8.

\[ T_{corr} = T_{meas} - r \left( \frac{V_r^2}{2C_p} \right) \]  

(8)

Where “r” is a recovery factor, \( V_r \) is the relative velocity of the sensor to the atmosphere, and \( C_p \) is the specific heat ratio at constant pressure. A number of similar capable platinum resistance thermocouples are available within the known performance specifications of the original model. Information provide to the ISSPO tool to aid in selecting a similar sensor type is shown in Table 41. The performance properties of the selected sensor are shown in Table 42.

<table>
<thead>
<tr>
<th>Table 41 ISSPO HASI Temperature Sensor Input File Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>1(^{st}) Column</td>
</tr>
<tr>
<td>“ATMOSPHERIC”</td>
</tr>
</tbody>
</table>
Table 42 ISSPO HASI Temperature Sensor Configuration

<table>
<thead>
<tr>
<th>ISSPO HASI Temperature Sensor Package Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model : M-0146MD</td>
</tr>
<tr>
<td><strong>Physical Properties:</strong></td>
</tr>
<tr>
<td>Mass (g)</td>
</tr>
<tr>
<td>Dimensions (mm)</td>
</tr>
<tr>
<td><strong>Sensing Parameters:</strong></td>
</tr>
<tr>
<td>Temperature Range</td>
</tr>
<tr>
<td>Stability</td>
</tr>
<tr>
<td>Repeatability</td>
</tr>
</tbody>
</table>

4. *PWA Acoustic Sensor*

The Acoustic sensor flight unit carried aboard the Huygens probe HASI experiment package [77] is tasked to measure the acoustic noise due to the turbulence of storms. However the environmental conditions on Titan placed additional constraints on the selection of the sensor. Due to the cold thermal environment, the reasoning [78] was made to use a pressure sensor capable of surviving the harsh environment, and sacrificing the ability to detect low level noise environments. This reduced ability to detect sounds prevents the pressure sensor from monitoring quiet sounds but allowed it sufficient resolution to detect strong winds and thunder as specified in the mission requirement. Information provided to the ISSPO tool to select this sensor is shown in Table 43. A summary of the relevant performance parameters is listed in Table 44.
Table 43 ISSPO HASI Acoustic Sensor Input File Format

<table>
<thead>
<tr>
<th>1st Column</th>
<th>2nd Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ACOUSTICS&quot;</td>
<td>&quot;SENSOR&quot;</td>
</tr>
</tbody>
</table>

Table 44 ISSPO HASI Acoustic Sensor Configuration

<table>
<thead>
<tr>
<th>ISSPO HASI Acoustic Sensor Package Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model:</strong> Kulite CT-190M</td>
</tr>
<tr>
<td><strong>Physical Properties:</strong></td>
</tr>
<tr>
<td>Mass (g)</td>
</tr>
<tr>
<td>Dimensions (mm)</td>
</tr>
<tr>
<td><strong>Sensing Parameters:</strong></td>
</tr>
<tr>
<td>Acoustic Range</td>
</tr>
<tr>
<td>Resonance</td>
</tr>
<tr>
<td>Temperature Range</td>
</tr>
</tbody>
</table>

5. **PWA Digital Signal Processing**

Data collected from the various sensors in the PWA package that required real time processing to trigger events, or interpret the data was processed via a Digital Signal Processor. In essence these components are micro-computers, with fixed amounts of program and data memory. In selecting the right component for any use, several key operational parameters must be taken into account – including processor speed, memory, number of signal inputs, and special functionality. In selecting a processor speed, faster is usually better, however several limitations are placed on this. The device chosen must be fast enough to read in the sampled data at a fast enough rate, and be able to generate the results quickly enough to be useful. For time critical events, the data must be sampled, analyzed, and results output before too much time has elapsed or else a critical
time event may be missed. The counterpoint to this is to opt for the fastest processor speed, as this would provide more than sufficient computational power. This trade-off comes with a price, as the faster processor is usually heavier, larger, and requires more power to operate. Additionally the processor might be so over-powered that the processor may never use its full capacity and end up wasting precious spacecraft resources.

An additional critical selection criteria is the amount of available memory for the both the program and for data. In development aspects the amount of program code should be kept to a minimum to ensure all the data will be able fit onto a single processor’s memory, as difficulties are often encountered when program data has to be stored in separate memory locations. Attempts to reduce the amount of code used and optimize the organizational pattern can improve the performance of the overall component. Additional care, in the processing of data is also crucial to ensure the program efficiently uses the available memory to prevent possible data corruption if the amount of available data memory is overflowed.

An additional crucial selection consideration in determining the correct DSP is the type of tasks the processor is required to complete. Components are selected based on the specific tasks the processor is required to perform. Special versions of processors are available, tailored with specific computational processing abilities, or algorithms to be able to perform specific types of calculations.

Selection of the DSP unit flown on the Huygens Probe is based on the required computation speed of the processor and the attached components. Sensor data from the
mutual impedance, relaxation probes, acoustic sensor, and radar altimeter unit is passed to the DSP unit and undergoes Fast Fourier Transform analysis before transmitting the data to the Huygens orbiter for the return trip to Earth. Data provided to the ISSPO tool to select the DSP component is shown in Table 45. A summary of properties of the ISSPO resulting selection is in Table 46.

<table>
<thead>
<tr>
<th>Table 45 ISSPO DSP Component Input File Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st Column</td>
</tr>
<tr>
<td>&quot;DATA ANALYSIS&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 46 ISSPO HASI DSP Component Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ISSPO HASI DSP Sensor Package Results</strong></td>
</tr>
<tr>
<td><strong>Model</strong>: Analog Devices ADSP-2100</td>
</tr>
<tr>
<td><strong>Physical Properties:</strong></td>
</tr>
<tr>
<td>Output Power (mW)</td>
</tr>
<tr>
<td>Dimensions (mm)</td>
</tr>
<tr>
<td><strong>Operating Parameters:</strong></td>
</tr>
<tr>
<td>CPU Operating Speed</td>
</tr>
<tr>
<td>Data Memory</td>
</tr>
<tr>
<td>Program Memory</td>
</tr>
<tr>
<td>Temperature Range</td>
</tr>
</tbody>
</table>

F. **Surface Science Package (SSP)**

The Surface Science Package (SSP) carried aboard the Huygens Titan probe is comprised of nine different sensors monitoring seven different types of planetary science data. The charge of the SSP is to measure and determine the nature of Titan’s surface
properties. Most of the sensors were mounted on the external surface of the probe to allow for direct measure of the surface environment. These sensors were mounted on a cylindrical unit dubbed “Top Hat” [79] that extended slightly forward of the surface to provide direct contact with the surface. A summary of the SSP’s scientific objectives is shown in Table 47. The main components of the SSP are located along the probes x-axis at the center of the front of the vehicle. The remaining sensors are mounted internal to the spacecraft. A vent tube that extends through the probe’s structure is attached to the Top Hat to allow Titan’s fluid to the internal sensors if Huygens lands in an ocean environment.

<table>
<thead>
<tr>
<th>Surface Science Package (SSP) Scientific Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine the physical nature and condition of Titan’s surface at the landing site</td>
</tr>
<tr>
<td>Determine the abundances of the major constituents, placing bounds on atmospheric and ocean evolution</td>
</tr>
<tr>
<td>Measure the thermal, optical, acoustic and electrical properties and density of any ocean, providing data to validate physical and chemical models</td>
</tr>
<tr>
<td>Determine wave properties and ocean/atmosphere interaction</td>
</tr>
<tr>
<td>Provide ground truth for interpreting the large-scale Orbiter Radar Mapper and other experimental data</td>
</tr>
</tbody>
</table>

A schematic layout of the sensor contained within the “Top Hat” is shown in Figure 9. The two sensor elements not shown in the diagram below are the two tilt sensor components which are mounted internal to the spacecraft housing atop the SSP electronics box.
The location of the SSP within the overall Huygens probe structure is shown in Figure 10 with the impact penetrometer extending out beyond the surface of the probe vehicle. Based on the summary of mission requirements input into the ISSPO program, a summary sensor design package is detailed in Table 48. The inputs and summary comparison for each sensor design is discussed in the following sections. Information on commercial components used in the SSP is scarce and several of the sensors are approximated in the final ISSPO design solution by similar components.
Several of the components utilized in the SSP represented custom design solutions for the specific application they were tasked for on the Huygens probe.
1. **Accelerometers ACC**

To determine the impact acceleration loads on the vehicle during the final phase of the atmospheric descent, two different types of units are used. One unit (ACC-I) will measure the landing acceleration loads dependent on the surface material composition. The other probe is based on a penetrometer design and extends forward of the outer skin of the vehicle and is designed to measure the surface resistance force on the penetrometer tip as the probe lands on the surface. The ACC-I unit flown is an Endevco 2271AM20 piezoelectric accelerometer [80]. The unit is mounted internal to the spacecraft at the base of the SSP electronics box and records the acceleration profile along the spacecraft’s primary x-axis. Properties of the resulting ISSPO sensor configuration are shown in Table 50.

<table>
<thead>
<tr>
<th>1st Column</th>
<th>2nd Column</th>
<th>3rd Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>“ACCELERATION”</td>
<td>“IMPACT”</td>
<td>“Self Generating”</td>
</tr>
</tbody>
</table>

### Table 49 ISSPO SSP Accelerometer Sensor Input File Format

<table>
<thead>
<tr>
<th>1st Column</th>
<th>2nd Column</th>
<th>3rd Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>“ACCELERATION”</td>
<td>“IMPACT”</td>
<td>“Self Generating”</td>
</tr>
</tbody>
</table>

### Table 50 ISSPO SSP Accelerometer Configuration

**ISSPO SSP Accelerometer Sensor Package Results**

<table>
<thead>
<tr>
<th>Model</th>
<th>Endevco 2271AM20</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Physical Properties:</strong></td>
<td></td>
</tr>
<tr>
<td>Dimensions (mm)</td>
<td>29.2 x 15.88 x 15.88 L x W x H</td>
</tr>
<tr>
<td><strong>Operating Parameters:</strong></td>
<td></td>
</tr>
<tr>
<td>Sensing Range</td>
<td>0 – 10,000 g’s</td>
</tr>
<tr>
<td>Transverse Sensitivity</td>
<td>3 %</td>
</tr>
<tr>
<td>Temperature Range</td>
<td>-269.0 – 260.0 deg C</td>
</tr>
</tbody>
</table>
2. **Tilt Sensor TIL**

The tilt sensor [81] carried aboard the Huygens probe primary purpose was to return orientation data about the spacecraft if it were to land in an ocean. Combined with data from the onboard accelerometer, the position and movement of the probe could be obtained. If the probe lands on solid ground, it will return the final resting orientation. The unit was also designed to operate during the entry phase, and provide orientation data that would improve the fidelity of data collected in the DWE by adding probe orientation data to determine atmospheric wind speeds.

The sensor component flown aboard Huygens consists of two single axis tilt sensors mounted in a dual axis configuration on a small mounting block [82]. Sensors used on the SSP were chosen for their quality, performance and high reliability. The selection criteria required as input information to the ISSPO tool is shown in Table 51. A summary of the resultant sensor design is shown in Table 52.

<table>
<thead>
<tr>
<th>1st Column</th>
<th>2nd Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>“INCLINATION”</td>
<td>“HIGH”</td>
</tr>
</tbody>
</table>

**Table 51 ISSPO SSP Tilt Sensor Input File Format**

**Table 52 ISSPO SSP Tilt Sensor Configuration**

<table>
<thead>
<tr>
<th>ISSPO SSP Tilt Sensor Package Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong>: Spectron L-211U</td>
</tr>
<tr>
<td><strong>Physical Properties:</strong></td>
</tr>
<tr>
<td>Dimensions (mm)</td>
</tr>
<tr>
<td><strong>Operating Parameters:</strong></td>
</tr>
<tr>
<td>Tilt Range</td>
</tr>
<tr>
<td>Resolution</td>
</tr>
<tr>
<td>Temperature Range</td>
</tr>
</tbody>
</table>
3. Temperature Sensor THP

The Temperature sensor carried aboard the Huygens probe as part of the SSP is a dual operation hot wire thermometer to measure surface thermal conductivity. The sensor configuration consists of a redundant pair of cylinders housing a platinum wire resistor. Since the final landing environment was unknown, and the possibility existed that the probe could land on either solid ground or in an ocean environment, each cylinder is designed for a liquid measurement or gaseous atmospheric sample. Each cylinder contains a platinum wire resistor configured to operate as a four wire resistance thermometer, and operates via a transient hot wire method. [83] A schematic drawing of a pair of the four sensing cylinders with a cut-a-way view of the internal structure is shown in Figure 11. The left-hand thinner cylinder is configured for sampling in a liquied environment while the right-hand wider cylinder is configured with a thinner sensing wire to sample the atmosphere.

![Figure 11 Schematic Diagram of Huygens SSP THP Sensor](image)

The thermal conductivity is back calculated from the temperature change in the wire when a known power is applied to the wire. Calculation of the thermal conductivity of the air / fluid is based on equations 9 – 11.
In Equations 9 - 11, $\Delta T$ is the temperature rise in the wire, $\lambda$ is the thermal conductivity of the medium, $\kappa$ is the thermal diffusivity of the medium, $q$ is the power per unit length of wire, $a$ is the radius of the wire, $t$ is the time interval of the sampling period, and $C$ is a constant with a value of $e^8$, and $g$ is Euler’s constant. The $A$ and $B$ terms in Equation 4, represent the gradient and intercept values, respectively, of the plot of $\Delta T$ vs. $\ln(t)$.

The material construction of this sensor is the same platinum as the thermal sensor carried in the HASI sensor package. The configuration of the components here is custom but the same basic configuration can be converted to suite the purpose of the experiment. The selection of this same type of sensor is achieved by loading the same input criteria into the ISSPO tool as shown in Table 53.

<table>
<thead>
<tr>
<th>1st Column</th>
<th>2nd Column</th>
<th>3rd Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>“ATMOSPHERIC”</td>
<td>“TEMPERATURE”</td>
<td>“RESISTANCE”</td>
</tr>
</tbody>
</table>
4. **Acoustic Sensor API**

The Acoustic Properties Instrument (API) is comprised of two separate instruments. A pair of piezoelectric ceramic transducers [80] comprising the API-V element are mounted on opposite sides of the Top Hot facing each other. The two units transmit and receive ultrasonic pulses [84] over the unobstructed path. A timing circuit is used to measure the time it takes the pulses to travel between the sensors. With the known distance between the sensor elements and the signal delay time the speed of sound could be resolved to 8 cm/s. The calculation of the speed of sound is made via several different forms depending on the nature of the medium that the sensors are interacting with, for solids Equation 12a applies, and Equation 12b applies for liquids and gasses.

\[
  c = \sqrt{\frac{B}{\rho}} \quad \text{(12a)}
\]

\[
  c = \sqrt{\frac{\gamma RT}{M}} \quad \text{(12b)}
\]
In Equation 12a, B represents the Young's modulus of the medium, and Equation 12b is a variation of the Ideal Gas equation, where $\gamma$ is the ratio of specific heats (~ 1 for liquids, 1.1-1.4 for gasses), R is the universal gas constant and M is the molecular weight of the gas sample. The other component to the API sensor is an array of 10 resonant piezoelectric plates and operates similar to a SODAR array (Sound Detection and Ranging). This component uses an acoustic beam transmitted to the surface to detect from the signal scattering the number of particles in the atmosphere. The data results will be compared to the radar altimeter and verify results. The use of an acoustic array in this manner is not widely flown on planetary missions, thus a database of commercially available components featuring this ability is limited and was excluded from direct comparison in this analysis.

The API components were designed and built under research collaborations between academia and industry. Data on the configuration of the flight unit version of the components for the sound velocity sensor was unavailable. However, a wide enough database of acoustic transducers, allow the ISSPO tool to determine a similarly capable option that could measure the speed of sound. Input criterion for the sensor selection by the ISSPO tool is shown in Table 53. The performance properties of the resultant ISSPO design configuration are shown in Table 54.

| Table 55 ISSPO SSP Acoustic Properties Instrument Speed of Sound Input File Format |
|---------------------------------|---------------------------------|
| **1st Column**                 | **2nd Column**                  |
| “ACOUSTICS”                    | “VELOCITY”                      |
Table 56  ISSPO SSP Acoustic Sensor Configuration

<table>
<thead>
<tr>
<th>ISSPO SSP Acoustic Sensor Package Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong>: Physical Acoustics R80 Alpha</td>
</tr>
<tr>
<td><strong>Physical Properties:</strong></td>
</tr>
<tr>
<td>Mass (g)</td>
</tr>
<tr>
<td>Dimensions (mm)</td>
</tr>
<tr>
<td><strong>Operating Parameters:</strong></td>
</tr>
<tr>
<td>Frequency Range</td>
</tr>
<tr>
<td>Temperature Range</td>
</tr>
</tbody>
</table>

5.  *Permittivity PER*

The task of the permittivity sensor flown aboard Huygens is to measure the static permittivity of any fluid samples found on the surface. In part this was to aide in the determination of the mixing ratio of ethane to methane in any surface liquids. From the permittivity of the sample, and data from the density sensor, an estimate of the mean molecular weight of the sample can be obtained from the Clausius-Mossotti function, Equation 13. Where \( \rho \) is the density, \( \alpha \) is the molecular polarisability, \( N_A \) is Avogadro’s number, \( \varepsilon_0 \) is the electrical permittivity of free space and \( \varepsilon_0 \) is the refractive index permittivity from the refractometer.

\[
\alpha = \frac{3\varepsilon_0(\varepsilon_r - 1)}{\rho N_A(\varepsilon_r + 2)} \tag{13}
\]

The Huygens design consists of 22 stacked open plate capacitors. [80, 81] A schematic layout of the sensor design is shown in Figure 12. The design of this type of
The sensor is specifically tied to the mission and would be custom built for each application. The design can be based on any number of mission constraints or tuned specifically to determine a number of fluid or gaseous properties can be determined. These units serve a variety of functions that serve other industries and uses, such that no standard design configurations exist. Capacitance units for this application can be configured via many different parameters and are best suited being designed for each application.

Figure 12 Schematic Diagram of Huygens SSP Permittivity Sensor Design

6. **Density \( DEN \)**

The density sensor flown aboard the Huygens probe employs Archimedes' principle in its operation. A buoyant float is suspended off the Top Hat inner wall by a thin flexible beam [81] and the resulting forces on the beam are measured by strain gauges. Additional data on the specific configuration of the sensor are not available. The unit was configured as a Wheatstone bridge circuit, and the unit was designed to operate in the region of 400 – 700 kg/m\(^3\) with a science mission requirement of a resolution of ±2 kg/m\(^3\) at a density of 550 kg/m\(^3\). The design of the electronic circuit can be based on
any circuit configuration and no data on the cylindrical float unit incorporated into the
design is available. Only a constant voltage source is required and the resulting
deflection of the float alters the Wheatstone bridge circuit resistance, and changes the
output voltage. Knowledge of the beam material and its properties allows the
determination of the density of the fluid that caused the resulting deflection and thus the
circuit voltage change. Density sensor configurations like this are simple common
components, but usually custom configured to the exact system use. Sensor
configurations currently within the ISSPO database are based on larger scale Archimedes
buoyancy float designs for measuring density in pipe flows or other large liquid systems.

7. **Refractive Index REF**

The Refractive Index (REF) sensor carried aboard the Huygens probe measured
the refractive index of the Titan atmosphere determine the abundance of methane and
ethane in any liquid samples on the surface. The flight model is a linear critical angle
refractometer [81], which uses a sapphire prism to aid in determining the refractive index
range. The flight unit also incorporates a detector array which incorporates a self-
scanning linear photodiode array. This detector then measures the light incident upon it
and indicates the transition between light and dark on the array. A schematic
representation of the operation of a refractometer is shown in Figure 13. Two LED’s
transmit light through and external to the prism to measure the refraction index. Light
from the internal source “A” refracts of the prism’s external surface and terminates in the
detector array. Knowing the “r” and “s” distances allows the calculation of the refractive index angle \( \Theta_c \) of the prism. Rearranged the equation [85] for “s” is given as a function of the prism (\( \mu_S \)) and fluid (\( \mu_L \)) refractive indexes, in Equation 14. The external light source “B” passes through the sample, either gaseous or liquid, before entering the prism and landing on the detector array.

\[
s = r \times \left( \frac{\mu_L}{\mu_S} \right) \quad (14)
\]

Figure 13 Schematic Diagram of a Refractometer Prism

The final flight configuration is a combination of commercial components which was designed for laboratory use and custom elements to aid in detection and operation. The component was built as a collaboration between the University of Manchester Institute of Science and Technology (UMIST) and Rutherford Appleton Laboratory. The overall design of the refractometer flight unit includes the refractometer sensor, detector, light sources, and associated electronics to record the data.
The location of the refractometer component is within the center of the Top Hat on the Huygens probe. It is oriented with the sapphire prism face facing downward; to interact with the gaseous atmosphere or liquid surface once it lands. [80]

Data on the original laboratory model was not available to include in the ISSPO database, however enough data is available on the performance and configuration of the flight unit to add it to the database for refractometer sensors and is the resulting design from the ISSPO tool. The selection of the “REFRACTION” sensor option in the ISSPO input file requires no additional modifiers, and the resulting design is chosen by the refraction index range and data based on the planetary bodies environment. The summary of the resulting sensor data is then shown in Table 57. The photodiode array [85, 86] is a commercially available unit mounted to the refractometer to record the data sampled during the mission.

Table 57 ISSPO SSP Refractometer Summary Configuration

<table>
<thead>
<tr>
<th>ISSPO SSP Refractometer Sensor Package Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong>: Huygens - Ref</td>
</tr>
<tr>
<td><strong>Physical Properties:</strong></td>
</tr>
<tr>
<td>Dimensions (cm)</td>
</tr>
<tr>
<td>10.0 x 10.0 x 11.61</td>
</tr>
<tr>
<td>Power (mW)</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td><strong>Operating Parameters:</strong></td>
</tr>
<tr>
<td>Refractive Index Range</td>
</tr>
<tr>
<td>1.25 – 1.45</td>
</tr>
<tr>
<td>Accuracy</td>
</tr>
<tr>
<td>0.001</td>
</tr>
</tbody>
</table>
G. ISSPO Huygens Summary

The results of the ISSPO tool comparison with the Huygens mission requirements shows a high correlation of the ISSPO tool results with the flown sensors. Given the mission requirements for science data and the sensor package power and mass limits, the ISSPO tool was able to assemble a sensor configuration capable of meeting the requirements. Data on sensor configurations is based on product specified data sheets available from commercial suppliers. Variations in reported data amongst different vendors make for difficult product comparisons. Data not available for all sensor configurations amongst suppliers is zeroed out within the sensor databases.

For each sensor package the payload totals were used for ISSPO program inputs. Often power or mass properties of the individual components were not available for the individual components to compare against the total sensor package weight. Total sensor package weight used in the inputs file also includes mounting hardware for the components, electrical wiring components, and miscellaneous elements. Component level mass properties for the different sensor elements are not all available for individual level comparisons.

Power requirements for each specific component are also not available and made direct component level comparison difficult. If data is available for both sensor voltage and current then the sensor power level is derived. Some datasheets provide voltage or current, not both making comparison difficult. Power consumption levels for some components are available for different levels of operation, i.e. minimum, typical, and maximum. When maximum power data was available the data is passed to the power
properties function as the total maximum sensor power must be less than the input power limit to pass the program power limit check.

Custom configured sensors for some of the sensor carried aboard the Huygens probe were not included in the benchmark case, as they represent a single point design or use of the sensor has not been used enough to develop a database of sensor configurations. Additional space missions with similar science objectives are required to develop additional sensor databases.

V. Venus Atmospheric Properties Mission

With the development of the ISSPO tool, a study for an interplanetary mission can be made. Much interest as of late has been focused on Mars and the outer gas giant planets. Often forgotten are the inner planets from which there is still a wealth of information to be determined. If the inner planets, one still poses many interesting questions, that have yet to be answered. Venus offers many additional atmospheric questions, and represents one of the most extreme environments to survive.

Venus's atmosphere [1] represents an environment with a surface pressure about 90 times that of earth, planetary surface temperature of 730 K, and a carbon dioxide (CO$_2$) atmosphere with cloud layers compromised of sulfuric acid. The upper region in the atmosphere also rotates about the planet in a period of about 4 days, about 60 times faster than at the surface. Currently it is unknown what the driving mechanism is behind this observation. Flow around the planet also tends toward the poles, but no observed
flows head away from the polar regions. The nature of the harsh environment on Venus makes long term observation of atmospheric properties difficult, and the underlying operation of some of the atmospheric properties are likely to remain a mystery until new methods can be found to survive the environment.

The objective goals of the mission will be to provide additional information on the structure of Venus's atmosphere, and to try to provide insight into additional mechanisms that could be causing the observed atmospheric properties. Along with the atmospheric suite, additional sensors carried aboard the craft can provide additional contextual information to exactly what is occurring during the mission timeframe. Unfortunately due to the extremely hot and corrosive nature of Venus's atmosphere, a long term mission that would operate over several weeks or months is not possible, so as much information on what is occurring must be recorded in the mission time available. The mission window would be on the order of several hours Earth time, and last during the atmospheric entry and for a period of time on the surface.

The sensor elements required during this planetary mission are shown in Table 58, along with information on the nature of the observation and its benefit to the planned mission concept. With the relative closeness of Venus, the mission timeframe is reduced, and the transmission time delay is significantly shorter than missions to the outer planets.
Table 58  Venus Atmospheric Mission Objectives

<table>
<thead>
<tr>
<th>Mission Data</th>
<th>Objective Reasoning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATMOSPHERE - Temperature</td>
<td>Provide Atmospheric Temperature consistent with mission trajectory, provides temperature calibration data for other sensors onboard</td>
</tr>
<tr>
<td>ATMOSPHERE - Pressure</td>
<td>Provide Atmospheric Pressure consistent with mission trajectory, provides pressure calibration data for other sensors onboard</td>
</tr>
<tr>
<td>ATMOSPHERE – Wind Velocities</td>
<td>Measure Wind Velocity profiles throughout the atmospheric descent, providing data to validate physical models</td>
</tr>
<tr>
<td>INCLINATION</td>
<td>Provides spacecraft orientation information to determine relative wind flows around spacecraft</td>
</tr>
<tr>
<td>ACCELERATION</td>
<td>Provide descent deceleration profile information and use for event sequence timings</td>
</tr>
<tr>
<td>EM FIELD</td>
<td>Map planetary magnetic field to determine if it accounts for observed wind flows</td>
</tr>
<tr>
<td>RADIATION</td>
<td>Detect charged particle occurrences in the atmosphere looking for energetic reactions in the atmosphere</td>
</tr>
<tr>
<td>OPTICS</td>
<td>Provide in-situ visual observations of wind flows in visual, ultraviolet, and infrared wavelengths to track cloud motion patterns</td>
</tr>
</tbody>
</table>

These sensor data types are then loaded into an ISSPO input file to investigate the destination planet Venus, and use the “SI” unit system. In this case the mission is being planned and there are no fixed limits to apply to the sensor design for a preliminary configuration. Values are still required by the program, but can be set to any values. If the resulting package is less than the input values, the mission design will “appear” to “succeed”, and if the input limits the tool will “appear” to “fail”. However, since there is no driving goal to meet for this mission concept, a failure case merely means that the input limits are too low. This can be resolved by finding ways to reduce the weight, volume, or power consumption, through a custom designed configuration, or find ways to
increase the payload budget limits. Either of these two methods usually means additional development time and additional budgetary costs.

<table>
<thead>
<tr>
<th>1st Column</th>
<th>2nd Column</th>
<th>3rd Column</th>
<th>4th Column</th>
<th>5th Column</th>
<th>6th Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ATMOSPHERE&quot;</td>
<td>&quot;WIND VELOCITY&quot;</td>
<td>&quot;ANEMOMETER&quot;</td>
<td>&quot;TEMPERATURE&quot;</td>
<td>&quot;VOLTAGE&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;INCLINATION&quot;</td>
<td>&quot;MEDIUM&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;ACCELERATION&quot;</td>
<td>&quot;MEDIUM&quot;</td>
<td>&quot;Voltage&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;EM FIELD&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;RADIATION&quot;</td>
<td>&quot;Charged Particle&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;OPTICS&quot;</td>
<td>&quot;ARRAY&quot;</td>
<td>&quot;MEDIUM&quot;</td>
<td>&quot;VISUAL&quot;</td>
<td>&quot;UV&quot;</td>
<td>&quot;NIR&quot;</td>
</tr>
</tbody>
</table>

A summary of the resulting design for the different sensor is shown in Table 59. It includes a minimum design for each sensor, and the quality of the data returned would be improved by incorporating all spacecraft axis and some redundancy into the sensor design. Sensors that record one-dimensional data, i.e. wind velocity, inclination, and acceleration would require at least one sensor along each primary spacecraft axis to provide a complete description of the data being recorded. With the primary data interest in wind velocity, use of a redundant design would be beneficial to the returned science data.
### Table 60 ISSPO Venus Atmosphere Mission Sensor Package Summary Results

<table>
<thead>
<tr>
<th>SENSOR</th>
<th>NAME</th>
<th>RANGE</th>
<th>MASS</th>
<th>VOLUME</th>
<th>POWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATMOSPHERE - Temperature</td>
<td>K</td>
<td>-200 - 1250 deg C</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ATMOSPHERE - Pressure</td>
<td>13(C,U)2000P</td>
<td>0 - 13789.5 kPa</td>
<td>0</td>
<td>29.41 cm³</td>
<td>0.030 W</td>
</tr>
<tr>
<td>ATMOSPHERE - Wind Velocities</td>
<td>FT 702</td>
<td>0 - 70 m/s</td>
<td>0.500 kg</td>
<td>0.0004 m³</td>
<td>0</td>
</tr>
<tr>
<td>INCLINATION</td>
<td>L-212T</td>
<td>-45 - 45 deg</td>
<td>0</td>
<td>12.00 cm³</td>
<td>0</td>
</tr>
<tr>
<td>ACCELERATION</td>
<td>MA15</td>
<td>-50 - 50 g's</td>
<td>0.1417 kg</td>
<td>47.72 cm³</td>
<td>0.64 W</td>
</tr>
<tr>
<td>EM FIELD</td>
<td>TAM-1</td>
<td>0 - 1000 mG</td>
<td>1.080 kg</td>
<td>0.0016 m³</td>
<td>0</td>
</tr>
<tr>
<td>RADIATION</td>
<td>LPD</td>
<td>0 - 250 MeV</td>
<td>7.00</td>
<td>0.0067 m³</td>
<td>15 W</td>
</tr>
<tr>
<td>OPTICS</td>
<td>CCD 3041</td>
<td>UV VISUAL NIR</td>
<td>0</td>
<td>3.384 cm³</td>
<td>0</td>
</tr>
<tr>
<td>SUMMARY</td>
<td></td>
<td></td>
<td>8.7217 kg</td>
<td>0.00871 m³</td>
<td>15.77 W</td>
</tr>
</tbody>
</table>

**NOTE:** All values are approximate and subject to change based on mission requirements and data acquisition.
The full details of the assembled sensor package are noted in Appendix C. Data for some of the components is not available, and the results might vary with additional sensor data. With the sensor configuration known it can be incorporated into the spacecraft development process and provide information for the size and power of the vehicle needed to carry the science payload.

VI. Summary

The ISSPO Tool is developed as a preliminary tool to determine the sensors required to answer planetary science mission questions. Given a planetary body and a description of planetary science objectives to be accomplished, the tool will determine the sensors needed to return the results. ISSPO selects the correct sensor components based on the input criteria and the planetary environmental properties. Selection of the final sensor components are then based on additional operational parameters that vary for each different sensor component. Databases for each type of sensor are based on commercially available sensors. The goal of this is to reduce the amount of time spent performing trades studies to determine the correct sensor to use. The use of commercially developed products and flight rated hardware reduces the time and cost of new space missions by having to develop new custom hardware components for each mission.

A benchmark case proving the sensor design algorithms was performed on the Huygens probe mission to Titan. Science objectives for each sensor package were
broken down into the different types of sensor data that was to be provided. The requirements for the different sensors were input into the ISSPO input file along with the sensor package weight and power limits. Evaluating each sensor package resulted in selection of the Huygens hardware for known flight components. For sensors where the commercial component was unknown, similar performing sensors were chosen based on operational range, environmental properties and the other sensor requirements input into the ISSPO program file. The resulting design configurations agree within a reasonable margin to the sensor package mass and power requirements.

A sensor payload configuration was developed for a conceptual mission design to Venus to answer additional question on the mechanisms that drive the intense atmospheric winds. The sensor package was designed to monitor atmospheric winds, probe orientation, acceleration, planetary magnetic fields, charged particle radiation, and optical imaging to provide a visual record of atmospheric cloud movements. At this conceptual level no restrictions on payload mass, volume, or power constraints are applied to the mission design. In this manner the ISSPO tool can provide preliminary budgets, for mission requirements, to determine the minimum sensor configuration needed to obtain the mission science data.

The summary package comprises commercially available design components, and could be modified further by the use of custom designed sensors to provide higher fidelity data or modified to reduce mass or power requirements. The use of custom mission components, adds complexity, time, and cost into the development time frame for
any mission, as the unit must be fully qualified and tested before being certified for flight use.

The ISSPO tool also aids in mission design. Based on the knowledge of the planetary science mission a preliminary payload mass budget can be determined. With this knowledge the associated spacecraft hardware can be determined required to support the mission. The size of the spacecraft will determine the required size of the launch vehicle and the operating capabilities it will need to have to send the probe to its target. This information then can be used to determine the operating timeline for the mission.

VII. Future Work

The framework has been established here for a tool to determine the optimal sensor configuration based on the mission science requirements. Refinements can be made to each available sensor database, update existing database values, add additional performance values, incorporate sensors based on different governing equations, new sensor types, etc. As each new planetary mission is executed, sensor components within the database can be updated, and the selection algorithms refined. Additional refinements can be made to the tool at system level parameters and account for interactions between the sensors and the spacecraft systems.

With each new planetary mission the databases of flight proven equipment will grow, and new methods will be evaluated to obtain science data. Development of custom configurations today will lead to common components in the future. There
always exists a need for new custom sensor components to answer planetary science questions or deliver new levels of data fidelity that were unobtainable before. The science and technology for these new sensor designs can be built into the ISSPO Program and expanded as new methods are found to determine answers to new scientific questions.
REFERENCES


[84] Svedham, H., Lebreton, J-P., Zarnecki, J., Hathi, B., “Using Speed of Sound Measurements to Constrain the Huygens Probe Descent Profile,” *Proceedings from*

APPENDIX A:

N2 Systems Diagram

The N2 systems diagram demonstrates the hierarchical nature of the program's operation and the flow of requirements and parameters between the program modules. It visually illustrates the dependent nature of the design selection based on all the other sensor component elements within the system.

The diagonal flow nature of the chart details the order of operation of the program and illustrates the direction movement of program inputs and outputs between the modules. The goal is begin with the broadest goals for the final system and use the top level programs to determine all the inputs needed by the sub-functions and process them in the most efficient manner. Arrows in the upper right diagonal portion of the diagram indicate the inputs and outputs of the different modules and how the chosen components provide inputs to other modules. Arrows below the main diagonal shape indicate a direct relational dependency on a higher level component by another component lower along the diagonal. The organization of the individual components is to reduce the number of "up-flowing" program elements in the illustration.

By reducing the number of up-flowing elements the design cycle is reduced and fewer iterations are required before a final design is achieved. The input file format required by the ISSPO program, loads all the required program data and determines which sensor to run and provides them in a simple method, reducing the number of design cycle iterations. The full system diagram detailing the relationship between all the program modules, and flow down of data is detailed in Figure A-1. The atmospheric
suite is shown as the single sensor package system as it is called in main program. The individual sensors contained within this module are tightly coupled together, and more dependent on the output of other sensor within this module. The $N^2$ diagram showing the interconnectivity of atmospheric sensors is shown in Figure A-2.
Figure A-1  ISSPO Systems N2 Flowchart
Circular nodes in the two diagrams indicate a connection between data flow between the different sensor modules. These sensors are often designed to survive harsh environments and include the ability to monitor multiple atmospheric parameters. Depending on the environment and desired sensor properties, use of a single sensor may prove to be a superior option than separate sensors (e.g., pressure sensor monitoring temperature).
APPENDIX B

Huygens Probe Benchmark Results

1. Descent Imager / Spectral Radiometer
   The resultant input design files for the Descent Imager / Spectral Radiometer package in the ISSPO comparison of the Huygens Mission is detailed in the subsequent data files.

Input File
function Huygens_DISR

§*** Declare and Load all Input Variables ***
BODY = 'Titan'; % 'SI' or 'British' Units
UNITS = 'SI';
PRINTFLG = 'N'; % 'Y' or 'N'

MASS_LIMIT = 8.1; % Max Allowable Sensor Package Wt 'lbm' or 'kg'
POWER_LIMIT = 70; % Max Total Power Available 'BTU's/hr' or 'W'
VOLUME_LIMIT = 0.3; % Max Allowable Volume 'ft^3' or 'm^3'

% Type of Science Data to be returned
SENSOR_DATA = { 'OPTICS' 'APRAY' 'LOW' 'VISUAL' 'UV' 'NIR' };

Output File:

******************************************************************************
*** IN-SITU SENSOR PAYLOAD OPTIMIZATION TOOL ***
******************************************************************************

*** PROGRAM INPUTS ***
User Input Filename: Huygens_DISR
Planetary Body Name: Titan
Unit System: SI
Print Summary Option: N

Payload Mass Limit: 8.100 kg
Payload Volume Limit: 0.3000 m^3
Payload Power Limit: 70.000 W
SENSOR PACKAGES DESIGNED:

OPTICS
- ARRAY
- LOW
- VISUAL
- UV
- NIR

*** PAY LOAD SENSOR BREAK DOWN ***

<table>
<thead>
<tr>
<th>SENSOR</th>
<th>MASS</th>
<th>VOLUME</th>
<th>POWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTICS</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>SENSOR PAYLOAD TOTAL</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

*** CASE LIMITS SUMMARY ***

CASE MASS LIMIT: SUCCEEDED!
Designed Mission Sensor Package Meets Mass Limit!
Minimum Sensor Requirements met allows for System Redundancy in Components.

CASE VOLUME LIMIT: SUCCEEDED!
Designed Mission Sensor Package Meets Volume Limit!
Minimum Sensor Requirements met allows for System Redundancy in Components.

CASE POWER LIMIT: SUCCEEDED!
Designed Mission Sensor Package Meets Power Limit!
Minimum Sensor Requirements met allows for System Redundancy in Components.

*** PLANETARY DATABASE SUMMARY RESULTS ***

*** INPUTS ***
Planetary Body Name: TITAN
Unit System: SI

*** PLANETARY BULK PROPERTIES ***

Planetary Mass: 1.34550e+023 kg
Planetary Volume (10^10): 7.15188e+009 m³
Planetary Equatorial Radius: 2.57500e+006 m
Planetary Polar Radius: 2.57500e+006 m
Planetary Gravity : 1.35200e+000 m/s^2
Planetary Density : 1.88000e+003 kg/m^3
Planetary Ellipticity : 0.00000e+000
Planetary Escape Velocity : 2.63900e+003 m/s
Planetary GM : 0.00000e+000 km^3/s^2
Planetary Bond Albedo : 2.20000e-001
Planetary Visual Geometric Albedo : 2.20000e-001
Planetary Visual Magnitude : 7.90000e+000
Planetary Solar Irradiance : 1.50440e+001 W/m^2
Planetary Black Body Temperature : 0.00000e+000 deg C
Planetary Magnetic Field Strength: 0.00000e+000 mGauss
Planetary Moment of Inertia : 0.00000e+000
Planetary Satellites : 0

*** PLANETARY ORBITAL PARAMETERS ***
Semi-Major Axis : 1.22183e+006 m
Sidereal Orbit Period : 1.59450e+001 Earth Days
Tropical Orbit Period : 1.59450e+001 Earth Days
Perihelion Distance : 0.00000e+000 m
Aphelion Distance : 0.00000e+000 m
Synodic Period : 1.59450e+001 Earth Days
Mean Orbital Velocity : 0.00000e+000 m/s
Maximum Orbital Velocity : 0.00000e+000 m/s
Minimum Orbital Velocity : 0.00000e+000 m/s
Orbital Inclination : 3.00000e-001 deg
Orbital Eccentricity : 2.92000e-002
Sidereal Rotation Period : 3.82690e+002 Earth Hours
Length of Day : 3.82690e+002 Earth Hours
Obliquity to Orbit : 0.00000e+000 deg

*** ATMOSPHERIC PROPERTIES ***
Planetary Surface Temperature : -179.45 deg C
Planetary Surface Pressure : 1.46700e+005 Pa
Planetary Surface Density : 5.50000e+000 kg/m^3

ATMOSPHERIC COMPOSITION
Major Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>% Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2</td>
<td>98.400 %</td>
</tr>
<tr>
<td>CH4</td>
<td>1.600 %</td>
</tr>
</tbody>
</table>

Trace Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Amount</th>
<th>Concentration</th>
</tr>
</thead>
</table>

*** I M A G I N G S E N S O R S U M M A R Y R E S U L T S ***

*** I N P U T S ***
System Type: ARRAY
Resolution: LOW
Image Type: VISUAL UV NIR
Unit System: SI

*******************************************************************************

** Sensor Properties **
Imaging Sensor Type: CCD 424
Imaging Spectrum: X-RAY UV VISUAL NIR MICRO
Array Dimensions - Length: 1024 # Pixels
Width: 1024 # Pixels
Sensor Pixel Size: 21.0000 micro-m
Imaging Array Size - Length: 21.500 mm
Width: 21.500 mm
Read Noise at 1 MHz: 0 e-
250 kHz: 0 e-
Full Well Capacity - Pixel: 0 ke-
Register: 500 ke-
Sensor Gain: 0.00 e-/ADU
Sensor Linearity: 0.00 %
ADC Dynamic Range: N/A N/A
Readout Rates: 0.0000 MHz
Readout Time: 0.000000 sec
Frame Rates: 0.00 fps

Environmental Properties
Operating Temperature Range - Low: -30.00 deg C
High: 100.00 deg C
Cooled Temperature: 0.00 deg C
Cooling Method: N/A N/A

Physical Properties
Dimensions - Length: 73.15 mm
Width: 52.83 mm
Height: 6.10 mm
Sensor Mass: 0.000 kg

Power Requirements
Input Voltage: 24.000 V
Input Voltage Type: DC Type
Input Amperage: 0.000 mA
Power Requirement: 0.000 mW

Sensor Comments:
Split Frame Transfer CCD Array
2x - 1024 x 530 pixel Storage Areas
Rapid image capture capability
Multiple output image formats
Four-port readout
Low Read Noise
Space Qualified - Deep Impact Mission
Software Gain & Binning
Scientific precision and accuracy
2. Doppler Wind Experiment (DWE)

The resultant input design files for the Doppler Wind Experiment (DWE) package in the ISSPO comparison of the Huygens Mission is detailed in the subsequent data files. In the ISSPO configuration the Wind Sensor option is part of the Atmospheric suite and runs all components. The other options can be disregarded and look at the Wind Velocity component only.

Input File:

function Huygens_DWE

%*** Declare and Load all Input Variables ***
BODY = 'Titan';
UNITS = 'SI';  % 'SI' or 'British' Units
PRINTFLG  = 'N';  % 'Y' or 'N'

MASS_LIMIT = 1.9;  % Max Allowable Sensor Package Wt 'lbm' or 'kg'
POWER_LIMIT = 18;  % Max Total Power Available 'BTU's/hr' or 'W'
VOLUME_LIMIT = .0025;  % Max Allowable Volume 'ft''3' or 'm''3'

% Type of Science Data to be returned
SENSOR_DATA = { 'ATMOSPHERE' 'WIND VELOCITY' 'DOPPLER' };

Output File:

**************************************************
*** IN-SITU SENSOR PAYLOAD OPTIMIZATION TOOL ***
 **************************************************

*** PROGRAM INPUTS ***
User Input Filename: Huygens_DWE
Planetary Body Name: Titan
Unit System: SI
Print Summary Option: N

Payload Mass Limit: 1.900kg
Payload Volume Limit: 0.0025 m'3
Payload Power Limit: 18.000 W

**************************************************

SENSOR PACKAGES DESIGNED:
ATMOSPHERE
- Temperature
- Pressure
- Wind Velocity
- Humidity
- Density

**PAYLOAD SENSOR BREAKDOWN**

<table>
<thead>
<tr>
<th>SENSOR</th>
<th>MASS</th>
<th>VOLUME</th>
<th>POWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATMOSPHERE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Pressure</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0250</td>
</tr>
<tr>
<td>Wind Velocity Probe</td>
<td>1.8000</td>
<td>0.0006</td>
<td>35.0000</td>
</tr>
<tr>
<td>Humidity</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Density</td>
<td>5.0000</td>
<td>0.0070</td>
<td>0.8250</td>
</tr>
<tr>
<td>SENSOR PAYLOAD TOTAL</td>
<td>5.9000</td>
<td>0.0076</td>
<td>35.8500</td>
</tr>
</tbody>
</table>

***CASE LIMITS SUMMARY***

- **CASE MASS LIMIT:** FAILED!
  Design:
  - Designed Mission Sensor Package EXCEEDS Mass Limits
  - Database Configuration for Required Sensors Exceeds Input Sensor Package Mass Limit
  - Reconfigure Sensor Package for Less Components or Select Multiple Data Type Sensors in

- **CASE VOLUME LIMIT:** FAILED!
  Design:
  - Designed Mission Sensor Package EXCEEDS Mass Limits
  - Database Configuration for Required Sensors Exceeds Input Sensor Package Volume Limit
  - Reconfigure Sensor Package for Less Components or Select Multiple Data Type Sensors in

- **CASE POWER LIMIT:** FAILED!
  Planning:
  - Designed Mission Sensor Package EXCEEDS Power Limits
  - Database Configuration for Required Sensors Exceeds Input Sensor Package Power Limit
  - Total Power Limit Achievable by Selecting Sensor Component Duty Cycles with Mission
  - Reconfigure Sensor Package for Less Components or Select Multiple Data Type Sensors in

***PLANETARY DATABASE SUMMARY RESULTS***

***INPUTS***

- Planetary Body Name: TITAN
- Unit System: SI
*** PLANETARY BULK PROPERTIES ***

- Planetary Mass: 1.34550e+023 kg
- Planetary Volume (10^10): 7.15188e+009 m^3
- Planetary Equatorial Radius: 2.57500e+006 m
- Planetary Polar Radius: 2.57500e+006 m
- Planetary Gravity: 1.35200e+000 m/s^2
- Planetary Density: 1.88000e+003 kg/m^3
- Planetary Ellipticity: 0.00000e+000
- Planetary Escape Velocity: 2.63900e+003 m/s
- Planetary GM: 0.00000e+000 km^3/s^2
- Planetary Bond Albedo: 2.20000e-001
- Planetary Visual Geometric Albedo: 2.20000e-001
- Planetary Visual Magnitude: 7.90000e+000
- Planetary Solar Irradiance: 1.50440e+001 W/m^2
- Planetary Black Body Temperature: 2.63900e+003 deg C
- Planetary Magnetic Field Strength: 0.00000e+000 mGauss
- Planetary Moment of Inertia: 7.90000e+000
- Planetary Satellites: 0

*** PLANETARY ORBITAL PARAMETERS ***

- Semi-Major Axis: 1.22183e+006 m
- Sidereal Orbit Period: 1.59450e+001 Earth Days
- Tropical Orbit Period: 1.59450e+001 Earth Days
- Perihelion Distance: 0.00000e+000 m
- Aphelion Distance: 0.00000e+000 m
- Synod Period: 1.59450e+001 Earth Days
- Mean Orbital Velocity: 0.00000e+000 m/s
- Maximum Orbital Velocity: 0.00000e+000 m/s
- Minimum Orbital Velocity: 0.00000e+000 m/s
- Orbital Inclination: 3.00000e+001 deg
- Orbital Eccentricity: 2.92000e-002
- Sidereal Rotation Period: 3.82690e+002 Earth Hours
- Length of Day: 3.82690e+002 Earth Hours
- Obliquity to Orbit: 0.00000e+000 deg

*** ATMOSPHERIC PROPERTIES ***

- Planetary Surface Temperature: -179.45 deg C
- Planetary Surface Pressure: 1.46700e+005 Pa
- Planetary Surface Density: 5.50000e+000 kg/m^3

ATMOSPHERIC COMPOSITION

Major Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>% Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2</td>
<td>99.400 %</td>
</tr>
<tr>
<td>CH4</td>
<td>1.600 %</td>
</tr>
</tbody>
</table>

Trace Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Amount</th>
<th>Concentration</th>
</tr>
</thead>
</table>
** WIND VELOCITY SENSOR SUMMARY RESULTS **

** INPUTS **
Wind Velocity Sensor Type: DOPPLER
Unit System: SI

** Sensor Properties **
Wind Velocity Sensor Type: 8130A
Sensor Output Frequency: 10.000 MHz
Sensor Output Level: 7.000 dBm
Output Tolerance: 1.500 dBm
Spectral Purity - Harmonics: -30.000 dBc
Spectral Purity - Non-Harmonics: -80.000 dBc
Aging Data - Month: 5.00E-011 Hz/month
- 10 years: 1.00E-009 10 yrs
Sensor Accuracy: 1.00E-011 Hz @ 25 deg C
Sensor Stability: 3.00E-012 Hz @ 100 sec
Sensor Warm Up Time: 14.000 min
Sensor Lifetime: 20.0 years

Environmental Properties
Operating Temperature Range - Low: -40.000 deg C
High: 68.000 deg C
Storage Temperature Range - Low: -62.000 deg C
High: 85.000 deg C
Temperature Sensitivity - Op Range: 3.00E-010 Hz
Orientation Sensitivity: 5.00E-011 Hz
Pressure Sensitivity: 1.00E-013 Hz/mbar

Power Requirements
Voltage Type: DC Type
System Voltage Levels: 22.000 32.000 V
Warm Up Time Max Power: 35.000 W
Steady State Power: 22.000 W
Input Power - Quiescent: 12.000 W

Physical Properties
Sensor Mass: 0.900 Kg
Sensor Dimensions - Length: 10.260 cm
Width: 7.410 cm
Height: 7.260 cm

Sensor Comments:
Modern Militarized Design
5 and 10 MHz Sinewave Outputs
RS-232 Digital Control and Monitoring
Ruggedized High Performance Rb Physics Package
Meets many Mil-Spec Standards
Data for Single Unit
Two Units required for Doppler Tracking
Space Qualified Hardware Production capability

3. **Gas Chromatograph Mass Spectrometer (GCMS)**
   The resultant input design files for the Gas Chromatograph Mass Spectrometer package in the ISSPO comparison of the Huygens Mission is detailed in the subsequent data files.

**Input File:**

```matlab
function Huygens_GCMS
%*** Declare and Load all Input Variables ***
BODY = 'Titan';
UNITS = 'SI'; % 'SI' or 'British' Units
PRINTFLG = 'N'; % 'Y' or 'N'
MASS_LIMIT = 17.3; % Max Allowable Sensor Package Wt 'lbm' or 'kg'
POWER_LIMIT = 79.0; % Max Total Power Available 'BTU's/hr' or 'W'
VOLUME_LIMIT = .0185; % Max Allowable Volume 'ft^3' or 'm^3'

% Type of Science Data to be returned
SENSOR_DATA = { 'GAS ANALYSIS' 'MASS-CHARGE' 'MEDIUM' };
```

**Output File:**

```
******************************************************************************
*** IN-SITU SENSOR PAYLOAD OPTIMIZATION TOOL ***
******************************************************************************

*** PROGRAM INPUTS ***
User Input Filename: Huygens_GCMS
Planetary Body Name: Titan
Unit System: SI
Print Summary Option: N

Payload Mass Limit: 17.300 kg
Payload Volume Limit: 0.0185 m^3
Payload Power Limit: 79.000 W

******************************************************************************

SENSOR PACKAGES DESIGNED:
******************************************************************************
```
GAS ANALYSIS

*** PAYLOAD SENSOR BREAKDOWN ***

<table>
<thead>
<tr>
<th>SENSOR</th>
<th>MASS</th>
<th>VOLUME</th>
<th>POWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAS ANALYSIS</td>
<td>17.3000</td>
<td>0.0184</td>
<td>71.0000</td>
</tr>
<tr>
<td>SENSOR PAYLOAD TOTAL</td>
<td>17.3000</td>
<td>0.0184</td>
<td>71.0000</td>
</tr>
</tbody>
</table>

*** CASE LIMITS SUMMARY ***

CASE MASS LIMIT: SUCCEEDED!

Designed Mission Sensor Package Meets Mass Limit!
Minimum Sensor Requirements met allows for System Redundancy in Components.

CASE VOLUME LIMIT: SUCCEEDED!

Designed Mission Sensor Package Meets Volume Limit!
Minimum Sensor Requirements met allows for System Redundancy in Components.

CASE POWER LIMIT: SUCCEEDED!

Designed Mission Sensor Package Meets Power Limit!
Minimum Sensor Requirements met allows for System Redundancy in Components.

*** PLANETARY DATABASE SUMMARY RESULTS ***

*** INPUTS ***

Planetary Body Name: TITAN
Unit System: SI

*** PLANETARY BULK PROPERTIES ***

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planetary Mass</td>
<td>1.34550e+023 kg</td>
</tr>
<tr>
<td>Planetary Volume (10^10)</td>
<td>7.15188e+009 m^3</td>
</tr>
<tr>
<td>Planetary Equatorial Radius</td>
<td>2.57500e+006 m</td>
</tr>
<tr>
<td>Planetary Polar Radius</td>
<td>2.57500e+006 m</td>
</tr>
<tr>
<td>Planetary Gravity</td>
<td>1.35200e+000 m/s^2</td>
</tr>
<tr>
<td>Planetary Density</td>
<td>1.88000e+003 kg/m^2</td>
</tr>
<tr>
<td>Planetary Ellipticity</td>
<td>0.00000e+000</td>
</tr>
<tr>
<td>Planetary Escape Velocity</td>
<td>2.63900e+003 m/s</td>
</tr>
<tr>
<td>Planetary Bond Albedo</td>
<td>2.20000e-001</td>
</tr>
<tr>
<td>Planetary Visual Geometric Albedo</td>
<td>2.20000e-001</td>
</tr>
<tr>
<td>Planetary Visual Magnitude</td>
<td>7.90000e+000</td>
</tr>
<tr>
<td>Planetary Solar Irradiance</td>
<td>1.50440e+001 W/m^2</td>
</tr>
<tr>
<td>Planetary Black Body Temperature</td>
<td>0.00000e+000 deg C</td>
</tr>
</tbody>
</table>
Planetary Magnetic Field Strength: 0.00000e+000 mGauss
Planetary Moment of Inertia: 0.00000e+000
Planetary J2 Parameter: 0.00000e+000
Planetary Satellites: 0

*** PLANETARY ORBITAL PARAMETERS ***
Semi-Major Axis: 1.22183e+006 m
Sidereal Orbit Period: 1.59450e+001 Earth Days
Tropical Orbit Period: 1.59450e+001 Earth Days
Perihelion Distance: 0.00000e+000 m
Aphelion Distance: 0.00000e+000 m
Synodic Period: 1.59450e+001 Earth Days
Mean Orbital Velocity: 0.00000e+000 m/s
Maximum Orbital Velocity: 0.00000e+000 m/s
Minimum Orbital Velocity: 0.00000e+000 m/s
Orbital Inclination: 0.00000e+000 deg
Orbital Eccentricity: 0.00000e+000
Sidereal Rotation Period: 2.92000e-002 Earth Hours
Length of Day: 3.82690e+002 Earth Hours
Obliquity to Orbit: 0.00000e+000 deg

*** ATMOSPHERIC PROPERTIES ***
Planetary Surface Temperature: -179.45 deg C
Planetary Surface Pressure: 1.46700e+005 Pa
Planetary Surface Density: 5.50000e+000 kg/m^2

ATMOSPHERIC COMPOSITION
Major Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>% Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2</td>
<td>98.400 %</td>
</tr>
<tr>
<td>CH4</td>
<td>1.600 %</td>
</tr>
</tbody>
</table>

Trace Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Amount</th>
<th>Concentration</th>
</tr>
</thead>
</table>

*** GCMS SENSOR SUMMARY RESULTS ***

*** INPUTS ***
GCMS Operational Type: MASS-CHARGE
GCMS Operational Range: MEDIUM
Unit System: SI

Sensor Properties
GCMS Sensor Type: GCMS-Huygens
Sensing Mass/Charge Range - Low: 2.00 amu
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass Filter Type</td>
<td>Quadrupole</td>
</tr>
<tr>
<td>Detector Operational Type</td>
<td>Electron Multiplier</td>
</tr>
<tr>
<td>Dynamic Mass-charge Scanning Range (amu)</td>
<td>141.0</td>
</tr>
<tr>
<td>Mass-charge Scanning Resolution</td>
<td>0.8</td>
</tr>
<tr>
<td>Minimum Detected Partial Pressure</td>
<td>0.00 A/Torr</td>
</tr>
<tr>
<td>Operating Temperature</td>
<td>10.05 C</td>
</tr>
<tr>
<td>Number of Ion Sources</td>
<td>4</td>
</tr>
<tr>
<td>Charge Field Range (Low:High)</td>
<td>8.00 - 12.00 V</td>
</tr>
<tr>
<td>Electron Source Voltage (Low:High)</td>
<td>8.00 - 12.00 V</td>
</tr>
<tr>
<td>Warm Up Period Mass Stability</td>
<td>31.0</td>
</tr>
<tr>
<td>Sensor Return Data Rate</td>
<td>31.0 bits/sec</td>
</tr>
<tr>
<td>Sensor Mass</td>
<td>17.30 kg</td>
</tr>
<tr>
<td>Sensor Dimensions (Length:Width:Height)</td>
<td>47.00 cm:15.00 cm:5.00 cm</td>
</tr>
<tr>
<td>Power Source Voltage (Typical:Average:Peak)</td>
<td>1.68 A:41.00 W:71.00 W</td>
</tr>
<tr>
<td>Power Supply Voltage Type</td>
<td>DC</td>
</tr>
<tr>
<td>Power Requirements (Typical:Average:Peak)</td>
<td>41.00 W:19.00 W:31.00 W</td>
</tr>
<tr>
<td>Probe design consists of 5 ion sources, quadrupole mass filter, and 2 Electron Multipliers</td>
<td></td>
</tr>
<tr>
<td>Hydrogen gas used as transport medium in sampling chambers</td>
<td></td>
</tr>
<tr>
<td>Huygens Probe GCMS System - Custom Flight Proven configuration</td>
<td></td>
</tr>
</tbody>
</table>
The resultant input design files for the Huygens Atmosphere Structure Instrument package in the ISSPO comparison of the Huygens Mission is detailed in the subsequent data files. In the ISSPO configuration, the Atmospheric suite runs Temperature, Pressure, Wind Velocity, Humidity, and Density. The results for components in the atmospheric suite not part of the HASI suite can be subtracted from the results.

**Input File:**

```matlab
function Huygens_HASI
% *** Declare and Load all Input Variables ***
BODY = 'Titan';
UNIT = 'SI'; % 'SI' or 'British' Units
PRINTFLG = 'N'; % 'Y' or 'N'
MASS_LIMIT = 6.3; % Max Allowable Sensor Package WT 'Ibm' or 'kg'
POWER_LIMIT = 85; % Max Total Power Available 'BTU/hr' or 'W'
VOLUME_LIMIT = 0.5; % Max Allowable Volume 'ft^3' or 'm^3'

% Type of Science Data to be returned
SENSOR_DATA = { 'ATMOSPHERE', 'TEMPERATURE', 'RESISTANCE', 'ACCELERATION', 'ACOUSTICS', 'SENSOR', 'DATA PROCESSING' },
```

**Output File:**

```plaintext
****************************************************
*** IN-SITU SENSOR PAYLOAD OPTIMIZATION TOOL ***
****************************************************

*** PROGRAM INPUTS ***
User Input Filename: Huygens_HASI
Planetary Body Name: Titan
Unit System: SI
Print Summary Option: N
```

```
User Input Filename: Huygens_HASI
Planetary Body Name: Titan
Unit System: SI
Print Summary Option: N
```

```
Payload Mass Limit: 6.300 kg
Payload Power Limit: 85.000 W
```
SENSOR PACKAGES DESIGNED:

ATMOSPHERE
- Temperature
- Pressure
- Wind Velocity
- Humidity
- Density

ACCELERATION
ACOUSTICS
DATA PROCESSING

*** PAYLOAD SENSOR BREAKDOWN ***

<table>
<thead>
<tr>
<th>SENSOR</th>
<th>MASS (kg)</th>
<th>VOLUME (m³)</th>
<th>POWER (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATMOSPHERE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Temperature</td>
<td>0.0004</td>
<td>0.0000</td>
<td>0.0600</td>
</tr>
<tr>
<td>- Pressure</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0250</td>
</tr>
<tr>
<td>- Wind Velocity Probe</td>
<td>0.2270</td>
<td>0.0004</td>
<td>0.0000</td>
</tr>
<tr>
<td>- Handheld Unit</td>
<td>0.0000</td>
<td>0.0003</td>
<td>0.0000</td>
</tr>
<tr>
<td>- Humidity</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>- Density</td>
<td>5.0000</td>
<td>0.0070</td>
<td>0.8250</td>
</tr>
<tr>
<td>ACCELERATION</td>
<td>0.0710</td>
<td>0.0000</td>
<td>0.4800</td>
</tr>
<tr>
<td>ACOUSTICS</td>
<td>0.0040</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>DATA PROCESSING</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.7900</td>
</tr>
<tr>
<td>SENSOR PAYLOAD TOTAL</td>
<td>5.3023</td>
<td>0.0077</td>
<td>2.1800</td>
</tr>
</tbody>
</table>

*** CASE LIMITS SUMMARY ***

CASE MASS LIMIT: SUCCEEDED!
Designed Mission Sensor Package Meets Mass Limit!
Minimum Sensor Requirements met allows for System Redundancy in Components.

CASE VOLUME LIMIT: SUCCEEDED!
Designed Mission Sensor Package Meets Volume Limit!
CASE POWER LIMIT: SUCCEEDED!

Minimum Sensor Requirements met allows for System Redundancy in Components.

Designed Mission Sensor Package Meets Power Limit!
Minimum Sensor Requirements met allows for System Redundancy in Components.

*** PLANETARY DATABASE SUMMARY RESULTS ***

*** INPUTS ***
Planetary Body Name: TITAN
Unit System: SI

PLANETARY BULK PROPERTIES
- Planetary Mass: 1.34550e+023 kg
- Planetary Volume (10^10): 7.15188e+009 m^3
- Planetary Equatorial Radius: 2.57500e+006 m
- Planetary Polar Radius: 2.57500e+006 m
- Planetary Gravity: 1.35200e+000 m/s^2
- Planetary Density: 1.88000e+003 kg/m^3
- Planetary Ellipticity: 0.00000e+000
- Planetary Escape Velocity: 2.63900e+003 m/s
- Planetary GM: 5.94500e+001 m^3/s^2
- Planetary Bond Albedo: 2.20000e-001
- Planetary Visual Geometric Albedo: 2.20000e-001
- Planetary Visual Magnitude: 7.90000e+000
- Planetary Solar Irradiance: 1.50440e+001 W/m^2
- Planetary Black Body Temperature: 2.00000e+000 deg C
- Planetary Magnetic Field Strength: 0.00000e+000 mGauss
- Planetary Moment of Inertia: 0.00000e+000
- Planetary J2 Parameter: 0.00000e+000
- Planetary Satellites: 0

PLANETARY ORBITAL PARAMETERS
- Semi-Major Axis: 2.22183e+006 m
- Sidereal Orbit Period: 1.59450e+001 Earth Days
- Tropical Orbit Period: 1.59450e+001 Earth Days
- Perihelion Distance: 0.00000e+000 m
- Aphelion Distance: 0.00000e+000 m
- Synodic Period: 1.59450e+001 Earth Days
- Mean Orbital Velocity: 0.00000e+000 m/s
- Maximum Orbital Velocity: 0.00000e+000 m/s
- Minimum Orbital Velocity: 0.00000e+000 m/s
- Orbital Inclination: 3.30000e-001 deg
- Orbital Eccentricity: 2.92000e-002
- Sidereal Rotation Period: 3.82690e+002 Earth Hours
- Length of Day: 3.82690e+002 Earth Hours
- Obliquity to Orbit: 0.00000e+000 deg

ATMOSPHERIC PROPERTIES
- Planetary Surface Temperature: 179.45 deg C
- Planetary Surface Pressure: 1.46700e+005 Pa
- Planetary Surface Density: 5.50000e+000 kg/m^3
ATMOSPHERIC COMPOSITION
Major Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>% Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>N₂</td>
<td>98.400 %</td>
</tr>
<tr>
<td>CH₄</td>
<td>1.600 %</td>
</tr>
</tbody>
</table>

Trace Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Amount</th>
<th>Concentration</th>
</tr>
</thead>
</table>

*** TEMPERATURE SENSOR SUMMARY RESULTS ***

*** INPUTS ***
Planetary Surface Temperature: -179.45 deg C
Unit System: SI

Sensor Properties
- **Thermal Sensor Type:** M-0146MD
- **Thermal Sensor Low Limit:** -269.000 deg C
- **Thermal Sensor High Limit:** 400.000 deg C
- **Power Requirements:** 0.060 W
- **Mass Requirements:** 0.350 g
- **Dimensions - Length:** 45.720 mm
  Width: 1.524 mm
  Height: 1.524 mm
- **Operating Pressure:** 0.000 Pascals
- **Error Tolerance:** 0.100 %
  0.260 deg C
- **TRL Level:** 0 TRL

Thermocouple Construction Materials
- **Positive Terminal:** Platinum +Lead
- **Negative Terminal:** Platinum -Lead

EMF Voltage spread over full Temperature Range: 0.000 mV

Sensor Comments:

*** PRESSURE SENSOR SUMMARY RESULTS ***
** INPUTS ***

Planetary Surface Pressure: 146700.00 Pa
Unit System: SI

******************************************************************************

** Sensor Properties **

Pressure Sensor Type: Ser 48-0025
Pressure Sensor Accuracy: 0.500 %
Pressure Sensor Stability: 0.250 % & 1 yr
Over Pressure Rated Value: 2.000 N/A
Burst Pressure Rating: 5.000 N/A
Life Cycles: 1.000E+008 Cycles
Sensing Pressure Range - Low: 0.000 kPa
High: 172.369 kPa
Proof Pressure: 344.738 kPa
Burst Pressure: 861.845 kPa

Environmental Properties
Maximum Shock Load: 100.000 g's
Shock Impulse Time: 0.011 sec
Vibration Shock Limit: 20.000 g's
Operating Frequency Range - Low: 20.000 Hz
High: 200.000 Hz
Operating Humidity Range - Low: 0.000 % RH
High: 95.000 % RH
Operating Temperature Range - Low: -40.000 deg C
High: 104.440 deg C
Storage Temperature Range - Low: -51.100 deg C
High: 121.110 deg C

Power Requirements
Input Voltage: 5.000 V
Input Amperage: 5.000 mA
Sensing Bandwidth: -3.000 dB

Physical Properties
Sensor Material: Stainless Steel
Case Material: Stainless Steel
Dimensions - Length: 48.560 mm
Width: 22.210 mm
Height: 22.230 mm
Sensor Mass: 0.000 gr

Temperature Sensing Range
Lower Sensing Limit: -51.900 deg C
Upper Sensing Limit: 121.100 deg C
Accuracy: 1.100 deg C

Sensor Comments:
*** WIND VELOCITY SENSOR SUMMARY RESULTS ***

** INPUTS **
Wind Velocity Sensor Type: ANEMOMETER
Unit System: SI

** Sensor Properties **
Wind Velocity Sensor Type: HH-30A
Velocity Sensing Range - Low: 0.2032 m/s
High: 39.6240 m/s
Velocity Accuracy: 0.5000 % FS
Volumetric Flow Rate: NO

Environmental Properties
Probe Temperature Range - Low: -20.000 deg C
High: 100.000 deg C
Instrument Temperature Range - Low: 0.000 deg C
High: 50.000 deg C

Power Requirements
Number of Batteries: 2 N/A
Battery Type: AA Type

Physical Properties
Sensor Mass: 0.227 kg
Handheld Dimensions - Length: 180.340 mm
Width: 76.200 mm
Height: 20.320 mm
Probe Dimensions - Length: 124.000 mm
Width: 70.000 mm
Height: 41.000 mm

Sensor Comments:
Compact Design. 3 Extension Rods. 2 or 16 sec Averaging Period 1"/2.75" Probe Diameters.

*** HUMIDITY SENSOR SUMMARY RESULTS ***

** INPUTS **
Unit System: SI

*** WARNING IN HUMIDITY SENSOR PROGRAM ***
Humidity Sensor Not Required In Sensor Package Design!!
Water Component not found in Selected Planetary Atmosphere.

*******************************************************************************
** Sensor Properties **

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Humidity Sensor Type</td>
<td>N/A</td>
</tr>
<tr>
<td>Humidity Sensing Range Low</td>
<td>0.000 % RH</td>
</tr>
<tr>
<td>High</td>
<td>0.000 % RH</td>
</tr>
<tr>
<td>Capacitance at 55% RH Low</td>
<td>0.000 pF</td>
</tr>
<tr>
<td>Typical</td>
<td>0.000 pF</td>
</tr>
<tr>
<td>Max</td>
<td>0.000 pF</td>
</tr>
<tr>
<td>Sensor Sensitivity</td>
<td>0.000 pF/%RH</td>
</tr>
<tr>
<td>Sensor Hysteresis</td>
<td>0.000 %RH</td>
</tr>
<tr>
<td>Response Time</td>
<td>0.000 sec</td>
</tr>
<tr>
<td>Sensor Stability</td>
<td>0.000 %RH/yr</td>
</tr>
</tbody>
</table>

** Environmental Properties **

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature Range Low</td>
<td>0.000 deg C</td>
</tr>
<tr>
<td>High</td>
<td>0.000 deg C</td>
</tr>
<tr>
<td>Sensing Frequency Range Min</td>
<td>0.000 kHz</td>
</tr>
<tr>
<td>Max</td>
<td>0.000 kHz</td>
</tr>
</tbody>
</table>

** Physical Properties **

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor Dimensions - Length</td>
<td>0.000 mm</td>
</tr>
<tr>
<td>Width</td>
<td>0.000 mm</td>
</tr>
<tr>
<td>Height</td>
<td>0.000 mm</td>
</tr>
<tr>
<td>Sensor Mass</td>
<td>0.000 gr</td>
</tr>
</tbody>
</table>

*** DENSITY SENSOR SUMMARY RESULTS ***

*** INPUTS ***

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density Sensor Operating Type</td>
<td>GAS</td>
</tr>
<tr>
<td>Planetary Surface Density</td>
<td>5.50 kg/m^3</td>
</tr>
<tr>
<td>Unit System</td>
<td>SI</td>
</tr>
</tbody>
</table>
Vibration Loads: 0.000 g's

Physical Properties
Sensor Mass: 0.050 kg
Sensor Dimensions - Length: 364.000 mm
Width: 139.000 mm
Height: 139.000 mm

Power Requirements
Input Voltage: 33.000 V-DC
Input Amperage: 25.000 mA

Sensor Comments:
Gas blending & Direct measurement of ethylene density - Process Gas Must be dry

*** ACCELEROMETER SENSOR SUMMARY RESULTS ***

*** INPUTS ***
Mission Acceleration Profile: MEDIUM
Accelerometer Power Type: Const Current
Unit System: SI

** Sensor Properties **
Accelerometer Sensor Type: QA2000-030
Accelerometer Sensing Range - Low: -60.00 g's
High: 60.00 g's
Sensor Sensitivity: 0.00 mV/g
Transverse Sensitivity: 0.00 %
Operating Frequency Range - Low: 20.00 Hz
High: 2000.00 Hz
Sensor Linearity: 0.00 %
Resonance Frequency: 0.00 Hz
Shock Limits: 250.00 g's
Vibration Limits: 15.00 g's

Power Requirements
Input Type: Const Current
Current Range - Low: 16.00 mA
High: 16.00 mA
Bias Voltage: 28.00 V
Bias Voltage Type: DC
Electrical Noise: 0.30 mG's
Electrical Power: 0.48 W

Environmental Properties
Sensor Temperature Range - Low: -65.00 deg C
High: 95.00 deg C
Temperature Sensitivity: 0.000 g/s/deg C
### Physical Properties

- **Sensor Mass:** 71.000 gr
- **Sensor Dimensions - Length:** 3.350 cm
- **Width:** 3.350 cm
- **Height:** 2.718 cm

**Sensor Comments:**
- Flight Proven - Huygens Lander Descent phase HASI.

### Acoustic Sensor Summary Results

#### **Inputs**

- **Unit System:** SI

#### **Sensor Properties**

- **Acoustic Sensor Type:** CT-190M
- **Dynamic Sensing Range - Low:** 56.80 dB
- **High:** 135.00 dB
- **Operating Frequency Range - Low:** 10.00 Hz
- **High:** 2000.00 Hz
- **Sensitivity at 250 Hz - Range:** 0.00 dB
  - **Low:** 0.00 dB
  - **High:** 0.00 dB
- **Resonance Frequency:** 500000.00 Hz
- **Inherent Noise:** 56.80 dB
- **Clipping Limit:** 135.00 dB
- **Stability Conditions - Time:** 0.00 dB/yr
- **Temperature:** 0.00 deg C
- **Relative Humidity:** 0.00 %RH

#### Power Requirements

- **Voltage Range - Min:** 10.00 V
- **Max:** 15.00 V
- **Nominal Current:** 0.00 mA

#### Environmental Properties

- **Operational Temperature Range - Low:** -195.50 deg C
  - **High:** 37.00 deg C
- **Storage Temperature Range - Low:** -195.50 deg C
  - **High:** 120.00 deg C
- **Temperature Coefficient at 250 Hz:** 0.00 dB/deg C
- **Pressure Coefficient at 250 Hz:** 0.00 dB/kPa

#### Physical Properties

- **Sensor Dimensions - Length:** 34.360 mm
- **Width:** 9.500 mm
Sensor Mass:
Height: 9.500 mm
4.000 gram

Sensor Comments:

*** DIGITAL SIGNAL PROCESSING RESULTS ***

** INPUTS **
CPU Operating Speed: LOW
Unit System: SI

** Sensor Properties **
Digital Signal Processing Unit Type: ADSP-2100
CPU Operating Speed: LOW
CPU Clock Speed: 8.192 MHz
Number of CPU Processing Cores: 1 #
Amount of On-Board Data Memory: 16.0 K-Words
Number of Data Bins: 16 #
Amount of On-Board Program Memory: 32.0 K-Words
Number of Memory Bins: 24 #
Number of Computational Units: 3 #
CPU Cycle Time: 80.000 ns
Number of External Interrupts: 4 #

Power Requirements
CPU Output Power: 790.000 mW
Input Voltage - Min: -0.300 V
Max: 7.000 V
Voltage Type: DC N/A
Input Current: 100.000 mA

Physical Properties
Processor Core Mass: 0.00 kg
Sensor Dimensions - Length: 33.830 mm
Width: 33.830 mm
Height: 9.120 mm

Environmental Properties
Operating Temperature Range - Low: -55.00 deg C
High: 125.00 deg C
Storage Temperature Range - Low: -65.00 deg C
High: 150.00 deg C

Sensor Comments:
Dual Purpose Program Memory for Both Instruction and Data Storage
Three Independent Computational Units: ALU, Multiplier/Accumulator and Barrel Shifter
Two Independent Data Address Shifters
Powerful Program Sequencer
Internal Instruction Cache
Provisions for Multiprecision Computation and Saturation Logic
Single-Cycle Instruction Execution
Multifunction Instructions
APPLICATIONS
- Optimized for DSP Algorithms including: Digital Filtering, Fast Fourier Transforms
- Image Processing
- Radar, Sonar
- Speech Processing
- Telecommunications
5. Surface Science Package (SSP)

The resultant input design files for the Surface Science Package in the ISSPO comparison of the Huygens Mission is detailed in the subsequent data files. In the ISSPO configuration the Atmospheric suite runs Temperature, Pressure, Wind Velocity, Humidity, and Density. The results for components in the atmospheric suite not part of the SSP can be subtracted from the results.

Input File:

```matlab
function Huygens_SSP
%*** Declare and Load all Input Variables ***
BODY = 'Titan';
UNITS = 'SI'; % 'SI' or 'British' Units
PRINTFLG = 'N'; % 'Y' or 'N'
MASS_LIMIT = 3.9; % Max Allowable Sensor Package Wt 'lbm' or 'kg'
POWER_LIMIT = 11; % Max Total Power Available 'BTU's/hr' or 'W'
VOLUME_LIMIT = 0.1; % Max Allowable Volume 'ft3' or 'm3'

% Type of Science Data to be returned
SENSOR_DATA = {
    'ATMOSPHERE', 'TEMPERATURE', 'RESISTANCE',
    'INCLINATION', 'HIGH', 'N/A',
    'ACCELERATION', 'IMPACT', 'Self Generating',
    'ACOUSTICS', 'VELOCITY', 'N/A',
    'REFRACTION', 'N/A', 'N/A'};
```

Output File:

```
*** IN-SITU SENSOR PAYLOAD OPTIMIZATION TOOL ***

User Input Filename: Huygens_SSP
Planetary Body Name: Titan
Unit System: SI
Print Summary Option: N

Payload Mass Limit: 3.900 kg
Payload Volume Limit: 0.1000 m3
Payload Power Limit: 11.000 W
```
SENSOR PACKAGES DESIGNED:

- **ATMOSPHERE**
  - Temperature
  - Pressure
  - Wind Velocity
  - Humidity
  - Density

- **INCLINATION**

- **ACCELERATION**

- **ACOUSTICS**

- **REFRACTION**

---

<table>
<thead>
<tr>
<th>SENSOR PAYLOAD TOTAL</th>
<th>MASS</th>
<th>VOLUME</th>
<th>POWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2864</td>
<td>0.0089</td>
<td>10.9100</td>
<td></td>
</tr>
</tbody>
</table>
CASE MASS LIMIT: FAILED!

- Designed Mission Sensor Package EXCEEDS Mass Limits
- Database Configuration for Required Sensors Exceeds Input Sensor Package Mass Limit
- Reconfigure Sensor Package for Less Components or Select Multiple Data Type Sensors

CASE VOLUME LIMIT: SUCCEEDED!

- Designed Mission Sensor Package Meets Volume Limit!
- Minimum Sensor Requirements met allows for System Redundancy in Components.

CASE POWER LIMIT: SUCCEEDED!

- Designed Mission Sensor Package Meets Power Limit!
- Minimum Sensor Requirements met allows for System Redundancy in Components.

**PLANETARY DATABASE SUMMARY RESULTS**

**INPUTS**
- Planetary Body Name: TITAN
- Unit System: SI

**PLANETARY BULK PROPERTIES**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planetary Mass</td>
<td>1.34550e+023 kg</td>
</tr>
<tr>
<td>Planetary Volume (10^10)</td>
<td>7.15188e+009 m^3</td>
</tr>
<tr>
<td>Planetary Equatorial Radius</td>
<td>2.57500e+006 m</td>
</tr>
<tr>
<td>Planetary Polar Radius</td>
<td>2.57500e+006 m</td>
</tr>
<tr>
<td>Planetary Gravity</td>
<td>1.35200e+000 m/s^2</td>
</tr>
<tr>
<td>Planetary Density</td>
<td>1.88000e+003 Kg/m^2</td>
</tr>
<tr>
<td>Planetary Escape Velocity</td>
<td>2.63900e+003 m/s</td>
</tr>
<tr>
<td>Planetary GM</td>
<td>0.00000e+000 km^3/s^2</td>
</tr>
<tr>
<td>Planetary Bond Albedo</td>
<td>2.20000e-001</td>
</tr>
<tr>
<td>Planetary Visual Geometric Albedo</td>
<td>2.00000e+001</td>
</tr>
<tr>
<td>Planetary Visual Magnitude</td>
<td>7.90000e+000</td>
</tr>
<tr>
<td>Planetary Solar Irradiance</td>
<td>1.50440e+001 W/m^2</td>
</tr>
<tr>
<td>Planetary Black Body Temperature</td>
<td>0.00000e+000 deg C</td>
</tr>
<tr>
<td>Planetary Magnetic Field Strength</td>
<td>0.00000e+000 mGauss</td>
</tr>
<tr>
<td>Planetary Moment of Inertia</td>
<td>0.00000e+000</td>
</tr>
<tr>
<td>Planetary J2 Parameter</td>
<td>0.00000e+000</td>
</tr>
<tr>
<td>Planetary Satellites</td>
<td>0</td>
</tr>
</tbody>
</table>

**PLANETARY ORBITAL PARAMETERS**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semi-Major Axis</td>
<td>1.22183e+006 m</td>
</tr>
<tr>
<td>Sidereal Orbit Period</td>
<td>1.59450e+001 Earth Days</td>
</tr>
<tr>
<td>Tropical Orbit Period</td>
<td>1.59450e+001 Earth Days</td>
</tr>
<tr>
<td>Perihelion Distance</td>
<td>0.00000e+000 m</td>
</tr>
<tr>
<td>Aphelion Distance</td>
<td>0.00000e+000 m</td>
</tr>
<tr>
<td>Symodic Period</td>
<td>1.59450e+001 Earth Days</td>
</tr>
<tr>
<td>Mean Orbital Velocity</td>
<td>0.00000e+000 m/s</td>
</tr>
<tr>
<td>Maximum Orbital Velocity</td>
<td>0.00000e+000 m/s</td>
</tr>
<tr>
<td>Minimum Orbital Velocity</td>
<td>0.00000e+000 m/s</td>
</tr>
<tr>
<td>Orbital Inclination</td>
<td>3.30000e-001 deg</td>
</tr>
<tr>
<td>Orbital Eccentricity</td>
<td>2.92000e-002</td>
</tr>
<tr>
<td>Sidereal Rotation Period</td>
<td>3.82690e+002 Earth Hours</td>
</tr>
</tbody>
</table>
Length of Day: 3.82690e+002 Earth Hours
Obliquity to Orbit: 0.00000e+000 deg

*** ATMOSPHERIC PROPERTIES ***
Planetary Surface Temperature: -179.45 deg C
Planetary Surface Pressure: 1.46700e+005 Pa
Planetary Surface Density: 5.50000e+000 kg/m²

ATMOSPHERIC COMPOSITION
Major Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>% Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>N₂</td>
<td>98.400 %</td>
</tr>
<tr>
<td>CH₄</td>
<td>1.600 %</td>
</tr>
</tbody>
</table>

Trace Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Amount</th>
<th>Concentration</th>
</tr>
</thead>
</table>

*** TEMPERATURE SENSOR SUMMARY RESULTS ***

*** INPUTS ***
Planetary Surface Temperature: -179.45 deg C
Unit System: SI

** Sensor Properties **
Thermal Sensor Type: M-0146MD
Thermal Sensor Low Limit: -269.000 deg C
Thermal Sensor High Limit: 400.000 deg C
Power Requirements: 0.060 W
Mass Requirements: 0.350 g
Dimensions: Length: 45.720 mm
           Width: 1.524 mm
           Height: 1.524 mm
Operating Pressure: 0.000 Pascals
Error Tolerance: 0.100 %
                0.260 deg C
TRL Level: 0 TRL

Thermocouple Construction Materials
Positive Terminal: Platinum +Lead
Negative Terminal: Platinum -Lead

EMF Voltage spread over full Temperature Range: 0.000 mV

Sensor Comments:

*** PRESSURE SENSOR SUMMARY RESULTS ***

*** INPUTS ***
Planetary Surface Pressure: 146700.00 Pa
Unit System: SI

******************************************************************************

** Sensor Properties **
Pressure Sensor Type: Ser 48-0025
Pressure Sensor Accuracy: 0.500 %
Pressure Sensor Stability: 0.250 % @ 1 yr
Over Pressure Rated Value: 2.000 N/A
Burst Pressure Rating: 5.000 N/A
Life Cycles: 1,000+008 Cycles
Sensing Pressure Range - Low: 0.000 kPa
High: 172.369 kPa
Proof Pressure: 344.738 kPa
Burst Pressure: 861.845 kPa

Environmental Properties
Maximum Shock Load: 100.000 g's
Shock Impulse Time: 0.011 sec
Vibration Shock Limit: 20.000 g's
Operating Frequency Range - Low: 20.000 Hz
High: 2000.000 Hz
Operating Humidity Range - Low: 0.000 % RH
High: 95.000 % RH
Operating Temperature Range - Low: -40.000 deg C
High: 104.440 deg C
Storage Temperature Range - Low: -51.100 deg C
High: 121.110 deg C

Power Requirements
Input Voltage: 5.000 V
Input Amperage: 5.000 mA
Sensing Bandwidth: -3.000 dB

Physical Properties
Sensor Material: Stainless Steel
Case Material: Stainless Steel
Dimensions - Length: 48.560 mm
Width: 22.230 mm
Height: 22.230 mm
Sensor Mass: 0.000 gr

Temperature Sensing Range
Lower Sensing Limit: -53.900 deg C
Upper Sensing Limit: 121.100 deg C
Accuracy: 1.100 deg C


*** WIND VELOCITY SENSOR SUMMARY RESULTS ***

*** INPUTS ***
Wind Velocity Sensor Type: ANEMOMETER
Unit System: SI

** Sensor Properties **
Wind Velocity Sensor Type: HH-30A
Velocity Sensing Range - Low: 0.2032 m/s
Velocity Sensing Range - High: 39.6240 m/s
Velocity Accuracy: 0.5000 % FS
Volumetric Flow Rate: NO

Environmental Properties
Probe Temperature Range - Low: -20.000 deg C
Probe Temperature Range - High: 100.000 deg C
Instrument Temperature Range - Low: 0.000 deg C
Instrument Temperature Range - High: 50.000 deg C

Power Requirements
Number of Batteries: 2
Battery Type: AA

Physical Properties
Sensor Mass: 0.227 kg
Handheld Dimensions - Length: 180.340 mm
Handheld Dimensions - Width: 76.200 mm
Handheld Dimensions - Height: 20.320 mm
Probe Dimensions - Length: 124.000 mm
Probe Dimensions - Width: 70.000 mm
Probe Dimensions - Height: 41.000 mm

Sensor Comments: Compact Design. 3 Extension Rods. 2 or 16 sec Averaging Period. 1"/2.75" Probe Diameters.

*** HUMIDITY SENSOR SUMMARY RESULTS ***

*** INPUTS ***
Unit System: SI
*** WARNING IN HUMIDITY SENSOR PROGRAM ***
Humidity Sensor Not Required In Sensor Package Design!!
Water Component not found in Selected Planetary Atmosphere.

** Sensor Properties **
Humidity Sensor Type: N/A
Humidity Sensing Range - Low: 0.000 % RH
High: 0.000 % RH
Capacitance at 55% RH - Low: 0.000 pF
Typical: 0.000 pF
Max: 0.000 pF
Sensor Sensitivity: 0.000 pF/%RH
Sensor Hysteresis: 0.000 +-%RH
Response Time: 0.000 sec
Sensor Stability: 0.000 %RH/yr

Environmental Properties
Temperature Range - Low: 0.000 deg C
High: 0.000 deg C
Sensing Frequency Range - Min: 0.000 kHz
Max: 0.000 kHz

Physical Properties
Sensor Dimensions - Length: 0.000 mm
Width: 0.000 mm
Height: 0.000 mm
Sensor Mass: 0.000 gr

*** DENSITY SENSOR SUMMARY RESULTS ***

*** INPUTS ***
Density Sensor Operating Type: GAS
Planetary Surface Density: 5.50 kg/m^3
Unit System: SI

** Sensor Properties **
Density Sensor Type: GDT 7812
Density Sensing Range - Low: 0.001 g/cc
High: 4.000 g/cc
Sensor Accuracy: 0.001000 g/cc
Sensor Repeatability: 0.002000 g/cc
Sensor Tolerance: 0.000000 g/cc
Sensor Viscosity: 0.000 cP
Sensing Flowrates: 216.000 ltr/hr
Environmental Properties
Temperature Range - Low: -20.000 deg C  High: 85.000 deg C
Test Pressure: 0.000 kPa
Max Operating Pressure: 15000.000 kPa
Vibration Loads: 0.000 g's

Physical Properties
Sensor Mass: 5.000 kg
Sensor Dimensions - Length: 364.000 mm
Width: 139.000 mm
Height: 139.000 mm

Power Requirements
Input Voltage: 33.000 V-DC
Input Amperage: 25.000 mA

Sensor Comments:
Gas blending & Direct measurement of ethylene density - Process Gas Must be dry

*** INCLINOMETER SENSOR SUMMARY RESULTS ***

*** INPUTS ***
Inclination Angle Range: HIGH
Unit System: SI

** Sensor Properties **
Inclinometer Sensor Type: L-211U
Sensing Inclination Range - Low: -60.000 deg
High: 60.000 deg
Output Data Format: Analog mV
Sensor Resolution: 0.0083000 deg
Sensor Sensitivity: 100.000 mV/deg
Sensor Non-Linearity: 3.3300 %
Response Time: 0.000 sec
Sensor Bandwidth: 0.00 Hz

Environmental Properties
Operating Temperature Range - Low: -54.000 deg C
High: 124.000 deg C
Storage Temperature Range - Low: -54.000 deg C
High: 124.000 deg C

Power Requirements
Input Voltage Type: AC N/A
Input Voltage: 0 V
Physical Properties
Dimensions - Length: 41.000 mm
          Width: 18.400 mm
          Height: 15.900 mm
Sensor Mass: 0.000 gram

Sensor Comments:
Two Single Axis Sensors Mounted on Single Unit.
Wide angular range, high accuracy, dynamic output attenuation.
L Series incorporates a specially designed dampening orifice.
Includes hermetic sealing, compact size, and are available in a variety of housing configurations.
Applications - Industrial, Aerospace, Military, Photonics, Geotechnical, Oceanographic, Construction.
Flight Rated Production Company.

---

*** ACCELEROMETER SENSOR SUMMARY RESULTS ***

---

*** INPUTS ***
Mission Acceleration Profile: IMPACT
Accelerometer Power Type: Self Generating
Unit System: SI

---

** Sensor Properties **
Accelerometer Sensor Type: 2271A/AM20
Accelerometer Sensing Range - Low: 0.00 g/s
          High: 10000.00 g/s
Sensor Sensitivity: 0.00 mV/g
Transverse Sensitivity: 3.00 %
Operating Frequency Range - Low: 0.50 Hz
          High: 7000.00 Hz
Sensor Linearity: 1.00 %
Resonance Frequency: 27000.00 Hz
Shock Limits: 10000.00 g/s
Vibration Limits: 1000.00 g/s

Power Requirements
Input Type: Self Generating
Current Range - Low: 0.00 mA
          High: 0.00 mA
Bias Voltage: 0.00 V
Bias Voltage Type: Charge
Electrical Noise: 0.30 mG/s
Electrical Power: 0.00 W

Environmental Properties
Sensor Temperature Range - Low: -269.00 deg C
          High: 260.30 deg C
Temperature Sensitivity: 0.000 g/s/deg C
Physical Properties
Sensor Mass: 27.000 gr
Sensor Dimensions - Length: 2.920 cm
Width: 1.588 cm
Height: 1.588 cm

Sensor Comments:
Single Axis, wide temperature range piezoelectric accelerometer for cryogenic applications. Self generating power device.
Flight Proven - Huygens Lander SSP.

*** ACOUSTIC SENSOR SUMMARY RESULTS ***

*** INPUTS ***
Unit System: SI

** Sensor Properties **
Acoustic Sensor Type: R80-Alpha
Dynamic Sensing Range - Low: 30.00 dB
High: 62.00 dB
Operating Frequency Range - Low: 200000.00 Hz
High: 1000000.00 Hz
Sensitivity at 250 Hz - Range: 0.00 dB
Tolerance - Low: 0.00 dB
High: 0.00 dB
Resonance Frequency: 800000.00 Hz
Inherent Noise: 58.00 dB
Clipping Limit: 120.00 dB
Stability Conditions - Time: 0.00 dB/yr
Temperature: 0.00 deg C
Relative Humidity: 0.00 %RH

Power Requirements
Voltage Range - Min: 0.00 V
Max: 0.00 V
Nominal Current: 0.00 mA

Environmental Properties
Operational Temperature Range - Low: -65.00 deg C
High: 175.00 deg C
Storage Temperature Range - Low: -65.00 deg C
High: 175.00 deg C
Temperature Coefficient at 250 Hz: 0.00 dB/deg C
Pressure Coefficient at 250 Hz: 0.00 dB/kPa

Physical Properties
Sensor Dimensions - Length: 19.000 mm
Width: 19.000 mm
Sensor Comments:
General Purpose Acoustic Sensor. High Frequency sensor. Robust, Reliable Sensor.

*** REFRACTION SENSOR SUMMARY RESULTS ***

*** INPUTS ***

Unit System: SI

Sensor Comments:
Unique Design Solution - Combination NMOS Linear Image Sensor and Sapphire Refraction Prism
Self-Scanning Photodiode Array
Linear critical-angle refractometer.
Flight Proven Hardware configuration - Huygens Lander SSP.
APPENDIX C

Venus Atmospheric Properties Mission Design

The attached design file and output file are for the Venus Atmospheric Properties Mission Design. The goal is to better understand the atmospheric properties, and will investigate: Atmospherics, Inclination, Acceleration, EM Field, Radiation, and Optical properties.

**Input File:**

```matlab
function Venus_ATM
%*** Declare and Load all Input Variables ***
BODY = 'Venus';
UNITSYSTEM = 'SI'; % 'SI' or 'British' Units
PRINTFLG = 'N'; % 'Y' or 'N'
MASS LIMIT = 30;
POWER LIMIT = 200;
VOLUME LIMIT = 0.50;

% Max Allowable Sensor Package Wt 'lbm' or 'kg'
% Max Total Power Available 'BTU's/hr' or 'W'
% Max Allowable Volume 'ft^3' or 'm^3'

% Type of Science Data to be returned
SENSOR_DATA = {'ATMOSPHERE', 'INCLINATION', 'ACCELERATION', 'EM FIELD', 'RADIATION', 'OPTICS'};

%*** IN-SITU SENSOR PAYLOAD OPTIMIZATION TOOL ***
```

**Output File:**

```
*******************************************************************************************
*** IN-SITU SENSOR PAYLOAD OPTIMIZATION TOOL ***
*******************************************************************************************

*** PROGRAM INPUTS ***
User Input Filename: Venus_ATM
Planetary Body Name: Venus
Unit System: SI
Print Summary Option: N

Payload Mass Limit: 30.000 kg
Payload Volume Limit: 0.5000 m^3
```
Payload Power Limit:  200,000  W

SENSOR PACKAGES DESIGNED:

ATMOSPHERE
- Temperature
- Pressure
- Wind Velocity
- Humidity
- Density

INCLINATION

ACCELERATION

EM FIELD

RADIATION
- Charged Particle

OPTICS
- ARRAY
- MEDIUM
- VISUAL
- UV
- NIR

*** PAYLOAD SENSOR BREAKDOWN ***

<table>
<thead>
<tr>
<th>SENSOR</th>
<th>MASS (kg)</th>
<th>VOLUME (m³)</th>
<th>POWER (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATMOSPHERE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Pressure</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Wind Velocity Probe</td>
<td>0.5000</td>
<td>0.0004</td>
<td>0.1000</td>
</tr>
<tr>
<td>Handheld Unit</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Humidity</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Density</td>
<td>5.0000</td>
<td>0.0070</td>
<td>0.8250</td>
</tr>
<tr>
<td>INCLINATION</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>ACCELERATION</td>
<td>0.1417</td>
<td>0.0000</td>
<td>0.6400</td>
</tr>
<tr>
<td>EM FIELD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Handheld Unit</td>
<td>0.3100</td>
<td>0.0004</td>
<td>0.0000</td>
</tr>
<tr>
<td></td>
<td>0.7700</td>
<td>0.0012</td>
<td>0.0000</td>
</tr>
</tbody>
</table>
RADIATION | 7.0000 | 0.0067 | 15.0000
---|---|---|---
OPTICS | 0.0000 | 0.0000 | 0.0000
---|---|---|---
SENSOR PAYLOAD TOTAL | 13.7217 | 0.0159 | 16.5950

*** CASE LIMITS SUMMARY ***

CASE MASS LIMIT: SUCCEEDED!
Designed Mission Sensor Package Meets Mass Limit!
Minimum Sensor Requirements met allows for System Redundancy in Components.

CASE VOLUME LIMIT: SUCCEEDED!
Designed Mission Sensor Package Meets Volume Limit!
Minimum Sensor Requirements met allows for System Redundancy in Components.

CASE POWER LIMIT: SUCCEEDED!
Designed Mission Sensor Package Meets Power Limit!
Minimum Sensor Requirements met allows for System Redundancy in Components.

*** PLANETARY DATABASE SUMMARY RESULTS ***

*** INPUTS ***
Planetary Body Name: VENUS
Unit System: SI

*** PLANETARY BULK PROPERTIES ***

Planetary Mass: 4.86900e+024 kg
Planetary Volume (10^10): 9.28430e+010 m^3
Planetary Equatorial Radius: 6.05180e+005 m
Planetary Polar Radius: 6.05180e+006 m
Planetary Gravity: 8.87000e+000 m/s^2
Planetary Density: 5.24300e+003 kg/m^2
Planetary Ellipticity: 0.00000e+000
Planetary Escape Velocity: 1.03600e+004 m/s
Planetary GM: 3.24900e+005 km^3/s^2
Planetary Bond Albedo: 7.50000e-001
Planetary Visual Geometric Albedo: 6.50000e-001
Planetary Visual Magnitude: -4.40000e+000
Planetary Solar Irradiance: 2.61390e+003 W/m^2
Planetary Black Body Temperature: 2.31700e+002 deg C
Planetary Magnetic Field Strength: 0.00000e+000 mGauss
Planetary Moment of Inertia: 3.30000e-001
Planetary J2 Parameter: 4.45800e+000
Planetary Satellites

*** PLANETARY ORBITAL PARAMETERS ***

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semi-Major Axis</td>
<td>1.08210e+011 m</td>
</tr>
<tr>
<td>Sidereal Orbit Period</td>
<td>2.24701e+002 Earth Days</td>
</tr>
<tr>
<td>Tropical Orbit Period</td>
<td>2.24695e+002 Earth Days</td>
</tr>
<tr>
<td>Perihelion Distance</td>
<td>1.07480e+011 m</td>
</tr>
<tr>
<td>Aphelion Distance</td>
<td>1.08940e+011 m</td>
</tr>
<tr>
<td>Synodic Period</td>
<td>5.83920e+002 Earth Days</td>
</tr>
<tr>
<td>Mean Orbital Velocity</td>
<td>3.50200e+004 m/s</td>
</tr>
<tr>
<td>Maximum Orbital Velocity</td>
<td>3.52600e+004 m/s</td>
</tr>
<tr>
<td>Minimum Orbital Velocity</td>
<td>3.47900e+004 m/s</td>
</tr>
<tr>
<td>Orbital Inclination</td>
<td>3.93000e+000 deg</td>
</tr>
<tr>
<td>Orbital Eccentricity</td>
<td>6.70000e+002</td>
</tr>
<tr>
<td>Sidereal Rotation Period</td>
<td>-5.83250e+003 Earth Hours</td>
</tr>
<tr>
<td>Length of Day</td>
<td>2.80200e+003 Earth Hours</td>
</tr>
<tr>
<td>Obliquity to Orbit</td>
<td>1.77360e+002 deg</td>
</tr>
</tbody>
</table>

*** ATMOSPHERIC PROPERTIES ***

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planetary Surface Temperature</td>
<td>464.00 deg C</td>
</tr>
<tr>
<td>Planetary Surface Pressure</td>
<td>9.20000e+006 Pa</td>
</tr>
<tr>
<td>Planetary Surface Density</td>
<td>6.44000e+001 kg/m^2</td>
</tr>
</tbody>
</table>

**ATMOSPHERIC COMPOSITION**

<table>
<thead>
<tr>
<th>Element</th>
<th>% Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO2</td>
<td>96.500 %</td>
</tr>
<tr>
<td>N2</td>
<td>3.500 %</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element</th>
<th>Amount</th>
<th>Concentration</th>
</tr>
</thead>
<tbody>
<tr>
<td>SO2</td>
<td>150.000 ppm</td>
<td></td>
</tr>
<tr>
<td>Ar</td>
<td>70.000 ppm</td>
<td></td>
</tr>
<tr>
<td>H2O</td>
<td>20.000 ppm</td>
<td></td>
</tr>
<tr>
<td>CO</td>
<td>17.000 ppm</td>
<td></td>
</tr>
<tr>
<td>He</td>
<td>12.000 ppm</td>
<td></td>
</tr>
<tr>
<td>Ne</td>
<td>7.000 ppm</td>
<td></td>
</tr>
</tbody>
</table>

*** TEMPERATURE SENSOR SUMMARY RESULTS ***

*** INPUTS ***

Planetary Surface Temperature: 464.00 deg C
Unit System: SI

*******************************************************************************
** Sensor Properties **

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermal Sensor Type</td>
<td>K</td>
</tr>
<tr>
<td>Thermal Sensor Low Limit</td>
<td>-200.000 deg C</td>
</tr>
<tr>
<td>Thermal Sensor High Limit</td>
<td>1250.000 deg C</td>
</tr>
<tr>
<td>Power Requirements</td>
<td>0.000 W</td>
</tr>
<tr>
<td>Mass Requirements</td>
<td>0.000 g</td>
</tr>
<tr>
<td>Dimensions - Length</td>
<td>0.000 mm</td>
</tr>
<tr>
<td></td>
<td>Width: 0.000 mm</td>
</tr>
<tr>
<td></td>
<td>Height: 0.000 mm</td>
</tr>
<tr>
<td>Operating Pressure</td>
<td>0.000 Pascals</td>
</tr>
<tr>
<td>Error Tolerance</td>
<td>0.750 %</td>
</tr>
<tr>
<td>TRL Level</td>
<td>0</td>
</tr>
</tbody>
</table>

Thermocouple Construction Materials

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Material</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>Ni-Cr +Lead</td>
</tr>
<tr>
<td>Negative</td>
<td>Ni-Al -Lead</td>
</tr>
</tbody>
</table>

EMF Voltage spread over full Temperature Range: 48.428 mV

Sensor Comments:
Clean Oxidizing and Inert. Limited Use in Vacuum or Reducing. Wide Temperature Range, Most Popular Calibration.

** INSULATION MATERIAL PROPERTIES **

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insulation Type</td>
<td>GG</td>
</tr>
<tr>
<td>Insulation Lower Limit</td>
<td>-73.000 deg C</td>
</tr>
<tr>
<td>Insulation Upper Limit</td>
<td>482.000 deg C</td>
</tr>
</tbody>
</table>

Environmental Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abrasion Resistance</td>
<td>Poor</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Good</td>
</tr>
<tr>
<td>Water Submersion</td>
<td>Poor</td>
</tr>
<tr>
<td>Resistance to Solvents</td>
<td>Excellent</td>
</tr>
<tr>
<td>Resistance to Acids</td>
<td>Excellent</td>
</tr>
<tr>
<td>Resistance to Bases</td>
<td>Excellent</td>
</tr>
<tr>
<td>Resistance to Flame</td>
<td>Excellent</td>
</tr>
<tr>
<td>Resistance to Humidity</td>
<td>Fair</td>
</tr>
<tr>
<td>Overall Material Construction</td>
<td>Glass Braid</td>
</tr>
<tr>
<td>Conductor Material Construction</td>
<td>Glass Braid</td>
</tr>
</tbody>
</table>

*** Pressure Sensor Summary Results ***

*** Inputs ***

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planetary Surface Pressure</td>
<td>9200000.00 Pa</td>
</tr>
<tr>
<td>Unit System</td>
<td>SI</td>
</tr>
</tbody>
</table>

** Sensor Properties **

Pressure Sensor Type: 13(C,U)2000P
Pressure Sensor Accuracy: 2.000 \%  
Pressure Sensor Stability: 0.600 \% \pm 1 \text{ yr}  
Over Pressure Rated Value: 3.000 N/A  
Burst Pressure Rating: 5.000 N/A  
Life Cycles: 1.000 \times 10^6 \text{ Cycles}  
Sensing Pressure Range - Low: 0.000 kPa  
High: 13789.500 kPa  
Proof Pressure: 41368.500 kPa  
Burst Pressure: 68947.600 kPa  

Environmental Properties  
Maximum Shock Load: 100.000 g's  
Shock Impulse Time: 0.011 sec  
Vibration Shock Limit: 10.000 g's  
Operating Frequency Range - Low: 20.000 Hz  
High: 2000.000 Hz  
Operating Humidity Range - Low: 0.000 \% RH  
High: 0.000 \% RH  
Operating Temperature Range - Low: -40.000 deg C  
High: 125.000 deg C  
Storage Temperature Range - Low: -40.000 deg C  
High: 125.000 deg C  

Power Requirements  
Input Voltage: 15.000 V  
Input Amperage: 2.000 mA  
Sensing Bandwidth: 0.000 dB  

Physical Properties  
Sensor Material: Stainless Steel  
Case Material: Stainless Steel  
Dimensions - Length: 81.030 mm  
Width: 19.050 mm  
Height: 19.050 mm  
Sensor Mass: 0.000 gr  

Temperature Sensing Range  
Lower Sensing Limit: 0.000 deg C  
Upper Sensing Limit: 0.000 deg C  
Accuracy: 0.000 deg C  

Sensor Comments:  

*** WIND VELOCITY SENSOR SUMMARY RESULTS ***  

*** INPUTS ***  
Wind Velocity Sensor Type: ANEMOMETER  
Unit System: SI  

*************************************************************************************************************************************
** Sensor Properties **

Wind Velocity Sensor Type: FT 702
Velocity Sensing Range - Low: 0.0000 m/s
High: 70.0000 m/s
Velocity Accuracy: 4.0000 % FS
Volumetric Flow Rate: NO

Environmental Properties
Probe Temperature Range - Low: -40.000 deg C
High: 85.000 deg C
Instrument Temperature Range - Low: -40.000 deg C
High: 85.000 deg C

Power Requirements
Number of Batteries: 7
Battery Type: 14.000 mA
Voltage Input: 100.000 mW
Voltage Type: DC N/A
Current Input: 0.000 N/A
Input Power: 0.000 N/A

Physical Properties
Sensor Mass: 0.500 kg
Handheld Dimensions - Length: 0.000 mm
Width: 0.000 mm
Height: 0.000 mm
Probe Dimensions - Length: 162.000 mm
Width: 50.000 mm
Height: 50.000 mm

Sensor Comments:
High accuracy wind speed and direction sensing. WIND DIRECTION MEASUREMENT. Compact, unobtrusive solid-state design with no moving parts.

*** HUMIDITY SENSOR SUMMARY RESULTS ***

*** INPUTS ***
Unit System: SI

*** WARNING IN HUMIDITY SENSOR PROGRAM ***
Humidity Sensor Not Required In Sensor Package Design!!
Water Component not found in Selected Planetary Atmosphere.

** Sensor Properties **
Humidity Sensor Type: N/A
Humidity Sensing Range - Low: 0.000 % RH
High: 0.000 % RH
Capacitance at 55% RH - Low: 0.000 pF
Typical: 0.000 pF
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor Sensitivity:</td>
<td>0.000 pF</td>
</tr>
<tr>
<td>Sensor Hysteresis:</td>
<td>0.000 %RH</td>
</tr>
<tr>
<td>Response Time:</td>
<td>0.000 sec</td>
</tr>
<tr>
<td>Sensor Stability:</td>
<td>0.000 %RH/yr</td>
</tr>
<tr>
<td>Temperature Range - Low:</td>
<td>0.000 deg C</td>
</tr>
<tr>
<td>Temperature Range - High:</td>
<td>0.000 deg C</td>
</tr>
<tr>
<td>Sensing Frequency Range - Min:</td>
<td>0.000 kHz</td>
</tr>
<tr>
<td>Sensing Frequency Range - Max:</td>
<td>0.000 kHz</td>
</tr>
<tr>
<td>Sensor Dimensions - Length:</td>
<td>0.000 mm</td>
</tr>
<tr>
<td>Sensor Dimensions - Width:</td>
<td>0.000 mm</td>
</tr>
<tr>
<td>Sensor Dimensions - Height:</td>
<td>0.000 mm</td>
</tr>
<tr>
<td>Sensor Mass:</td>
<td>0.000 gr</td>
</tr>
<tr>
<td>Density Sensor Operating Type:</td>
<td>GAS</td>
</tr>
<tr>
<td>Planetary Surface Density:</td>
<td>64.40 kg/m^3</td>
</tr>
<tr>
<td>Unit System:</td>
<td>SI</td>
</tr>
<tr>
<td>Density Sensor Type:</td>
<td>GDT 7812</td>
</tr>
<tr>
<td>Density Sensing Range - Low:</td>
<td>0.001 g/cc</td>
</tr>
<tr>
<td>Density Sensing Range - High:</td>
<td>4.000 g/cc</td>
</tr>
<tr>
<td>Sensor Accuracy:</td>
<td>0.001000 g/cc</td>
</tr>
<tr>
<td>Sensor Repeatability:</td>
<td>0.0020000 g/cc</td>
</tr>
<tr>
<td>Sensor Tolerance:</td>
<td>0.000600 g/cc</td>
</tr>
<tr>
<td>Sensor Viscosity:</td>
<td>0.000 cP</td>
</tr>
<tr>
<td>Sensing Flowrates:</td>
<td>216.000 ltr/hr</td>
</tr>
<tr>
<td>Temperature Range - Low:</td>
<td>-20.000 deg C</td>
</tr>
<tr>
<td>Temperature Range - High:</td>
<td>85.000 deg C</td>
</tr>
<tr>
<td>Test Pressure:</td>
<td>0.000 kPa</td>
</tr>
<tr>
<td>Max Operating Pressure:</td>
<td>15000.000 kPa</td>
</tr>
<tr>
<td>Vibration Loads:</td>
<td>0.000 g''</td>
</tr>
<tr>
<td>Sensor Mass:</td>
<td>5.000 kg</td>
</tr>
<tr>
<td>Sensor Dimensions - Length:</td>
<td>364.000 mm</td>
</tr>
<tr>
<td>Sensor Dimensions - Width:</td>
<td>139.000 mm</td>
</tr>
<tr>
<td>Sensor Dimensions - Height:</td>
<td>139.000 mm</td>
</tr>
<tr>
<td>Power Requirements</td>
<td></td>
</tr>
</tbody>
</table>
Input Voltage: 33.000 V-DC
Input Amperage: 25.000 mA

Sensor Comments:
Gas blending & Direct measurement of ethylene density - Process Gas Must be dry

*** INCLINOMETER SENSOR SUMMARY RESULTS ***

*** INPUTS ***
Inclination Angle Range: MEDIUM
Unit System: SI

** Sensor Properties **
Inclinometer Sensor Type: L-212T
Sensing Inclination Range - Low: -45.000 Deg
High: 45.000 Deg
Output Data Format: Analog mV
Sensor Resolution: 0.0056000 Deg
Sensor Sensitivity: 150.000 mV/deg
Sensor Non-Linearity: 2.2200 %
Response Time: 0.000 sec
Sensor Bandwidth: 0.00 Hz

Environmental Properties
Operating Temperature Range - Low: -54.000 deg C
High: 124.000 deg C
Storage Temperature Range - Low: -54.000 deg C
High: 124.000 deg C

Power Requirements
Input Voltage Type: AC N/A
Input Voltage: 0 V

Physical Properties
Dimensions - Length: 41.000 mm
Width: 18.400 mm
Height: 15.900 mm
Sensor Mass: 0.000 gram

Sensor Comments:
Two Single Axis Sensors Mounted on Single Unit
Wide angular range, high accuracy, dynamic output attenuation
L Series incorporates a specially designed dampening orifice
Includes hermetic sealing, compact size, and are available in a variety of housing configurations
Applications - Industrial, Aerospace, Military, Photonics, Geotechnical, Oceanographic, Construction.
Flight Rated Production Company
*** ACCELEROMETER SENSOR SUMMARY RESULTS ***

*** INPUTS ***
Mission Acceleration Profile: MEDIUM
Accelerometer Power Type: Voltage
Unit System: SI

** Sensor Properties **
Accelerometer Sensor Type: MA15
Accelerometer Sensing Range - Low: 50.00 g's
High: -50.00 g's
Sensor Sensitivity:
High: 0.00 mV/g
Transverse Sensitivity:
High: 5.00 %
Operating Frequency Range - Low: 2.00 Hz
High: 1000.00 Hz
Sensor Linearity:
High: 1.00 %
Resonance Frequency:
High: 5000.00 Hz
Shock Limits:
Low: 0.00 g's
Vibration Limits:
Low: 0.00 g's

Power Requirements
Input Type: Voltage Type
Current Range - Low: 4.00 mA
High: 20.00 mA
Bias Voltage:
High: 0.00 V
Bias Voltage Type: DC Type
Electrical Noise:
High: 0.30 mG's
Electrical Power:
High: 0.64 W

Environmental Properties
Sensor Temperature Range - Low: -23.30 deg C
High: 98.90 deg C
Temperature Sensitivity:
High: 0.261 g's/deg C

Physical Properties
Sensor Mass:
High: 141.700 gr
Sensor Dimensions - Length:
High: 5.690 cm
Width:
High: 2.856 cm
Height:
High: 2.896 cm

Sensor Comments:
General Purpose. 4 - 20 mA Proportional to G.

*** FIELD SENSOR SUMMARY RESULTS ***

*** INPUTS ***
Magnetic Field Strength: 0.000 mGauss
** Sensor Properties **

AC Field Sensor Type: TAM - 1
Sensor Axis Number: 3 N/A
AC Field Sensing Range - Low: 0.00 mG
High: 1000.00 mG
Operating Frequency Range - Low: 0.00 Hz
High: 60.00 Hz
Sensor Accuracy: 1.00 %
Number of Digit Accuracy: 0 N/A
Sampling Time Interval: 0.0000 sec
Sensor Tolerance: 10.00 mV/mG

Environmental Properties
Sensor Temperature Range - Low: -90.000 deg C
High: 80.000 deg C
Electronics Temperature Range - Low: -34.000 deg C
High: 71.000 deg C

Physical Properties
Sensor Dimensions - Length: 4.760 cm
Width: 6.680 cm
Height: 11.900 cm
Sensor Mass: 310.000 gram
Electronics Dimensions - Length: 5.230 cm
Width: 14.020 cm
Height: 16.390 cm
Sensor Mass: 770.000 gram

Power Requirements
Input Voltage: 9.000 V
Voltage Type: DC N/A

Sensor Comments:
TAM-1 series parts meet NASA and military programs (MIL-S1D-975). Fully Space Qualified

*** RADIATION SENSOR SUMMARY RESULTS ***

*** INPUTS ***

Unit System: SI

** Sensor Properties **

Radiation Sensor Type: LPD
Radiation Detected: Charged Particle
Proton Energy Range - Low: 1.000 MeV
High: 250.000 MeV
Number of Detection Bins: 15 # of Bins
Electron Energy Range - Low: 0.500 MeV
High: 20.000 MeV
Number of Detection Bins: 7 # of Bins
Alpha Particle Range - Low: 6.000 MeV
High: 250.000 MeV
Number of Detection Bins: 6 # of Bins
Nucleon/Heavy Ion Range - Low: 1.500 MeV
High: 0.000 MeV
Number of Detection Bins: 0 # of Bins
Beta Particle Range - Low: 0.000 keV
High: 0.000 keV
Number of Detection Bins: 0 # of Bins
Gamma Particle Range - Low: 0.000 keV
High: 0.000 keV
Number of Detection Bins: 0 # of Bins
X-Ray Energy Range - Low: 0.000 keV
High: 0.000 keV
Number of Detection Bins: 0 # of Bins
Particle Detection Rate: 100.000 kcps
G-Factor: 0.050

Physical Properties
Sensor Mass: 7.000 kg
Dimensions - Length: 150.000 mm
Width: 150.000 mm
Height: 300.000 mm
Power Requirement: 15.000 W

Sensor Comments:
16 Different Operational Modes
Excellent Particle Specificity
High Level Of Redundancy
Radiation Hardened Components
Fully Flight Qualified: DRTS-E, MDS-1, and ISS

*** IMAGING SENSOR SUMMARY RESULTS ***

*** INPUTS ***
System Type: ARRAY
Resolution: MEDIUM
Image Type: VISUAL UV NIR
Unit System: SI

** Sensor Properties **
Imaging Sensor Type: CCD 3041
### Imaging Spectrum:

<table>
<thead>
<tr>
<th>Spectrum</th>
<th>X-RAY</th>
<th>UV</th>
<th>VISUAL</th>
<th>NIR</th>
<th>MICRO</th>
</tr>
</thead>
</table>

### Array Dimensions

<table>
<thead>
<tr>
<th>Length</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>2048 # Pixels</td>
<td>2048 # Pixels</td>
</tr>
</tbody>
</table>

### Sensor Pixel Size

<table>
<thead>
<tr>
<th>Length</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.0000 micro-m</td>
<td>30.720 mm</td>
</tr>
</tbody>
</table>

### Imaging Array Size

<table>
<thead>
<tr>
<th>Length</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.720 mm</td>
<td>30.720 mm</td>
</tr>
</tbody>
</table>

### Read Noise at 1 MHz

- 250 kHz: 7 e-
- 0 e-

### Full Well Capacity

- Pixel: 0 e-
- Register: 750 ke-

### Sensor Gain

- 0.00 e-/ADU
- 0.00 %

### Sensor Linearity

ADC Dynamic Range: N/A N/A

### Readout Rates

<table>
<thead>
<tr>
<th>Rate</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0000 MHz</td>
<td>1.0000 MHz</td>
</tr>
</tbody>
</table>

### Readout Time

<table>
<thead>
<tr>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.100000 sec</td>
<td>0.364000 sec</td>
</tr>
</tbody>
</table>

### Frame Rates

<table>
<thead>
<tr>
<th>Rate</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.90</td>
<td>2.70</td>
</tr>
<tr>
<td>5.60 fps</td>
<td></td>
</tr>
</tbody>
</table>

### Environmental Properties

#### Operating Temperature Range

- Low: -100.00 deg C
- High: 40.00 deg C

#### Cooled Temperature

- 0.00 deg C

#### Cooling Method

N/A N/A

### Physical Properties

#### Dimensions

<table>
<thead>
<tr>
<th>Length</th>
<th>Width</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>45.72 mm</td>
<td>36.83 mm</td>
<td>2.01 mm</td>
</tr>
</tbody>
</table>

#### Sensor Mass

0.000 kg

### Power Requirements

<table>
<thead>
<tr>
<th>Input Voltage</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>24.000 V</td>
<td></td>
</tr>
</tbody>
</table>

| Input Voltage Type |
| DC Type |

<table>
<thead>
<tr>
<th>Input Amperage</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000 mA</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Power Requirement</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000 mW</td>
<td></td>
</tr>
</tbody>
</table>

### Sensor Comments

- Back-illuminated CCD Array
- Highest quantum efficiency
- Multiple output image formats
- Four-port readout
- Optimal design for speed and sensitivity
- Separate, optimized readout channels for lowest noise and highest speed
- Space Qualified - Production Company
- Software Gain & Binning
- Scientific precision and accuracy
APPENDIX D

ISSPO Program Module Code

D1. ISSPO Main Program
% function ISSPO
% Developed by: Keith Schreck
%                Mechanical and Aerospace Engineering
%                San Jose State University
% Date:         Fall 2007
% The In-Situ Sensor Payload Optimization (ISSPO) Tool is developed as
% part of the requirements for a Master's Degree in Aerospace Engineering
% from San Jose State University for the AE 295 Project Class.
% The tool calculates a preliminary design for an optimized sensor pack
% for a mission to a planet or into space based on given mission
% requirements for mass, power, volume, or mission goal. This main
% program calls multiple subroutine programs to obtain pertinent data on
% the planet under observation, and the required components.
% Program inputs are loaded into a single data input file called from the
% main program execution. This input format allows quick alteration of
% the inputs and rapid exaction without having to reenter all the program
% variables.

function ISSPO
clc;  
clear all;  
close all;  
format compact;  

% ***Print out Program header***
fprintf('********************************************************************
');  
fprintf('*** IN-SITU SENSOR PAYLOAD OPTIMIZATION TOOL ***');  
InputFile = input('

Enter a Input Filename to use: ', 's');
if exist(InputFile) == 2
    fprintf('

***Loading Input Design Parameters from File.***
');
    InputFileM = [ InputFile '.m' ];
else
    cd PRGM_FILES;
    ERROR_PRG(1);
    %cd ..;
    return;
end

% Delete Summary File from Prior Case if It Exists
if exist('ISSPO_SUMMARY.txt') == 0
    delete('ISSPO_SUMMARY.txt');
end

% MAKE Case Working Directory
cd WORK_DIR;
if exist(InputFile) == 7
    cd(InputFile);
    delete '*';
    cd DIR_FILES;
    delete '*';
    cd ..;
    % Changes Directory to WORK DIR
    cd ...
    % Returns to Main Directory
else
    mkdir(InputFile);
    cd(InputFile);
    mkdir DIR_FILES;
    cd ..;
end
    cd ..
    % Return to Main Directory

% Get Program Files from Program Folder
cd PRGM_FILES;
dirpath = [' . . /WORK_DIR/ ' InputFile '/DIR_FILES' ];
copyfile('*.m', dirpath);
    cd ..;
    dp2 = ['WORK_DIR/ ' InputFile '/DIR_FILES/InputDeck.m' ];
copyfile(InputFileM, dp2);

% Change Directory to DIR_FILES Working Directory to Run Programs
cd WORK_DIR;
cd(InputFile);
cd DIR_FILES;

% Loads InputFile Parameters from InputDeck
InputDeck;
Fname = ['InputFile '.mat '];
if exist(Fname) == 2
    load(Fname);
else
    ERROR_PRG(2);
    return;
end

% Verify InputDeck Program Values
if (strcmpi(UNITS,'SI') == 0) && (strcmpi(UNITS,'British') == 0)
    ERROR_PRG(50);
    return;
end

if (strcmpi(PRINTFLG, 'Y') == 0) && (strcmpi(PRINTFLG, 'N') == 0)
    ERROR_PRG(51);
    return;
end

% If MASS_LIMIT is undefined Call ERROR Program
if exist('MASS LIMIT') == 0
    ERROR_PRG(53);
    return;
end

% If POWER_LIMIT is undefined Call ERROR Program
if exist('POWER LIMIT') == 0
    ERROR_PRG(54);
    return;
end

% If VOLUME_LIMIT is undefined Call ERROR Program
if exist('VOLUME_LIMIT') == 0
    ERROR_PRG(55);
    return;
end

% Sensor Data Verifier Program
% Checks through input Sensors for invalid types
DataTypeERR_FLG = DataTypeVerifier(SENSOR_DATA);

% Error Flag Conditions Calls ERROR Program
if DataTypeERR_FLG == 1
    ERROR_PRG(52);
    return;
end

if DataTypeERR_FLG == 2
    ERROR_PRG(56);
    return;
end

if DataTypeERR_FLG == 3
    ERROR_PRG(57);
    return;
end
if DataTypeERR_FLG == 4
    ERROR_PRG(58);
    return;
end
if DataTypeERR_FLG == 5
    ERROR_PRG(59);
    return;
end
if DataTypeERR_FLG == 6
    ERROR_PRG(60);
    return;
end
if DataTypeERR_FLG == 7
    ERROR_PRG(61);
    return;
end
if DataTypeERR_FLG == 8
    ERROR_PRG(62);
    return;
end
if DataTypeERR_FLG == 9
    ERROR_PRG(63);
    return;
end
if DataTypeERR_FLG == 10
    ERROR_PRG(64);
    return;
end
if DataTypeERR_FLG == 11
    ERROR_PRG(65);
    return;
end
if DataTypeERR_FLG == 12
    ERROR_PRG(66);
    return;
end
if DataTypeERR_FLG == 13
    ERROR_PRG(67);
    return;
end
if DataTypeERR_FLG == 14
    ERROR_PRG(68);
    return;
end
if DataTypeERR_FLG == 15
    ERROR_PRG(69);
    return;
end
if DataTypeERR_FLG == 16
    ERROR_PRG(70);
    return;
end
if DataTypeERR_FLG == 17
    ERROR_PRG(71);
return;
end
if DataTypeERR_FLG == 10
ERROR_PRG(72);
return;
end
if DataTypeERR_FLG == 19
ERROR_PRG(73);
return;
end
if DataTypeERR_FLG == 20
ERROR_PRG(74);
return;
end
if DataTypeERR_FLG == 21
ERROR_PRG(75);
return;
end
if DataTypeERR_FLG == 22
ERROR_PRG(76);
return;
end
if DataTypeERR_FLG == 23
ERROR_PRG(77);
return;
end
if DataTypeERR_FLG == 24
ERROR_PRG(78);
return;
end
if DataTypeERR_FLG == 25
ERROR_PRG(79);
return;
end
if DataTypeERR_FLG == 26
ERROR_PRG(80);
return;
end
if DataTypeERR_FLG == 27
ERROR_PRG(81);
return;
end
if DataTypeERR_FLG == 28
ERROR_PRG(82);
return;
end
if DataTypeERR_FLG == 29
ERROR_PRG(83);
return;
end
if DataTypeERR_FLG == 30
ERROR_PRG(84);
return;
end
% Load Constants Subroutine
Constants(UNITS);
if exist('Constants_DB.mat') == 2
    load('Constants_DB.mat');
else
    ERROR_PRG(3);
    return;
end

% Load Planetary Data File
PlanetaryDatabase(BODY,UNITS,PRINTFLG);
if exist('Planet_Data.mat') == 2
    load('Planet_Data.mat');
else
    ERROR_PRG(4);
    return;
end

% Copies Summary file to Working Directory and Main directory
movefile('PlanetDB_Summary.txt','../');

% Design Loop for Each Sensor Element
for i = 1:size(SENSOR_DATA,1)
    % Runs the Atmospheric Data Sensor Suite Package
    if strcmpi(SENSOR_DATA{i,1},'ATMOSPHERE') == 1
        % Prints Program Sensor Header to Screen
        disp(' *** Executing ATMOSPHERIC Suite! ***');
        % Display Program Module Header
        disp(' ___ *** Executing TEMPERATURE Sensor! ***');
        % Load Temperature Sensor Data into program
        % Logic Check for Temperature Sensor Type
        for temp = 2:2:size(SENSOR_DATA,2)
            if strcmpi(SENSOR_DATA{i,temp}, 'TEMPERATURE') == 1
                TEMP_SENS_TYPE = SENSOR_DATA{i,temp+1};
            end
        end
        if exist('TEMP_SENS_TYPE') == 0
            TEMP_SENS_TYPE = 'VOLTAGE';
        end
        % Load Temperature Sensor Data into program
        TEMP_SENSORS(TEMP_SENS_TYPE,Planet_Properties{7},UNITS,BODY,PRINTFLG);
        if exist('TEMP_SENSOR_Data.mat') == 2
            load('TEMP_SENSOR_Data.mat');
            % Verify Sensor Summary File is created Properly
            if exist('TEMPERATURE_SENSOR_Summary.txt') == 2
                % Copies Summary file to Working Directory and Main directory
                movefile('TEMPERATURE_SENSOR_Summary.txt','../');
            else
                % Error handling for missing Sensor Summary file
                % Code for error handling...
            end
            % End of Temperature Sensor Loop
        end
    end
end
% Display Program Module Header
disp(' *** Executing PRESSURE Sensor! ***');

% Load Pressure Sensor Data into program
PRESSURE_SENSORS(Planet_Properties{8},UNITS,PRINTFLG);
if exist(['PRESSURE_SENSOR_Data.mat']) == 2
load(['PRESSURE_SENSOR_Data.mat']);
% Verify Sensor Summary File is created Properly
if exist('PRESSURE_SENSOR_Summary.txt') == 2
% Copies Summary file to Working Directory and Main directory
movefile('PRESSURE_SENSOR_Summary.txt','../');
else
ERROR_PRG(201);
return;
end
else
ERROR_PRG(101);
return;
end

% Display Program Module Header
disp(' *** Executing WIND VELOCITY Sensor! ***');
% Load Wind Velocity Sensor Data into program
% Logic Check for Wind Velocity Sensor Type
for pos = 2:2:size(SENSOR_DATA,2)
if strcmpi(SENSOR_DATA{i,pos},'WIND VELOCITY') == i
WV_SENS_TYPE = SENSOR_DATA{i,pos+1};
end
end
if exist('WV_SENS_TYPE') == 0
WV_SENS_TYPE = 'ANEMOMETER';
end
WIND_VELOCITY_SENSORS(WV_SENS_TYPE,UNITS,PRINTFLG);
if exist(['WIND_VELOCITY_SENSOR_Data.mat']) == 2
load(['WIND_VELOCITY_SENSOR_Data.mat']);
% Verify Sensor Summary File is created Properly
if exist('WIND_VELOCITY_SENSOR_Summary.txt') == 2
% Copies Summary file to Working Directory and Main directory
movefile('WIND_VELOCITY_SENSOR_Summary.txt','../');
else
ERROR_PRG(207);
return;
end
else
ERROR_PRG(107);
return;
end

% Display Program Module Header
disp(' *** Executing HUMIDITY Sensor! ***');
% Load Humidity Sensor Data into program
HUMIDITY_SENSORS(UNITS,PRINTFLG);
if exist('HUMIDITY_SENSOR_Data.mat') == 2
load('HUMIDITYSENSOR_Data.mat');
% Verify Sensor Summary File is created Properly
if exist('HUMIDITY_SENSOR_Summary.txt') == 2
% Copies Summary file to Working Directory and Main directory
movefile('HUMIDITY_SENSOR_Summary.txt','../');
else
ERROR_PRG(203);
return;
end
else
ERROR_PRG(103);
return;
end

% Display Program Module Header
disp(' *** Executing DENSITY Sensor! ***');
% Load Density Sensor Data into program
% Logic Check for Density Sensor Type
for temp = 2:2:size(SENSOR_DATA,2)
if strcmpi(SENSOR_DATA{i,temp}, 'DENSITY') == 1
DENS_SENS_TYPE = SENSOR_DATA{i,temp+1};
end
end
if exist('DENS_SENS_TYPE') == 0
DENS_SENS_TYPE = 'GAS';
end
DENSITY_SENSORS(DENS_SENS_TYPE,Planet_Properties(18),UNITS,PRINTFLG);
if exist('DENSITY_SENSOR_Data.mat') == 2
load('DENSITYSENSOR_Data.mat');
% Verify Sensor Summary File is created Properly
if exist('DENSITY_SENSOR_Summary.txt') == 2
% Copies Summary file to Working Directory and Main directory
movefile('DENSITY_SENSOR_Summary.txt','../');
else
ERROR_PRG(204);
return;
end
else
ERROR_PRG(104);
return;
end
end
% Ends Atmospheric sensor Suite Package

% Runs the Acoustic Sensor Program
if strcmpi(SENSOR_DATA{i,1}, 'ACOUSTIC') == 1
    % Prints Program Sensor Header to Screen
    disp(' *** Executing ACOUSTIC Sensor! ***');
    % Load Acoustic Sensor Data into program
    ACS_SENS_TYPE = SENSOR_DATA{i,2};
    ACOUSTIC_SENSORS(ACS_SENS_TYPE, UNITS, PRINTFLG);
    if exist('ACOUSTIC_SENSOR_Data.mat') == 2
        load('ACOUSTIC_SENSOR_Data.mat');
    end
    % Verify Sensor Summary File is created Properly
    if exist('ACOUSTIC_SENSOR_Summary.txt') == 2
        % Copies Summary file to Working Directory and Main directory
        movefile('ACOUSTIC_SENSOR_Summary.txt', '../');
    else
        ERROR_PRG(205);
        return;
    end
else
    ERROR_PRG(105);
    return;
end

% Runs the EM Field Sensor Program
if strcmpi(SENSOR_DATA{i,1}, 'EM FIELD') == 1
    % Prints Program Sensor Header to Screen
    disp(' *** Executing EM FIELD Sensor! ***');
    % Load AC Field Sensor Data into program
    AC_FIELD_SENSORS(Planet_Properties{19}, UNITS, PRINTFLG);
    if exist('AC_FIELD_SENSOR_Data.mat') == 2
        load('AC_FIELD_SENSOR_Data.mat');
    end
    % Verify Sensor Summary File is created Properly
    if exist('AC_FIELD_SENSOR_Summary.txt') == 2
        % Copies Summary file to Working Directory and Main directory
        movefile('AC_FIELD_SENSOR_Summary.txt', '../');
    else
        ERROR_PRG(206);
        return;
    end
else
    ERROR_PRG(106);
    return;
end
% Runs the Acceleration Sensor Program
if strcmp(SENSOR_DATA{i,1},'ACCELERATION') == 1

% Prints Program Sensor Header to Screen
disp(' *** Executing ACCELEROMETER Sensor! ***');

%Assigns SENSOR_DATA Inputs to Program Variables
A_TYPE = SENSOR_DATA{i,2};
P_TYPE = SENSOR_DATA{i,3};
NUM_ACCs = SENSOR_DATA{i,4};

% Load Accelerometer Sensor Data into program
ACCELEROMETER_SENSORS(A_TYPE, P_TYPE, UNITS, PRINTFLG);
if exist('ACCELEROMETER_SENSOR_Data.mat') == 2
load('ACCELEROMETER_SENSOR_Data.mat');
% Verify Sensor Summary File is created Properly
if exist('ACCELEROMETER_SENSOR_Summary.txt') == 2
% Copies Summary file to Working Directory and Main directory
movefile('ACCELEROMETER_SENSOR_Summary.txt','../');
else
ERROR_PRG(208);
return;
else
ERROR_PRG(108);
return;
end
end

% Runs the GCMS Sensor Program
if strcmpi(SENSOR_DATA{i,1},'GAS ANALYSIS') == 1

% Prints Program Sensor Header to Screen
disp(' *** Executing GAS ANALYSIS Sensor! ***');

%Assigns SENSOR_DATA Inputs to Program Variables
GCMS_SENS_TYPE = SENSOR_DATA{i,2};
GCMS_SENS_RANGE = SENSOR_DATA{i,3};
GCMS_SENS_RANGE = SENSOR_DATA{i,4};

% Load GCMS Sensor Data into program
GCMS_SENSORS(GCMS_SENS_TYPE, GCMS_SENS_RANGE, UNITS, PRINTFLG);
if exist('GCMS_SENSOR_Data.mat') == 2
load('GCMS_SENSOR_Data.mat');
% Verify Sensor Summary File is created Properly
if exist('GCMS_SENSOR_Summary.txt') == 2
% Copies Summary file to Working Directory and Main directory
movefile('GCMS_SENSOR_Summary.txt','../');
else
ERROR_PRG(209);
return;
end

end
else
    ERROR_PRG(109);
    return;
end

% Runs the Nephelometer Sensor Program
if strcmpi(SENSOR_DATA{i,1}, 'NEPHELOMETRY') == 1
    % Load Nephelometer Sensor Data into program
    % Prints Program Sensor Header to Screen
disp(' *** Executing NEPHELOMETER Sensor! ***');
    NEPHELOMETER_SENSORS(UNITS, PRINTFLG);
    if exist('NEPHELOMETER_SENSOR_Data.mat') == 2
        load('NEPHELOMETER_SENSOR_Data.mat');
        % Verify Sensor Summary File is created Properly
        if exist('NEPHELOMETER_SENSOR_Summary.txt') == 2
            % Copies Summary file to Working Directory and Main directory
            movefile('NEPHELOMETER_SENSOR_Summary.txt', '../');
        else
            ERROR_PRG(202);
            return;
        end
    else
        ERROR_PRG(102);
        return;
    end
end

% Runs the Radiation Sensor Program
if strcmpi(SENSOR_DATA{i,1}, 'RADIATION') == 1
    % Load Radiation Sensor Data into program
    % Prints Program Sensor Header to Screen
disp(' *** Executing RADIATION Sensor! ***');
    % Assigns SENSOR_DATA Inputs to Program Variables
    % Counts the length of the Radiation Type Array till End or sees N/A
    cntr = 0;
    for pt = 2:size(SENSOR_DATA,2)
        if strcmpi(SENSOR_DATA{i,pt}, 'N/A') == 0
            cntr = cntr + 1;
        else
            break;
        end
    end
    RAD = cellstr(SENSOR_DATA{i,2:1+cntr});
    RADIATION_SENSORS(RAD, UNITS, PRINTFLG);
    if exist('RADIATION_SENSOR_Data.mat') == 2
        load('RADIATION_SENSOR_Data.mat');
        % Verify Sensor Summary File is created Properly
if exist('RADIATION_SENSOR_Summary.txt') == 2
% Copies Summary file to Working Directory and Main directory
movefile('RADIATION_SENSOR_Summary.txt','../');
else
   ERROR_PRG(210);
   return;
end
else
   ERROR_PRG(110);
   return;
end

end

% Runs the Inclinometer Sensor Program
if strcmpi(SENSOR_DATA{i,1},'INCLINATION') == 1
% Prints Program Sensor Header to Screen
disp(' *** Executing INCLINATION Sensor! ***');
% Assigns SENSOR_DATA Inputs to Program Variables
INCL_RANGE ={ SENSOR_DATA{ i , 2 } };
% Load Inclinometer Sensor Data into program
INCLINOMETER_SENSORS(INCL_RANGE,UNITS,PRINTFLG);
if exist('INCLINOMETER_SENSOR_Data.mat') == 2
load('INCLINOMETER_SENSOR_Data.mat');
% Verify Sensor Summary File is created properly
if exist('INCLINOMETER_SENSOR_Summary.txt') == 2
% Copies Summary file to Working Directory and Main directory
movefile('INCLINOMETER_SENSOR_Summary.txt','../');
else
   ERROR_PRG(211);
   return;
end
else
   ERROR_PRG(111);
   return;
end
end

% Runs the Optical Imaging Sensor Program
if strcmpi(SENSOR_DATA{i,1},'OPTICS') == 1
% Prints Program Sensor Header to Screen
disp(' *** Executing OPTICS Sensor! ***');
% Assigns SENSOR_DATA Inputs to Program Variables
SYS_FUNC_TYPE = SENSOR_DATA{i,2};
RES = SENSOR_DATA{i,3};
% Counts the length of the Image Type Array till End or sees N/A
for pt = 4:size(SENSOR_DATA,2)
if strcmpi(SENSOR_DATA{i,pt},'N/A') == 0
    IMAGE{cntr} = SENSOR_DATA{i,pt};
    cntr = cntr + 1;
else
    break;
end

% IMAGE = cellstr(SENSOR_DATA{i,4:4+cntr});
% IMAGING_SENSORS(SYS_FUNC_TYPE,RES,IMAGE,UNITS,PRINTFLG);
% if exist('IMAGING_SENSOR_Data.mat') == 2
%    load('IMAGING_SENSOR_Data.mat');
%    % Verify Sensor Summary File is created Properly
%    if exist('IMAGING_SENSOR_Summary.txt') == 2
%        % Copies Summary file to Working Directory and Main directory
%        movefile('IMAGING_SENSOR_Summary.txt','../');
%    else
%        ERROR_PRG{212};
%        return;
%    end
% else
%    ERROR_PRG{112};
%    return;
% end

% Runs the Refraction Sensor Program
if strcmpi(SENSOR_DATA{i,1},'REFRACTION') == 1
    % Prints Program Sensor Header to Screen
    disp(' *** Executing REFRACTION Sensor! ***');
    % Load Refraction Sensor Data into program
    REFRACtion_SENSORS(UNITS,PRINTFLG);
    if exist('REFRACTION_SENSOR_Data.mat') == 2
        load('REFRACTION_SENSOR_Data.mat');
        % Verify Sensor Summary File is created Properly
        if exist('REFRACTION_SENSOR_Summary.txt') == 2
            % Copies Summary file to Working Directory and Main directory
            movefile('REFRACTION_SENSOR_Summary.txt','../');
        else
            ERROR_PRG{213};
            return;
        end
    else
        error('Sensors Sensor_Summary.txt', ' cannot be found. Round again.');
        return;
    end

% Runs the Digital Signal Processing Program
if strcmpi(SENSOR_DATA{i,1},'DATA PROCESSING') == 1
% Prints Program Sensor Header to Screen
disp('  *** Executing DIGITAL SIGNAL PROCESSING: ***');

% Assigns SENSOR_DATA Inputs to Program Variables
DIG_SIG_PROG_TYPE = SENSOR_DATA{1,2};
% Load Digital Signal Processing Data into program
DIGITAL_SIGNAL_PROCESSING(DIG_SIG_PROG_TYPE,UNITS,PRINTFLG);
if exist('DIGITAL_SIGNAL_PROCESSING_Data.mat') == 2
   load('DIGITAL_SIGNAL_PROCESSING_Data.mat');
else
   ERROR_PRG(214);
   return;
end
% Verify Sensor Summary File is created Properly
if exist('DIGITAL_SIGNAL_PROCESSING_Summary.txt') == 2
   % Copies Summary file to Working Directory and Main directory
   movefile('DIGITAL_SIGNAL_PROCESSING_Summary.txt','../');
else
   ERROR_PRG(214);
   return;
end

end
end
% Ends FOR Program Design Loop
% Main program Components After individual Optimization Programs is Run
% ISSPO Summary Optimization Section
% Determine the number of Sensor Programs Runs
NUM_SENSORS = size(SENSOR_DATA,1);
% Offset for Atmospheric Suite comprised of 5 Sensors
for i = 1:size(SENSOR_DATA,1)
   if strcmpi(SENSOR_DATA{i}, 'ATMOSPHERE') == 1
      NUM_SENSORS = NUM_SENSORS + 4;
   end
end
% Runs MASS_PROPERTIES Program Obtains Sensor Mass and Dimensions
if exist('MASS_PROPERTIES_Data.mat') == 2
   load('MASS_PROPERTIES_Data.mat');
else
   ERROR_PRG(215);
   return;
end
% Determine Each Sensors Power Requirements
if exist('POWER_PROPERTIES_Data.mat') == 2
   load('POWER_PROPERTIES_Data.mat');
else
% Determine Total Mass of all Selected Sensors
TOTAL_MASS = sum(MASS_PROPS(:,1));

% Determine Total Volume of all Selected Sensors
for i = 1:size(MASS_PROPS,1)
    VOLUME_SUM(i) = MASS_PROPS(i,2) \* MASS_PROPS(i,3) \* MASS_PROPS(i,4);
end
TOTAL_VOLUME = sum(VOLUME_SUM);

POWER_TOTAL = sum(POWER_PROPS);

% Determine Volume for Wind Velocity Sensor Handheld Unit if Chosen in Analysis
if exist('MASS_PROPS2') == 1
    TOTAL_MASS = TOTAL_MASS + MASS_PROPS2(1,1);
    VOLUME_SUM2 = MASS_PROPS2(1,2) \* MASS_PROPS2(1,3) \* MASS_PROPS2(1,4);
    TOTAL_VOLUME = TOTAL_VOLUME + VOLUME_SUM2;
    POWER_SUM2 = 0;
    POWER_TOTAL = POWER_TOTAL + POWER_PROPS2;
end

% Determine Volume for EM Field Sensor Handheld Unit if Chosen in Analysis
if exist('MASS_PROPS3') == 1
    TOTAL_MASS = TOTAL_MASS + MASS_PROPS3(1,1);
    VOLUME_SUM3 = MASS_PROPS3(1,2) \* MASS_PROPS3(1,3) \* MASS_PROPS3(1,4);
    TOTAL_VOLUME = TOTAL_VOLUME + VOLUME_SUM3;
    POWER_SUM3 = 0;
    POWER_TOTAL = POWER_TOTAL + POWER_PROPS3;
end

% Payload Requirements Logic Check
% Logic Checks for MASS, VOLUME, POWER meet MISSION INPUT Constraints.
if TOTAL_MASS <= MASS_LIMIT
    MASS_FLAG = 'SUCCEEDED!';
    MASS_FLAG_STR = [{'Designed Mission Sensor Package Meets Mass Limit!' \n        'Minimum Sensor Requirements met allows for System Redundancy in Components.'}];
else
    MASS_FLAG = 'FAILED!';
    MASS_FLAG_STR = [{'Designed Mission Sensor Package EXCEEDS Mass Limits' \n        'Database Configuration for Required Sensors Exceeds Input Sensor Package Mass Limit' \n        'Reconfigure Sensor Package for Less Components or Select Multiple Data Type Sensors in Design.'}];
end

if TOTAL_VOLUME <= VOLUME_LIMIT
    VOLUME_FLAG = 'SUCCEEDED!';
    VOLUME_FLAG_STR = [{'Designed Mission Sensor Package Meets Volume Limit!' \n        'Minimum Sensor Requirements met allows for System Redundancy in Components.'}];
else
    VOLUME_FLAG = 'FAILED!';
VOLUME_FLAG_STR = {'Designed Mission Sensor Package EXCEEDS Mass Limits'
  Database Configuration for Required Sensors Exceeds Input Sensor Package Volume Limit
  'Reconfigure Sensor Package for Less Components or Select Multiple Data Type Sensors in Design.'};

if POWER_TOTAL <= POWER_LIMIT
  POWER_FLAG = 'SUCCEEDED!'
  POWER_FLAG_STR = {'Designed Mission Sensor Package Meets Power Limit!'
  'Minimum Sensor Requirements met allows for System Redundancy in Components.'};
else
  POWER_FLAG = 'FAILED!'
  POWER_FLAG_STR = {'Designed Mission Sensor Package EXCEEDS Power Limits'
  'Database Configuration for Required Sensors Exceeds Input Sensor Package Power Limit'
  'Total Power Limit Achievable by Selecting Sensor Component Duty Cycles with Mission Planning.'
  'Reconfigure Sensor Package for Less Components or Select Multiple Data Type Sensors in Design.'};
end

%**+******************%**********+********+*++****++******+******++***+*******+++****+*+*******+*w*****+***+****************+***•+*+*******
%********* PRINT SUMMARY SECTION TO SCREEN

fprintf('

^~ ******************************************************************************************* i ) -

*** IN-SITU SENSOR PAYLOAD OPTIMIZATION TOOL ***

*** PROGRAM INPUTS ***

User Input Filename: %s', InputFile);

Planetary Body Name: %s', BODY);

Unit System: %s', UNITS);

Print Summary Option: %s', PRINTFLG);

Payload Mass Limit: %10.3f %s', MassLimit, MassUnit);

Payload Volume Limit: %10.4f %s', VolumeLimit, VolUnit);

Payload Power Limit: %10.3f %s', PowerLimit, PowerUnit);

**********************************************************************************************************

SENSOR PACKAGES DESIGNED:

for i = 1:size(SENSOR_DATA,1)
  fprintf('%s', SENSOR_DATA{1,1});
  if strcmpi(SENSOR_DATA{i}, 'ATMOSPHERE') == 1
    fprintf('%s - Temperature'
    fprintf('%s - Wind Velocity'
    fprintf('%s - Humidity'
    fprintf('%s - Density'
  end
  if strcmpi(SENSOR_DATA{i}, 'OPTICS') == 1
    fprintf('%s - %s', SENSOR_DATA{i,2});
    for r = 3:size(SENSOR_DATA,2)
      fprintf('%s - %s', SENSOR_DATA{i,r});
    end
end
```matlab
end
if strcmpi(SENSOR_DATA{i}, 'RADIATION') == 1
    for r = 2:size(SENSOR_DATA, 2)
        if strcmpi(SENSOR_DATA{i, r}, 'N/A') == 0
            fprintf(' - %s', SENSOR_DATA{i, r});
        else
            break;
        end
    end
end

% Print Mass Volume Breakdown Summary
k = 1;
fprintf('


*** PAYLOAD SENSOR BREAKDOWN ***

<table>
<thead>
<tr>
<th>SENSOR</th>
<th>MASS</th>
<th>VOLUME</th>
<th>POWER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>%4s</td>
<td>%6s</td>
<td>%6s</td>
</tr>
<tr>
<td>MassUnit, VolUnit, PowerUnit</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

for i = 1:size(SENSOR_DATA, 1)
    if strcmpi(SENSOR_DATA{i, 1}, 'ATMOSPHERE') == 1
        fprintf('| %17s | | |
                 | - Temperature | %10.4f | %10.4f | %10.4f |
                 | - Pressure | %10.4f | %10.4f | %10.4f |
                 | - Wind Velocity Probe | %10.4f | %10.4f | %10.4f |
                 | - Handheld Unit | %10.4f | %10.4f | %10.4f |
    end
    if strcmpi(WV_SENS_TYPE, 'ANEMOMETER') == 1
        fprintf('| %17s | | |
                 | - Wind Velocity Probe | %10.4f | %10.4f | %10.4f |
                | - Density | %10.4f | %10.4f | %10.4f |
    end
    if strcmpi(WV_SENS_TYPE, 'DOPPLER') == 1
        fprintf('| %17s | | |
                 | - Wind Velocity Probe | %10.4f | %10.4f | %10.4f |
    end
    fprintf('| %17s | | |
             | - Humidity | %10.4f | %10.4f | %10.4f |
             | - Density | %10.4f | %10.4f | %10.4f |
    end
end
```

```matlab
k = k + 5;
if strcmpi(SENSOR_DATA{i,1},'ACCELERATION') == 1
    fprintf('
| %17s | %10.4f | %10.4f | %10.4f
|',SENSOR_DATA{i,1},MASS_PROPS(k,1),VOLUME_SUM(k),POWER_PROPS(k));
end

%-----------------------------

k = k + 1;
if strcmpi(SENSOR_DATA{i,1},'ACOUSTICS') == 1
    fprintf('
| %17s | %10.4f | %10.4f | %10.4f
|',SENSOR_DATA{i,1},MASS_PROPS3(k,1),VOLUME_SUM3,POWER_PROPS3);
end

%-----------------------------

k = k + 1;
if strcmpi(SENSOR_DATA{i,1},'EM FIELD') == 1
    fprintf('
| %17s | %10.4f | %10.4f | %10.4f
|',SENSOR_DATA{i,1},MASS_PROPS3(k,1),VOLUME_SUM3,POWER_PROPS3);
end

%-----------------------------

k = k + 1;
if strcmpi(SENSOR_DATA{i,1},'GAS ANALYSIS') == 1
    fprintf('
| %17s | %10.4f | %10.4f | %10.4f
|',SENSOR_DATA{i,1},MASS_PROPS(k,1),VOLUME_SUM(k),POWER_PROPS(k));
end

%-----------------------------

k = k + 1;
if strcmpi(SENSOR_DATA{i,1},'INCLINATION') == 1
    fprintf('
| %17s | %10.4f | %10.4f | %10.4f
|',SENSOR_DATA{i,1},MASS_PROPS(k,1),VOLUME_SUM(k),POWER_PROPS(k));
end
```

```matlab
fprintf('k = k + 1;
end
if strcmpi(SENSOR_DATA{i,1}, 'NEPHELOMETRY') == 1
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ');
  fprintf(' | ')
end
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
fprintf('
| %17s | %10.4f | %10.4f | %10.4f
|-------------------------------|-----------------|-----------------|-----------------|
| %s | %10.4f | %10.4f | %10.4f
|-------------------------------|-----------------|-----------------|-----------------|

k = k + 1;
end

case = input('Enter case number: ','s');

% PRINT OUT CASE RESULTS FOR MASS VOLUME AND POWER LIMITS
fprintf('

*** CASE LIMITS SUMMARY ***');
fprintf('******a*************************************************************
 CASE MASS LIMIT: %s',MASS_FLAG);
for ln = 1:length(MASS_FLAG_STR)
fprintf('
					 %s',MASS_FLAG_STR{ln});
end
fprintf('
 CASE VOLUME LIMIT: %s',VOLUME_FLAG);
for ln = 1:length(VOLUME_FLAG_STR)
fprintf('
					 %s',VOLUME_FLAG_STR{In});
end
fprintf('
 CASE POWER LIMIT: %s',POWER_FLAG);
for ln = 1:length(POWER_FLAG_STR)
fprintf('
					 %s',POWER_FLAG_STR{In});
end

cd ..; % Return to Case Working Directory

for In = 1:length(MASS_FLAG_STR)
fprintf(PISSPO, '

** USER INPUTS **
 User Input Filename: %s', InputFile);
 fprintf(PISSPO, '
 Planetary Body Name: %s', BODY);
 fprintf(PISSPO, '
 Unit System: %s', UNITS);
 fprintf(PISSPO, '
 Print Summary Option: %s', PRINTFLG);
 fprintf(PISSPO, '

** CASE LIMITS **
 CASE MASS LIMIT: %10.3f %s', MASS_LIMIT, MassUnit);
 printf(PISSPO, '
 CASE VOLUME LIMIT: %10.4f %s', VOLUME_LIMIT, VolUnit);
 printf(PISSPO, '
 CASE POWER LIMIT: %10.3f %s', POWER_LIMIT, PowerUnit);

fprintf(FISSPO, '\n

*** PAYLOAD SENSOR BREAKDOWN ***');
fprintf(FISSPO, '\n| SENSOR | MASS | VOLUME |
| ==|==|==|
| MassUnit, VolUnit, PowerUnit);
<table>
<thead>
<tr>
<th>Sensor Type</th>
<th>Mass Props(k)</th>
<th>VOLUME SUM(k)</th>
<th>Power Props(k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>%10.4f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pressure</td>
<td>%10.4f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wind Velocity Probe</td>
<td>%10.4f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Handheld Unit</td>
<td>%10.4f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Humidity</td>
<td>%10.4f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Density</td>
<td>%10.4f</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```c
fprintf(FISSPO, 'n\n', SENSOR_DATA{k,1});
fprintf(FISSPO, 'n\n', MASS_PROPS(k,1), VOLUME_SUM(k), POWER_PROPS(k));
if strcmpi(WV_SENS_TYPE, 'ANEMOMETER') == 1
    fprintf(FISSPO, 'n\n', MASS_PROPS(k+1,1), VOLUME_SUM(k+1), POWER_PROPS(k+1));
    fprintf(FISSPO, 'n\n', MASS_PROPS(k+2,1), VOLUME_SUM(k+2), POWER_PROPS(k+2));
end
if strcmpi(WV_SENS_TYPE, 'DOPPLER') == 1
    fprintf(FISSPO, 'n\n', 2*MASS_PROPS(k+2,1), VOLUME_SUM(k+2), POWER_PROPS(k+2));
end
fprintf(FISSPO, 'n\n', MASS_PROPS(k+3,1), VOLUME_SUM(k+3), POWER_PROPS(k+3));
fprintf(FISSPO, 'n\n', MASS_PROPS(k+4,1), VOLUME_SUM(k+4), POWER_PROPS(k+4));
```

```c
if strcmpi(SENSOR_DATA{i,1}, 'ACCELERATION') == 1
    fprintf(FISSPO, 'n\n', SENSOR_DATA{i,1}, MASS_PROPS(k,1), VOLUME_SUM(k), POWER_PROPS(k));
end
```

```c
if strcmpi(SENSOR_DATA{i,1}, 'ACOUSTICS') == 1
    fprintf(FISSPO, 'n\n', SENSOR_DATA{i,1}, MASS_PROPS(k,1), VOLUME_SUM(k), POWER_PROPS(k));
end
```

```c
if strcmpi(SENSOR_DATA{i,1}, 'EM FIELD') == 1
    fprintf(FISSPO, 'n\n', SENSOR_DATA{i,1}, MASS_PROPS(k,1), VOLUME_SUM(k), POWER_PROPS(k));
end
```

```c
```
fprintf(FISSPO, '%10.4f
', MASS_PROPS3(1), VOLUME_SUM3, POWER_PROPS3);
fprintf(FISSPO, '
| Handheld Unit
| %10.4f %10.4f |
|', MASS_PROPS3(1), VOLUME_SUM3, POWER_PROPS3);
fprintf(FISSPO, '
<p>| %10.4f |</p>
<table>
<thead>
<tr>
<th>%10.4f</th>
</tr>
</thead>
</table>
| k = k + 1;
end
if strcmpi(SENSOR_DATA{i,1}, 'GAS ANALYSIS') == 1
fprintf(FISSPO, '
| %17s | %10.4f %10.4f %10.4f %10.4f |
|', SENSOR_DATA{i,1}, MASS_PROPS(k,1), VOLUME_SUM(k), POWER_PROPS(k));
fprintf(FISSPO, '
|-----------------------|
| %10.4f
| %10.4f
| k = k + 1;
end
if strcmpi(SENSOR_DATA{i,1}, 'INCLINATION') == 1
fprintf(FISSPO, '
| %17s | %10.4f %10.4f %10.4f %10.4f |
|', SENSOR_DATA{i,1}, MASS_PROPS(k,1), VOLUME_SUM(k), POWER_PROPS(k));
fprintf(FISSPO, '
|-----------------------|
| %10.4f
| %10.4f
| k = k + 1;
end
if strcmpi(SENSOR_DATA{i,1}, 'NEPHELOMETRY') == 1
fprintf(FISSPO, '
| %17s | %10.4f %10.4f %10.4f %10.4f |
|', SENSOR_DATA{i,1}, MASS_PROPS(k,1), VOLUME_SUM(k), POWER_PROPS(k));
fprintf(FISSPO, '
|-----------------------|
| %10.4f
| %10.4f
| k = k + 1;
end
if strcmpi(SENSOR_DATA{i,1}, 'OPTICS') == 1
fprintf(FISSPO, '
| %17s | %10.4f %10.4f %10.4f %10.4f |
|', SENSOR_DATA{i,1}, MASS_PROPS(k,1), VOLUME_SUM(k), POWER_PROPS(k));
fprintf(FISSPO, '
|-----------------------|
| %10.4f
| %10.4f
| k = k + 1;
end
if strcmpi(SENSOR_DATA{i,1}, 'RADIATION') == 1
fprintf(FISSPO, '
| %17s | %10.4f %10.4f %10.4f %10.4f |
|', SENSOR_DATA{i,1}, MASS_PROPS(k,1), VOLUME_SUM(k), POWER_PROPS(k));
fprintf(FISSPO, '
|-----------------------|
| %10.4f
| %10.4f
| k = k + 1;
end
if strcmpi(SENSOR_DATA{i,1}, 'RADIATION') == 1
fprintf(FISSPO, '
| %17s | %10.4f %10.4f %10.4f %10.4f |
|', SENSOR_DATA{i,1}, MASS_PROPS(k,1), VOLUME_SUM(k), POWER_PROPS(k));
fprintf(FISSPO, '
|-----------------------|
| %10.4f
| %10.4f
| k = k + 1;
end
fprintf(FISSPO, '
');
fprintf(FISSPO, '\n| %17s | %10.4f | %10.4f | %10.4f
| ', SENSOR_DATA{i, 1}, MASS_PROPS(k, 1), VOLUME_SUM(k), POWER_PROPS(k));
fprintf(FISSPO, '\n| ');
k = k + 1;
end
if strcmpi(SENSOR_DATA{i, 1}, 'REFRACTION') == 1
fprintf(FISSPO, '
| %17s | %10.4f | %10.4f | %10.4f
| ', SENSOR_DATA{i, 1}, MASS_PROPS(k, 1), VOLUME_SUM(k), POWER_PROPS(k));
fprintf(FISSPO, '\n| ');
k = k + 1;
end
if strcmpi(SENSOR_DATA{i, 1}, 'DATA PROCESSING') == 1
fprintf(FISSPO, '
| %17s | %10.4f | %10.4f | %10.4f
| ', SENSOR_DATA{i, 1}, MASS_PROPS(k, 1), VOLUME_SUM(k), POWER_PROPS(k));
fprintf(FISSPO, '\n| ');
k = k + 1;
end
end
fprintf(FISSPO, '\n| SENSOR PAYLOAD TOTAL | %10.4f | %10.4f | %10.4f
| ', TOTAL_MASS, TOTAL_VOLUME, POWER_TOTAL);
fprintf(FISSPO, 'CASE POWER LIMIT: %s',POWER_FLAG);
for ln = 1:length(POWER_FLAG_STR)
    fprintf(FISSPO, '%s' , POWER_FLAG_STR{ln});
end

% Planetary Database File ID
FPDB = fopen('PlanetDB_Summary.txt','r');
% Prints Planetary Database Information to Summary File
PDB_Data = fread(FPDB);
fwrite(FISSPO,PDB_Data,'char');
fprintf (FISSPO, ' 


' ) ,•
for i = 1:size(SENSOR_DATA,1)
    switch SENSOR_DATA{i,1}
    case 'ATMOSPHERE'
        %Reads and copies TEMPERATURE SUMMARY to ISSPO Summary File
        FIDT = fopen('TEMPERATURE_SENSOR_Summary.txt','r');
        DataT = fread(FIDT);
        fwrite(FISSPO,DataT,'char');
        fprintf(FISSPO, ' 


');
        fclose(FIDT);
    case 'ACCELERATION'
        %Reads and copies PRESSURE SUMMARY to ISSPO Summary File
        FIDP = fopen('PRESSURE_SENSOR_Summary.txt','r');
        DataP = fread(FIDP);
        fwrite(FISSPO,DataP,'char');
        fprintf(FISSPO, ' 


');
        fclose(FIDP);
        %Reads and copies WIND VELOCITY SUMMARY to ISSPO Summary File
        FIDW = fopen('WIND_VELOCITY_SENSOR_Summary.txt','r');
        DataW = fread(FIDW);
        fwrite(FISSPO,DataW,'char');
        fprintf(FISSPO, ' 


');
        fclose(FIDW);
        %Reads and copies HUMIDITY SUMMARY to ISSPO Summary File
        FIDH = fopen('HUMIDITY_SENSOR_Summary.txt','r');
        DataH = fread(FIDH);
        fwrite(FISSPO,DataH,'char');
        fprintf(FISSPO, ' 


');
        fclose(FIDH);
        %Reads and copies DENSITY SUMMARY to ISSPO Summary File
        FIDD = fopen('DENSITY_SENSOR_Summary.txt','r');
        DataD = fread(FIDD);
        fwrite(FISSPO, DataD, 'char') ;
        fprintf(FISSPO, ' 


');
        fclose(FIDD);
    case 'ACCELERATION'
        break
    end
}
close(FPDB);
% Reads and copies ACCELERATION SUMMARY to ISSPO Summary File
FIDACC = fopen('ACCELERATION_SENSOR_Summary.txt', 'r');
DataACC = fread(FIDACC);
fwrite(FISSPO, DataACC, 'char');
fprintf(FISSPO, 'n


');
fclose(FIDACC);

case 'ACOUSTICS'
% Reads and copies ACOUSTICS SUMMARY to ISSPO Summary File
FIDACS = fopen('ACOUSTIC_SENSOR_Summary.txt', 'r');
DataACS = fread(FIDACS);
fwrite(FISSPO, DataACS, 'char');
fprintf(FISSPO, 'n


');
fclose(FIDACS);

case 'EM FIELD'
% Reads and copies EM FIELD SUMMARY to ISSPO Summary File
FIDACF = fopen('EM_FIELD_SENSOR_Summary.txt', 'r');
DataACF = fread(FIDACF);
fwrite(FISSPO, DataACF, 'char');
fprintf(FISSPO, 'n


');
fclose(FIDACF);

case 'GAS ANALYSIS'
% Reads and copies GCMS SUMMARY to ISSPO Summary File
FIDGC = fopen('GCMS_SENSOR_Summary.txt', 'r');
DataGC = fread(FIDGC);
fwrite(FISSPO, DataGC, 'char');
fprintf(FISSPO, 'n


');
fclose(FIDGC);

case 'INCLINATION'
% Reads and copies INCLINATION SUMMARY to ISSPO Summary File
FIDIN = fopen('INCLINOMETER_SENSOR_Summary.txt', 'r');
DataIN = fread(FIDIN);
fwrite(FISSPO, DataIN, 'char');
fprintf(FISSPO, 'n


');
fclose(FIDIN);

case 'NEPHELOMETRY'
% Reads and copies NEPHELOMETER SUMMARY to ISSPO Summary File
FIDNPH = fopen('NEPHELOMETER_SENSOR_Summary.txt', 'r');
DataNPH = fread(FIDNPH);
fwrite(FISSPO, DataNPH, 'char');
fprintf(FISSPO, 'n


');
fclose(FIDNPH);

case 'OPTICS'
% Reads and copies OPTICS SUMMARY to ISSPO Summary File
FIDIMG = fopen('IMAGING_SENSOR_Summary.txt', 'r');
DataIMG = fread(FIDIMG);
fwrite(FISSPO, DataIMG, 'char');
fprintf(FISSPO, 'n


');
fclose(FIDIMG);

case 'RADIATION'
% Reads and copies RADIATION SUMMARY to ISSPO Summary File
FIDRAD = fopen('RADIATION_SENSOR_Summary.txt', 'r');
DataRAD = fread(FIDRAD);
fwrite(FISSPO, DataRAD, 'char');
fprintf(FISSPO, 'n


');
fclose(FIDRAD);
fclose(FIDRAD);
case 'REFRACTION'
% Reads and copies REFRACTION SUMMARY to ISSPO Summary File
FIDREF = fopen('REFRACTION_SENSOR_Summary.txt', 'r');
DataREF = fread(FIDREF);
fwrite(FISSPO, DataREF, 'char');
fprintf(FISSPO, \n

\n

');
fclose(FIDREF);
case 'DATA PROCESSING'
% Reads and copies DIGITAL SIGNAL PROCESSING SUMMARY to ISSPO Summary File
FIDDSP = fopen('DIGITAL_SIGNAL_PROCESSING_Summary.txt', 'r');
DataDSP = fread(FIDDSP);
fwrite(FISSPO, DataDSP, 'char');
fclose(FISSPO);
otherwise
fprintf('\n\nSENSOR_DATA(%d) : %s
ERROR_PRG(52);
return;
end % Ends SWITCH Case Block
end % Ends For loop Block
fclose(FISSPO);

% Moves Summary File to Main Directory
copyfile('ISSPO_SUMMARY.txt', '../..');
cd ..; % Return to WORK_DIR
cd ..; % Return to Main Directory
fprintf('\n\n***********************************************************************');
fprintf('\n ISSPO DESIGN TOOL FINISHED!!1');
fprintf('\n***********************************************************************');

D2. Constants.
% function Constants(Units)
% ISSPO Tool Constants
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
% This file contains Constant data values for the ISSPO program.
% Data here is loaded into the main program to allow for use of common
% properties amongst all the subfunctions. The Values here are conversion
% factors or other needed values to be used by any section of the code.
% Values are assigned to a variable name and saved to a MatLAB '.mat' file
% and loaded when calling the ISSPO program.
function Constants(UNITS)

if strcmpi(UNITS, 'SI') == 1
    Guniv = 6.67E-11;
    GunivLbl = 'Nm^2/kg^2';
    m2m = 1;
    m2mLbl = 'm/m';
    cm2m = 10;
    cm2mLbl = 'cm/m';
    mm2m = 1000;
    mm2mLbl = 'mm/m';
    gr2kg = 1000;
    gr2kgLbl = 'gram/kg';
    Kg2kg = 1;
    Kg2kgLbl = 'kg/kg';
    cm3_to_m3 = 1000000;
    cm3_to_m3Lbl = 'cm^3/m^3';
    kgm3_to_gcm3 = 1000;
    kgm3_to_gcm3Lbl = 'kg/m^3 / g/cm^3';
    mA2A = 1000;
    mA2ALbl = 'mA/A';
    mW2W = 1000;
    mW2WLbl = 'mW/W';
    Press_Cnvt = 1000;
    Press_CnvtLbl = 'Pa to kPa';
    Temp_C2K = 273.15;
    Temp_C2KLbl = 'deg C to deg K';
    MassUnit = 'kg';
    VolUnit = 'm^3';
    PowerUnit = 'W';
    DistUnit = 'm';
    GravUnit = 'm/s^2';
    TempUnit = 'deg C';
    PressureUnit = 'Pa';
    DensityUnit = 'kg/m^3';
    VelUnit = 'm/s';
    GMUnit = 'km^3/s^2';
    SolIradUnit = 'W/m^2';
    MagFieldUnit = 'mGauss';
elseif strcmpi(UNITS, 'British') == 1
    Guniv = 3.32E-11;
    GunivLbl = 'lbf*ft^2/lb^2';
    ft2mi = 5280;
    ft2miLbl = 'ft/mi';
    in2fc = 12;
    in2fcLbl = 'in/ft';
    oz2lbm = 16;
    oz2lbmLbl = 'oz/lbm';
D3. Data Type Verifier

% function DataTypeVerifier(DataArray)
% ISSPC Tool Sensor Data Type Verifier

% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
% This file verifies the entered Sensor Data Types for the ISSPO program.
% The requested sensor Data Types are entered into a cell string array.
% This program verifies that the entered types correspond to a valid
% sensor type to eliminate typographical errors or that an invalid sensor
% type was asked for. The program only generates an error message if it
% finds an unknown data type and sends an error message to the user.
% function DataTypeERR_FLG = DataTypeVerifier(DataArray);

DataTypeERR_FLG = 0;
for i = 1:size(DataArray,1)
  switch DataArray{i,1}
    case 'ATMOSPHERE'
    case 'ACCELERATION'
    case 'ACOUSTICS'
    case 'EM FIELD'
    case 'GAS ANALYSIS'
    case 'INCLINATION'
    case 'NEPHELOMETRY'
    case 'OPTICS'
    case 'RADIATION'
    case 'REFRACTION'
    case 'DATA PROCESSING'
  otherwise
    fprintf('

SENSOR_DATA{%d,1}: %s
', i, DataArray{i,1} ) ;
    DataTypeERR_FLG = 1;
  end
% For Sensors Where additional specifications are required
if strcmpi(DataArray{i,1}, 'ATMOSPHERE') == 1
  if size(DataArray,2) > 1
    for pos = 2:2:size(DataArray,2)
      if strcmpi(DataArray{i,pos}, 'WIND VELOCITY') == 1
        if size(DataArray,2) >= pos+1
          if strcmpi(DataArray{i,pos+1}, 'ANEMOMETER') == 1 && strcmpi(DataArray{i,pos+1}, 'DOPPLER') == 1
            fprintf('

SENSOR_DATA{%d,%d}: %s
', i, pos+1, DataArray{i,pos+1} ) ;
            DataTypeERR_FLG = 15;
            return;
          end
        else
          DataTypeERR_FLG = 14;
          return;
        end
      end
    end
  end
  if strcmpi(DataArray{i,pos}, 'TEMPERATURE') == 1
    if size(DataArray,2) >= pos+1
      if strcmpi(DataArray{i,pos+1}, 'VOLTAGE') == 1 && strcmpi(DataArray{i,pos+1}, 'RESISTANCE') == 1
        fprintf('

SENSOR_DATA{%d,%d}: %s
', i, pos+1, DataArray{i,pos+1} ) ;
        DataTypeERR_FLG = 22;
        return;
      end
    else
      DataTypeERR_FLG = 21;
      return;
    end
  end
end
end
if strcmpi(DataArray{i,1},'GAS ANALYSIS') == 1
    if size(DataArray,2) == 1
        fprintf('SENSOR_DATA{%d,1}: %s
', i, DataArray{i,1} );
        DataTypeERR_FLG = 2;
        return;
    end
    if (strcmpi(DataArray{i,2}, 'WAVELENGTH') == 1) || (strcmpi(DataArray{i,2}, 'MASS-CHARGE') == 1)
        if strcmpi(DataArray{i,2}, 'MASS-CHARGE') == 1
            if size(DataArray,2) < 3
                DataTypeERR_FLG = 17;
                return;
            end
            if (strcmpi(DataArray{i,3}, 'LOW') == 1) || (strcmpi(DataArray{i,3}, 'MEDIUM') == 1) ||
                (strcmpi(DataArray{i,3}, 'HIGH') == 1)
                else
                    fprintf('SENSOR_DATA{%d,2}: %s
', i, DataArray{i,2} );
                    DataTypeERR_FLG = 16;
                    return;
                end
            else
                fprintf('SENSOR_DATA{%d,3}: %s
', i, DataArray{i,3} );
                DataTypeERR_FLG = 18;
                return;
            end
        else
            fprintf('SENSOR_DATA{%d,2}: %s
', i, DataArray{i,2} );
            DataTypeERR_FLG = 16;
            return;
        end
    end
else
    fprintf('SENSOR_DATA{%d,1}: %s
', i, DataArray{i,1} );
    DataTypeERR_FLG = 2;
    return;
end
end

end

end

end

end

end

end
DataTypeERR_FLG = 3;
return;
end
if size(DataArray,2) == 3
fprintf('n\n\nSENSOR_DATA{d,1}: %s', i, DataArray{i,3} );
DataTypeERR_FLG = 4;
return;
end
if (strcmpi(DataArray{i,2}, 'CAMERA') == 1) || (strcmpi(DataArray{i,2}, 'ARRAY') == 1) ||
else
fprintf('n\n\nSENSOR_DATA{d,2}: %s', i, DataArray{i,2} );
DataTypeERR_FLG = 6;
return;
end
if (strcmpi(DataArray{i,3}, 'LOW') == 1) || (strcmpi(DataArray{i,3}, 'MEDIUM') == 1) || (strcmpi(DataArray{i,3}, 'HIGH') == 1)
else
fprintf('n\n\nSENSOR_DATA{d,3}: %s', i, DataArray{i,3} );
DataTypeERR_FLG = 5;
return;
end
for r = 4:size(DataArray,2)
% Reads Data Array Variables Till end of Array or Encounters N/A
if strcmpi(DataArray{i,r}, 'N/A') == 0
if (strcmpi(DataArray{i,r}, 'X-RAY') == 1) || (strcmpi(DataArray{i,r}, 'VISUAL') == 1) ||
else
fprintf('n\n\nSENSOR_DATA{d,%d}: %s', i, r, DataArray{i,r} );
DataTypeERR_FLG = 7;
return;
else
break;
end
end
% Closes For loop
end% Closes OPTICS Sensor Definition Area
% Additional Information Required for RADIATION SENSORS Program
if strcmpi(DataArray{i,1}, 'RADIATION') == 1
if size(DataArray,2) == 1
fprintf('n\n\nSENSOR_DATA{d,1}: %s', i, DataArray{i,1} );
DataTypeERR_FLG = 8;
return;
end
for r = 2:size(DataArray,2)
% Reads Data Array Variables Till end of Array or Encounters N/A
if strcmpi(DataArray{i,r}, 'N/A') == 0
if (strcmpi(DataArray{i,r}, 'Charged Particle') == 1) || (strcmpi(DataArray{i,r}, 'Alpha') == 1) || (strcmpi(DataArray{i,r}, 'Beta') == 1) || (strcmpi(DataArray{i,r}, 'Gamma') == 1) || (strcmpi(DataArray{i,r}, 'X-Ray') == 1)
else
    fprintf('
SENSOR_DATA{%d,%d}: %s
', i, r, DataArray{i,r} );
    DataTypeERR_FLG = 9;
    return;
end

else % if N/A is Encountered in DataArray, Stops Reading in Values
    break;
end % Closes For loop
end % Closes RADIATION Sensor Definition Area

% Additional Information Required for INCLINOMETER SENSORS Program
if strcmpi(DataArray{i,1},'INCLINATION') == 1
    if size(DataArray,2) == 1
        fprintf('
SENSOR_DATA{%d,1}: %s
', i, DataArray{i,1} );
        DataTypeERR_FLG = 10;
        return;
    end
    if (strcmpi(DataArray{i,2}, 'LOW') == 1) || (strcmpi(DataArray{i,2}, 'MEDIUM') == 1) || (strcmpi(DataArray{i,2}, 'HIGH') == 1)
    else
        fprintf('
SENSOR_DATA{%d,2}: %s
', i, DataArray{i,2} );
        DataTypeERR_FLG = 11;
        return;
    end
end % Closes INCLINOMETER Sensor Definition Area

% Additional Information Required for ACCELEROMETER SENSORS Program
if strcmpi(DataArray{i,1}, 'ACCELERATION') == 1
    if size(DataArray,2) == 1
        fprintf('
SENSOR_DATA{%d,1}: %s
', i, DataArray{i,1} );
        DataTypeERR_FLG = 12;
        return;
    end
    if size(DataArray,2) < 3
        fprintf('
SENSOR_DATA{%d,1}: %s
', i, DataArray{i,1} );
        DataTypeERR_FLG = 13;
        return;
    end
    if (strcmpi(DataArray{i,2}, 'SOFT') == 1) || (strcmpi(DataArray{i,2}, 'MEDIUM') == 1) || (strcmpi(DataArray{i,2}, 'HIGH') == 1) || (strcmpi(DataArray{i,2}, 'IMPACT') == 1) || (strcmpi(DataArray{i,2}, 'BALLISTIC') == 1)
    else
        fprintf('
SENSOR_DATA{%d,2}: %s
', i, DataArray{i,2} );
    end
end % Closes ACCELEROMETER Sensor Definition Area
dataTypeErr_flg = 13;
return;
end
if (strcmpi(DataArray{i,3},'Voltage')==1) || (strcmpi(DataArray{i,3},'Const Current')==1) ||...
    (strcmpi(DataArray{i,3},'Self Generating')==1)
else
    fprintf('
\nSENSOR_DATA{%d,3}: \%s
\n', i, DataArray{i,3} );
    dataTypeErr_flg = 20;
    return;
end
end % Closes ACCELEROMETER Sensor Definition Area

% Additional Information Required for ACOUSTIC SENSORS Program
if strcmpi(DataArray{i,1},'ACOUSTICS') == 1
    if size(DataArray,2) == 1
        fprintf('
\nSENSOR_DATA{%d,1}: \%s
\n', i, DataArray{i,1} );
        dataTypeErr_flg = 23;
        return;
    end
    if (strcmpi(DataArray{i,2},'SENSOR')==1) || (strcmpi(DataArray{i,2},'ARRAY')==1) ||
        (strcmpi(DataArray{i,2},'VELOCITY')==1)
    else
        fprintf('
\nSENSOR_DATA{%d,2}: \%s
\n', i, DataArray{i,2} );
        dataTypeErr_flg = 24;
        return;
    end
end % Closes ACOUSTIC Sensor Definition Area

% Additional Information Required for Digital Signal Processing Program
if strcmpi(DataArray{i,1},'DATA PROCESSING') == 1
    if size(DataArray,2) == 1
        fprintf('
\nSENSOR_DATA{%d,1}: \%s
\n', i, DataArray{i,1} );
        dataTypeErr_flg = 27;
        return;
    end
    if size(DataArray,2) < 3
        fprintf('
\nSENSOR_DATA{%d,1}: \%s
\n', i, DataArray{i,1} );
        dataTypeErr_flg = 28;
        return;
    end
    if (strcmpi(DataArray{i,2},'LOW')==1) || (strcmpi(DataArray{i,2},'MEDIUM')==1) ||
        (strcmpi(DataArray{i,2},'HIGH')==1)
else
    fprintf('
\nSENSOR_DATA{%d,2}: \%s
\n', i, DataArray{i,2} );
    dataTypeErr_flg = 25;
    return;
end
if isnumeric(DataArray{1,3}) == 0
    fprintf(['\n\nSENSOR_DATA{%d,3}: %s
\n', i, DataArray{i,3} );
    DataTypeERR_FLG = 30;
    return;
end
end  % Closes DIGITAL SIGNAL PROCESSING Sensor Definition Area

% return;

D4. Planetary Database

% setup for program
% clc;
% clear all;
close all;
format compact;
% define program input variables
% Checks Input number of variable to see if Properly Formatted
if (nargin == 3)
    %fprintf('

 Success!!
');
    BODY = varargin{1};
    switch BODY
    case {'Sun', 'sun'};
        LOC = 1;
    case {'Mercury', 'mercury'};
        LOC = 2;
    case {'Venus', 'venus'};
        LOC = 3;
    case {'Earth', 'earth'};
        LOC = 4;
    case {'Moon', 'moon'};
        LOC = 5;
    case {'Mars', 'mars'};
        LOC = 6;
    case {'Jupiter', 'jupiter'};
        LOC = 7;
    case {'Io', 'io'};
        LOC = 8;
    case {'Europa', 'europa'};
        LOC = 9;
    case {'Ganymede', 'ganymede'};
        LOC = 10;
    case {'Callisto', 'callisto'};
        LOC = 11;
    case {'Titania', 'titania'};
        LOC = 12;
    case {'Adrastea', 'adrastea'};
        LOC = 13;
    case {'Amalthea', 'amalthea'};
        LOC = 14;
    case {'Thebe', 'thebe'};
        LOC = 15;
    case {'Saturn', 'saturn'};
        LOC = 16;
    case {'Mimas', 'mimas'};
        LOC = 17;
    case {'Enceladus', 'enceladus'};
        LOC = 18;
    case {'Tethys', 'tethys'};
        LOC = 19;
    case {'Dione', 'dione'};
        LOC = 20;
    case {'Rhea', 'rhea'};
        LOC = 21;
    case {'Titan', 'titan'};
        LOC = 22;
    case {'Hyperion', 'hyperion'};
        LOC = 23;
    case {'Iapetus', 'iapetus'};
        LOC = 24;
    case {'Uranus', 'uranus'};
        LOC = 25;
    case {'Neptune', 'neptune'};
        LOC = 26;
    case {'Pluto', 'pluto'};
        LOC = 27;
    otherwise
        fprintf('
ERROR: Unknown Planetary Object in Planetary Database Program from Preloaded Inputs!!
');
    end
end
UNITS = varargin{2};
Print = varargin{3};
end

% Checks for 0 or Three Input variables and if Three is the Print flag set
if ( (nargin == 3) && strcmpi(Print,'Y') == 1) || (nargin == 0 )
clc;
fprintf('*****************************************************************************');
fprintf('

 PLANETARY DATABASE PROGRAM ');fprintf('

');
elseif ( (nargin == 3) && strcmpi(Print, 'N') == 1)
% Does not Print Anything does Not exit
else
fprintf('

 ***ERROR: Invalid Number of Input Parameters')
fprintf('
 Terminating Program Operation . . . ');
return;
end

if (nargin == 3) && (strcmpi(Print, 'Y')) == 1
fprintf('

 *** Loading Input Program Values ***

');
end

if nargin == 0
fprintf('

*** Enter Program Values ***

');
end

while strcmpi(BODY, 'Default') == 1
SwitchCase = input('

Enter a Planetary Body Name to Analyze: ', 's');

switch SwitchCase
  case {'Sun', 'sun'};
    BODY = 'Sun'; LOC = 1;
  case {'Mercury', 'mercury'};
    BODY = 'Mercury'; LOC = 2;
  case {'Venus', 'venus'};
    BODY = 'Venus'; LOC = 3;
  case {'Earth', 'earth'};
    BODY = 'Earth'; LOC = 4;
  case {'Moon', 'moon'};
    BODY = 'Moon'; LOC = 5;
  case {'Mars', 'mars'};
    BODY = 'Mars'; LOC = 6;
  case {'Jupiter', 'jupiter'};
    BODY = 'Jupiter'; LOC = 7;
  case {'Io', 'io'};
    BODY = 'Io'; LOC = 8;
  case {'Europa', 'europa'};
    BODY = 'Europa'; LOC = 9;
  case {'Ganymede', 'ganymede'};
    BODY = 'Ganymede'; LOC = 10;
  case {'Callisto', 'callisto'};
    BODY = 'Callisto'; LOC = 11;
  case {'Metis', 'metis'};
    BODY = 'Metis'; LOC = 12;
  case {'Adrastea', 'adrastea'};
    BODY = 'Adrastea'; LOC = 13;
  case {'Amalthea', 'amalthea'};
    BODY = 'Amalthea'; LOC = 14;
  case {'Thebe', 'thebe'};
    BODY = 'Thebe'; LOC = 15;
  case {'Saturn', 'saturn'};
    BODY = 'Saturn'; LOC = 16;
  case {'Mimas', 'mimas'};
    BODY = 'Mimas'; LOC = 17;
  case {'Enceladus', 'enceladus'};
    BODY = 'Enceladus'; LOC = 18;
  case {'Tethys', 'tethys'};
    BODY = 'Tethys'; LOC = 19;
  case {'Dione', 'dione'};
    BODY = 'Dione'; LOC = 20;
  case {'Rhea', 'rhea'};
    BODY = 'Rhea'; LOC = 21;
  case {'Titan', 'titan'};
    BODY = 'Titan'; LOC = 22;
  case {'Hyperion', 'hyperion'};
    BODY = 'Hyperion'; LOC = 23;
  case {'Iapetus', 'iapetus'};
    BODY = 'Iapetus'; LOC = 24;
  case {'Uranus', 'uranus'};
    BODY = 'Uranus'; LOC = 25;
  case {'Neptune', 'neptune'};
    BODY = 'Neptune'; LOC = 26;
case {'Pluto', 'pluto'};
BODY = 'Pluto';
LOC = 27;
otherwise
    fprintf('\nERROR: Unknown Planetary Object in Planetary Database Program!!\n');
end
end  % Ends Body Declaration Loop

while strcmpi(UNITS,'Default') == 1
    UnitCase = input('\nEnter a Unit System to Use ( 'SI' or 'British' ): ', 's');
    switch UnitCase
    case {'SI', 'si'}
        UNITS = 'SI';
    case {'British', 'british'}
        UNITS = 'British';
    otherwise
        fprintf('\nERROR: Unknown Unit System!!\n');
    end
end  % Ends UNITS Declaration Loop

PrintCase = input('\nPrint Planetary Summary Results to Screen ( 'Y' or 'N' ) : ', 's');
switch PrintCase
    case {'Y', 'y'}
        Print = 'Y';
    case {'N', 'n'}
        N = 'N';
    otherwise
end  % Ends Print Declaration Loop
end  % Ends Input Section

% Commented out Verification of Inputs for Debugging
% BODY;
% LOC;
% UNITS;
% Print;

% Set Unit Labels
if strcmpi(UNITS,'SI') == 1
    MassUnit = 'kg';
    VolUnit = 'm^3';
    DistUnit = 'm';
    GravUnit = 'm/s^2';
    TempUnit = 'deg C';
    PressureUnit = 'Pa';
    DensityUnit = 'kg/m^2';
    VelUnit = 'm/s';
    GMUnit = 'km^3/s^2';
    SolRadUnit = 'km^2';
elseif strcmpi(UNITS,'British') == 1
    MassUnit = 'lbm';
    VolUnit = 'ft^3';
    DistUnit = 'mi';
    GravUnit = 'ft/s^2';
else
    MassUnit = '???';
    VolUnit = '???';
    DistUnit = '???';
    GravUnit = '???';
end
TerapUnit = 'deg F';
PressureUnit = 'psi';
DensityUnit = 'lbm/ft^3';
VelUnit = 'ft/s';
GMUnit = 'ft^3/s^2';
SolradUnit = 'BTUs/h*ft^2';

% Planetary Data Base Created Based on Unit System
% 2 - D Matrix Planet_DB(BODY_LOC, Property)
if strcmpi(UNITS, 'SI') == 1

% Units NAME kg 10*10 in*'3 ra m m/s deg C Pascals kg/m^3 N/A m/'s km N/A m^3/s A^2 deg K kg/m^3
MAG FIELD Mom-inertia J2 NATURAL

Planet_DB = {'SUN' 1.9891E+30 1.41200E+17 6.960E+08 6.960E+08 2.740E+02 5778.00 8.680E+01
1.40800E+03 5.0000E-05 6.1760E+05 1.3271E+11 0.000E+00 0.0000E+00 5.778E+03 0.0000E+00
3.000E-02 6.0000E+09 0.0000E+00 6.440E+01

% Mercury 3.3022E+23 6.08300E+09 2.4397E+06 2.4397E+06 3.700E+00 167.00 1.000E-10
5.42700E+03 0.0000E+00 4.3000E+03 2.2030E+04 1.196E+01 1.000E+01 0.0000E+00 0.0000E+00
3.000E-03 6.0000E+01 6.5180E+06 8.800E+06 167.00 1.000E-10

% Venus 4.8830E+24 9.28430E+10 6.0518E+06 6.0518E+06 8.870E+00 464.00 9.200E+06
5.24300E+03 0.0000E+00 1.0360E+04 3.2490E+05 7.500E-01 6.500E-01 -4.40 2.6139E+03 2.3170E+02
3.300E-01 4.45800E+00

% Earth 5.9742E+24 1.08321E+11 6.3781E+06 6.3781E+06 9.800E+00 15.00
1.0135E+01 5.5150E+03 3.3500E+03 1.1200E+04 3.9860E+06 1.050E+01 3.670E+01 -3.86 1.3676E+03 2.5430E+02
1.2255E+00 3.1000E+03 3.300E+01 1.08263E+03 1.02700E+02 0.0000E+00 0.0000E+00 3.5300E+03 0.0000E+00
0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00

% Mars 6.4185E+23 1.63180E+10 3.3970E+06 3.3970E+06 3.710E+00 -63.00
6.3860E+02 3.93300E+03 6.4800E+03 2.3800E+03 4.9000E+03 1.100E+01 1.200E+01 -0.21 1.3676E+03 2.7480E+02
3.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00

% Jupiter 1.8986E+27 1.43128E+14 7.1492E+07 7.1492E+07 2.4790E+08 5.8920E+03 2.1010E+02
5.24300E+03 0.0000E+00 1.0360E+04 3.2490E+05 7.500E-01 6.500E-01 -4.40 2.6139E+03 2.3170E+02
4.8830E+03 1.59200E+09 1.5608E+06 1.5608E+06 5.000E+00 3.8500E+01 1.2250E+00 310.00 3.300E-01

% Io 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00

% Europa 4.88000E+22 1.59200E+09 1.5608E+06 1.5608E+06 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00

% Callisto 1.0759E+23 5.8645E+09 2.4103E+06 2.4103E+06 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00
<table>
<thead>
<tr>
<th>Satellite</th>
<th>Major Axis (m)</th>
<th>Minor Axis (m)</th>
<th>Rotation Period (E.Days)</th>
<th>Inclination (deg)</th>
<th>Synodic Period (E.Days)</th>
<th>Opposition Angle (deg)</th>
<th>Opposition Velocity (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>'METIS'</td>
<td>1.0000E+17</td>
<td>3.35103E+04</td>
<td>2.0000E+04</td>
<td>2.0000E+04</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>'ADRASTEA'</td>
<td>2.0000E+16</td>
<td>4.35634E+02</td>
<td>1.3000E+04</td>
<td>8.0000E+03</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>'AMALTHEA'</td>
<td>7.5000E+18</td>
<td>2.68385E+05</td>
<td>1.3100E+04</td>
<td>6.7000E+04</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>'THEBE'</td>
<td>8.0000E+17</td>
<td>5.70199E+04</td>
<td>1.0400E+04</td>
<td>-139.15</td>
<td>1.000E+08</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>'SATURN'</td>
<td>5.6846E+26</td>
<td>8.27130E+13</td>
<td>6.0260E+07</td>
<td>5.4364E+07</td>
<td>3.5500E+04</td>
<td>3.7931E+07</td>
<td>3.4200E-01</td>
</tr>
<tr>
<td>'MIMAS'</td>
<td>3.7900E+19</td>
<td>2.0900E+05</td>
<td>1.1500E+03</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>'ENCELADUS'</td>
<td>1.0800E+20</td>
<td>2.5600E+05</td>
<td>1.6100E+03</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>'TETHYS'</td>
<td>6.1800E+20</td>
<td>5.2600E+05</td>
<td>9.6000E+02</td>
<td>8.0000E+01</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>'TITAN'</td>
<td>2.3100E+21</td>
<td>7.6400E+05</td>
<td>1.2300E+03</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>'HYPERION'</td>
<td>5.5000E+18</td>
<td>1.8500E+05</td>
<td>5.7000E+02</td>
<td>1.2000E+03</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>'IAPETUS'</td>
<td>1.8100E+21</td>
<td>7.1800E+05</td>
<td>5.0000E+02</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>'URANUS'</td>
<td>2.5559E+12</td>
<td>5.1000E+01</td>
<td>1.0000E+07</td>
<td>3.7100E+00</td>
<td>5.8200E+01</td>
<td>8.7000E+00</td>
<td>197.00</td>
</tr>
<tr>
<td>'NEPTUNE'</td>
<td>2.541E+07</td>
<td>8.0000E+01</td>
<td>1.0000E+07</td>
<td>1.5100E+00</td>
<td>4.6600E+01</td>
<td>8.1000E+01</td>
<td>120.00</td>
</tr>
<tr>
<td>'PLUTO'</td>
<td>5.8000E-01</td>
<td>6.0000E-01</td>
<td>1.0000E+00</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
</tr>
</tbody>
</table>

% Properties [ SEMIMAJOR-Axis SIDERALEAL-Period TROPICAL-Period PERIHELION APHELION SYNODIC-Per]
% SIDE-VEL ORB-INCLIN ECCENTRICITY LENGTH-DAY OBLIQUITY to Orbit]
% Unites m/s m deg E. Days m E. Days m E. Days m/s m/s
<table>
<thead>
<tr>
<th>Planet_DB</th>
<th>Mass</th>
<th>Volume</th>
<th>Radius-Eq</th>
<th>Radius-Pol</th>
<th>Gravity</th>
<th>Temp</th>
<th>Pressure</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun</td>
<td>4.38452 E+30</td>
<td>4.98643 E+18</td>
<td>2.2834 E+09</td>
<td>2.2834 E+09</td>
<td>9.012 E+02</td>
<td>9940.73</td>
<td>1.015 E+01</td>
<td></td>
</tr>
<tr>
<td>Mercuray</td>
<td>7.2801 E+23</td>
<td>2.1419 E+11</td>
<td>8.0045 E+06</td>
<td>8.0045 E+06</td>
<td>1.214 E+01</td>
<td>332.33</td>
<td>1.450 E+13</td>
<td></td>
</tr>
<tr>
<td>Venus</td>
<td>3.3895 E+02</td>
<td>1.4107 E+04</td>
<td>7.7798 E+14</td>
<td>1.190 E+01</td>
<td>1.060 E-01</td>
<td>2.8931 E+03</td>
<td>3.368 E+02</td>
<td></td>
</tr>
<tr>
<td>Earth</td>
<td>1.3176 E+25</td>
<td>3.8253 E+12</td>
<td>2.0926 E+07</td>
<td>2.0926 E+07</td>
<td>3.219 E+01</td>
<td>59.00</td>
<td>0.000 E+00</td>
<td></td>
</tr>
<tr>
<td>Io</td>
<td>3.4429 E+02</td>
<td>3.3500 E+03</td>
<td>3.6745 E+04</td>
<td>1.4075 E+06</td>
<td>3.670 E-01</td>
<td>-3.86</td>
<td>4.3352 E+02</td>
<td>-1.916 E+02</td>
</tr>
<tr>
<td>Moon</td>
<td>3.010 E+03</td>
<td>3.010 E+03</td>
<td>1.1282 E+03</td>
<td>1.1282 E+03</td>
<td>5.000 E-01</td>
<td>2.430 E+00</td>
<td>3.368 E+00</td>
<td></td>
</tr>
<tr>
<td>Mars</td>
<td>4.351 E+14</td>
<td>2.0913 E+05</td>
<td>7.8048 E+05</td>
<td>7.8048 E+05</td>
<td>1.100 E+01</td>
<td>1.200 E-01</td>
<td>2.430 E+00</td>
<td></td>
</tr>
<tr>
<td>Jupiter</td>
<td>9.224 E+01</td>
<td>2.4553 E+02</td>
<td>1.6502 E+03</td>
<td>1.6502 E+03</td>
<td>1.500 E-01</td>
<td>-1.52</td>
<td>1.8678 E+02</td>
<td>-8.149 E+01</td>
</tr>
<tr>
<td>Io</td>
<td>3.660 E+02</td>
<td>3.660 E+02</td>
<td>1.9604 E+05</td>
<td>1.9604 E+05</td>
<td>2.000 E-01</td>
<td>2.000 E-01</td>
<td>1.8678 E+02</td>
<td>-8.149 E+01</td>
</tr>
<tr>
<td>Europa</td>
<td>8.7899 E+01</td>
<td>2.0263 E+06</td>
<td>4.6866 E+21</td>
<td>0.000 E+00</td>
<td>0.000 E+00</td>
<td>0.000 E+00</td>
<td>0.000 E+00</td>
<td></td>
</tr>
<tr>
<td>Ganymede</td>
<td>3.3880 E+02</td>
<td>1.4107 E+04</td>
<td>7.7798 E+14</td>
<td>1.190 E+01</td>
<td>1.060 E-01</td>
<td>2.8931 E+03</td>
<td>3.368 E+02</td>
<td></td>
</tr>
<tr>
<td>Callisto</td>
<td>3.3880 E+02</td>
<td>1.4107 E+04</td>
<td>7.7798 E+14</td>
<td>1.190 E+01</td>
<td>1.060 E-01</td>
<td>2.8931 E+03</td>
<td>3.368 E+02</td>
<td></td>
</tr>
</tbody>
</table>

end

if strcmpi(UNITS,'British') == 1

% Properties
% OBJECT MASS VOLUME RADIUS-EQ RADIUS-Pol GRAVITY TEMP PRESSURE
% DENSITY ELLIPTICITY VELOCITY-Esc GM BOND-ALBEDO GEO-ALBEDO VIS-MAG SOL-Irradiance BB-TEMP ATM DENSITY
% MAG FIELD Mom-Inertia J2 NATURAL

% Units
% NAME lbm 10*10 ft*3 ft ft ft/s*2 deg F psi lbm/ft*3 ft/s ft*3/s*2 N/A N/A V(1,0) BTUs/h*ft*2 deg F lbm/ft*3

% Planet_DB

% SATELLITES

% Earth

% Io

% Europa

% Ganymede

% Callisto

% Metis
UnitLabelStr2 = { 'ft' 'E. Days' 'ft' 'E. Days' 'ft' 'E. Days' 'ft/s' 'ft/s' 'ft/s' 'deg' 'N/A' 'E. Hours' 'E. Hours' 'deg' };

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% PLANETARY ATMOSPHERE PROPERTIES DEFINITION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Form Planet_Atmosphere(BODY_LOC, : ) = { Element, %composition, Element, %composition, Element, %composition, ... } = 

% Order of Decreasing Composition
%
Planet_AtmosphereDB = { Element, %Concentration, Element, %Concentration, Element, %Concentration, ... } =

<table>
<thead>
<tr>
<th>Element</th>
<th>%Concentration</th>
</tr>
</thead>
<tbody>
<tr>
<td>'H'</td>
<td>'90.965'</td>
</tr>
<tr>
<td>'He'</td>
<td>'8.889'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'O2'</td>
<td>'22'</td>
</tr>
<tr>
<td>'Na'</td>
<td>'29'</td>
</tr>
<tr>
<td>'H2'</td>
<td>'10.2'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td>'N/A'</td>
<td>'0.0'</td>
</tr>
<tr>
<td></td>
<td>Conc</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>N/A</td>
<td>0.0</td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
</tr>
</tbody>
</table>

### Trace elements

<table>
<thead>
<tr>
<th>Planet</th>
<th>Atmosphere</th>
<th>Element</th>
<th>Conc</th>
<th>Conc Unit</th>
<th>Conc</th>
<th>Conc Unit</th>
<th>Conc</th>
<th>Conc Unit</th>
<th>Conc</th>
<th>Conc Unit</th>
<th>Conc</th>
<th>Conc Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>'N/A'</td>
<td>0.0</td>
<td>ppm</td>
<td>'N/A'</td>
<td>0.0</td>
<td>'N/A'</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
<td>ppm</td>
<td>'N/A'</td>
<td>0.0</td>
<td>'N/A'</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
<td>ppm</td>
<td>'N/A'</td>
<td>0.0</td>
<td>'N/A'</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
<td>ppm</td>
<td>'N/A'</td>
<td>0.0</td>
<td>'N/A'</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
<td>ppm</td>
<td>'N/A'</td>
<td>0.0</td>
<td>'N/A'</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
<td>ppm</td>
<td>'N/A'</td>
<td>0.0</td>
<td>'N/A'</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
<td>ppm</td>
<td>'N/A'</td>
<td>0.0</td>
<td>'N/A'</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
<td>ppm</td>
<td>'N/A'</td>
<td>0.0</td>
<td>'N/A'</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
<td>ppm</td>
<td>'N/A'</td>
<td>0.0</td>
<td>'N/A'</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'N/A'</td>
<td>0.0</td>
<td>ppm</td>
<td>'N/A'</td>
<td>0.0</td>
<td>'N/A'</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### References

- 'N2': 19.4
- 'CH4': 1.6
- 'N/A': 0.0
- 'N/A': 0.0
- 'N/A': 0.0
- 'N/A': 0.0
- 'N/A': 0.0
- 'N/A': 0.0
- 'N/A': 0.0
- 'N/A': 0.0

### Additional Information

- 'N/A': 0.0
- 'N/A': 0.0
- 'N/A': 0.0
- 'N/A': 0.0
- 'N/A': 0.0
- 'N/A': 0.0
- 'N/A': 0.0
- 'N/A': 0.0
- 'N/A': 0.0
- 'N/A': 0.0

**Notes:**

- 'Sun': 1.02
- 'ppm': 1.02
- 'ppm': 1.02
- 'ppm': 1.02
- 'ppm': 1.02
- 'ppm': 1.02
- 'ppm': 1.02
- 'ppm': 1.02
- 'ppm': 1.02
- 'ppm': 1.02

**Reference Elements:**

- 'Fe': 43
- 'S': 35
- 'C': 15
- 'O2': 33
- 'H2O': 11
- 'N/A': 0
- 'N/A': 0
- 'N/A': 0
- 'N/A': 0
- 'N/A': 0
% Assign Planet Atmosphere Properties
PlanAtm_Prop = Planet_AtmosphereDB{LOC, :};
PlanAtm_MEProp = Planet_AtmosphereDB2{LOC, :};

% Determine Planetary Density If Atmosphere is Defined if Unknown
if Planet_DB{LOC,18} == 0 && (strcmpi(PlanAtm_Prop(1), 'N/A') == 0)
    fprintf('

 *** WARNING: Planetary Atmospheric Density Undefined. ***

 Extrapolating Value From Ideal Gas Relation !

') ,-
% Loads Gas Constant Values Based on Unit System
if strcmpi(UNITS, 'SI') == 1
    switch PlanAtm_Prop{1}
        case 'Are'; R = 208.0; % J/kg*K
        case 'CO'; R = 297.0; % J/kg*K
        case 'CO2'; R = 188.9; % J/kg*K
        case 'He'; R = 2077.0; % J/kg*K
        case 'N2'; R = 2077.0; % J/kg*K
        case 'O2'; R = 2077.0; % J/kg*K
        case 'H20'; R = 2077.0; % J/kg*K
        otherwise
            fprintf('
*** WARNING: Undefined Gas Constant In Database for Atmospheric Component: %s ***', PlanAtm_Prop{1});
            R = input('Enter the corresponding Component Gas Constant to determine Planetary Surface Density: ');
    end
end
end

% Ends R Definition Switch Case

% Determine Atmospheric Density From Database Values P = Rho*R*T
Planet_DB{LOC,18} = Planet_DB{LOC,8}/(R * (Planet_DB{LOC,7}+Temp_F2R)) ;

end

% Ends 'SI' R Gas Constant Definition Block

if strcmpi(UNITS,'British') == 1
    switch PlanAtm_Prop{1}
    case 'Are'
        R = 0.0; % ft*lb/slug*R - not Defined
    case 'CO'
        R = 0.0; % ft*lb/slug*R - not Defined
    case 'CO2'
        R = 1131.0; % ft*lb/slug*R
    case 'He'
        R = 12420.0; % ft*lb/slug*R
    case 'N2'
        R = 24660.0; % ft*lb/slug*R
    case 'CH4'
        R = 3099.0; % ft*lb/slug*R
    case 'O2'
        R = 1775.0; % ft*lb/slug*R
    case 'SO2'
        R = 1554.0; % ft*lb/slug*R
    case 'H2O'
        R = 0.0; % ft*lb/slug*R - not Defined
    case 'N20'
        R = 2760.0; % ft*lb/slug*R
    otherwise
        fprintf('
** WARNING: Undefined Gas Constant In Database for Atmospheric Component: %s **', PlanAtm_Prop{1} ) ;
        R = input('Enter the corresponding Component Gas Constant to determine Planetary Surface Density: ');
    end
end

% Ends R Definition Switch Case

% Determine Atmospheric Density From Database Values P = Rho*R*T
Planet_DB{LOC,18} = Planet_DB{LOC,8}/(R * (Planet_DB{LOC,7}+Temp_F2R)) ;

end

% Ends 'SI' R Gas Constant Definition Block

% Ends Definition of Planetary Atmospheric Density Section

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%骛
fprintf('
 Planetary Equatorial Radius : \%14.5e \%s', Planet_DB(LOC,4), DistUnit );
fprintf('
 Planetary Polar Radius : \%14.5e \%s', Planet_DB(LOC,5), DistUnit );
fprintf('
 Planetary Gravity : \%14.5e \%s', Planet_DB(LOC,6), GravUnit );
fprintf('
 Planetary Density : \%14.5e \%s', Planet_DB(LOC,9), DensityUnit );
fprintf('
 Planetary Ellipticity : \%14.5e \%s', Planet_DB(LOC,10) );
fprintf('
 Planetary Escape Velocity : \%14.5e \%s', Planet_DB(LOC,11), VelUnit );
fprintf('
 Planetary GM : \%14.5e \%s', Planet_DB(LOC,12), GMUnit );
fprintf('
 Planetary Bond Albedo : \%14.5e \%s', Planet_DB(LOC,13) );
fprintf('
 Planetary Visual Geometric Albedo : \%14.5e \%s', Planet_DB(LOC,14), );
fprintf('
 Planetary Escape Velocity : \%14.5e \%s', Planet_DB(LOC,15), );
fprintf('
 Planetary Solar Irradiance : \%14.5e \%s', Planet_DB(LOC,16), SolIradUnit );
fprintf('
 Planetary Black Body Temperature : \%14.5e \%s', Planet_DB(LOC,17), TempUnit );
fprintf('
 Planetary Magnetic Field Strength : \%14.5e \%s', Planet_DB(LOC,19), UnitLabelStrl{19} );
fprintf('
 Planetary Moment of Inertia : \%14.5e \%s', Planet_DB(LOC,20) );
fprintf('
 Planetary J2 Parameter : \%14.5e \%s', Planet_DB(LOC,21) );
fprintf('
 Planetary Satellites : \%14d \%s', Planet_DB(LOC,22) );

fprintf(*** PLANETARY ORBITAL PARAMETERS ***);
fprintf(\n Semi-Major Axis : \%14.5e \%s', Planet_DB2(LOC,1), DistUnit );
fprintf(\n Sidereal Orbit Period : \%14.5e Earth Days', Planet_DB2(LOC,2) );
fprintf(\n Tropical Orbit Period : \%14.5e Earth Days', Planet_DB2(LOC,3) );
fprintf(\n Perihelion Distance : \%14.5e \%s', Planet_DB2(LOC,4), DistUnit );
fprintf(\n Aphelion Distance : \%14.5e \%s', Planet_DB2(LOC,5), DistUnit );
fprintf(\n Synodic Period : \%14.5e Earth Days', Planet_DB2(LOC,6) );
fprintf(\n Mean Orbital Velocity : \%14.5e \%s', Planet_DB2(LOC,7), VelUnit );
fprintf(\n Maximum Orbital Velocity : \%14.5e \%s', Planet_DB2(LOC,8), VelUnit );
fprintf(\n Minimum Orbital Velocity : \%14.5e \%s', Planet_DB2(LOC,9), VelUnit );
fprintf(\n Orbital Inclination : \%14.5e deg', Planet_DB2(LOC,10) );
fprintf(\n Orbital Eccentricity : \%14.5e ', Planet_DB2(LOC,11) );
fprintf(\n Orbital Eccentricity : \%14.5e ', Planet_DB2(LOC,12) );
fprintf(\n Length of Day : \%14.5e Earth Hours', Planet_DB2(LOC,13) );
fprintf(\n Obliquity to Orbit : \%14.5e deg', Planet_DB2(LOC,14) );

fprintf(*** ATMOSPHERIC PROPERTIES ***);
fprintf(\n Planetary Surface Temperature : \%14.2f \%s', Planet_DB(LOC,7), TempUnit );
fprintf(\n Planetary Surface Pressure : \%14.5e \%s', Planet_DB(LOC,8), PressureUnit );
fprintf(\n Planetary Surface Density : \%14.5e \%s', Planet_DB(LOC,18), DensityUnit );

fprintf(\n ATOMIC COMPOSITION ');
fprintf(\n Element % % Composition ');
for i = 1:2:length(PlanAtm_Prop{1})
    if strcmpi(PlanAtm_Prop{i}, 'N/A') == 1
        fprintf('\n \%s \%7.3f \%s', PlanAtm_Prop{i}, str2double(PlanAtm_Prop{i+1}) );
    end
end

fprintf(\n Trace Elements ');
fprintf(\n Element Amount Composition ');
for i = 1:3:length(PlanAtm_MEProp)-2
```matlab
if strcmpi( PlanAtm_MEProp{i}, 'N/A' )
    fprintf('
 %6s %9.3f %6s
', PlanAtm_MEProp{i}, str2double(PlanAtm_MEProp{i+i}), PlanAtm_MEProp{i+2} );
end
end
Planet_Properties = { Planet_DB{LOC,:} Planet_DB2{LOC,:} }; UnitLabelStrl = [ UnitLabelStrl UnitLabelStr2 ];
save('Planet_Data.mat', 'Planet_Properties', 'UnitLabelStr', 'PlanAtm_Prop', 'PlanAtm_MEPrcp' );

**************************************************************************

*** PLANETARY INPUTS ***

Planetary Body Name: %s', Planet_DB{LOC,1}); Unit System: %s', UNITS);


```
fprintf(Fout, 'Mean Orbital Velocity : ' ); %14.5e %s', Planet_DB2(LOC,7), VelUnit );
fprintf(Fout, 'Maximum Orbital Velocity : ' ); %14.5e %s', Planet_DB2(LOC,8), VelUnit );
fprintf(Fout, 'Minimum Orbital Velocity : ' ); %14.5e %s', Planet_DB2(LOC,9), VelUnit );
fprintf(Fout, 'Orbital Inclination : ' ); %14.5e deg', Planet_DB2(LOC,10) );
fprintf(Fout, 'Orbital Eccentricity : ' ); %14.5e ', Planet_DB2(LOC,11) );
fprintf(Fout, 'Sidereal Rotation Period : ' ); %14.5e Earth Hours', Planet_DB2(LOC,12) );
fprintf(Fout, 'Length of Day : ' ); %14.5e Earth Hours', Planet_DB2(LOC,13) );
fprintf(Fout, 'Obliquity to Orbit : ' ); %14.5e deg', Planet_DB2(LOC,14) );

% % *** ATMOSPHERIC PROPERTIES ***
fprintf(Fout, 'Planetary Surface Temperature :
1
Planetary Surface Pressure :
1
Planetary Surface Density
%I4.5e %s
Planet_DB2(LOC, 7), VelUnit );
%f14.5e %s', Planet_DB2|LOC,8j, VelUnit );
%f14.5e %s', Planet_DB2{L0C,9), VelUnit );
%f14.5e deg', Planet_DB2{LOC,10} );
%f14.5e ', Planet_DB2{LOC,11} );
%f14.5e Earth Hours', Planet_DB2{LOC,12} );
%f14.5e Earth Hours', Planet_DB2{LOC,13} );
%f14.5e deg', Planet_DB2{LOC,14} );

% % ATMOSPHERIC COMPOSITION
fprintf(Fout, '

Major Elements
');
fprintf(Fout, 'Element %% Composition
');
for i = 1:2:length(PlanAtm_Prop)-1
if strcmpi( PlanAtm_Prop{i}, 'N/A' ) ~= 1
    fprintf(Fout, '
 %6s %7.3f %%', PlanAtm_Prop{i}, str2double(PlanAtm_Prop{i + 1}) );
end
end

% % Trace Elements
fprintf(Fout, 'Element Amount
');
for i = 1:3:length(PlanAtm_MEProp)-2
if strcmpi( PlanAtm_MEProp{i}, 'N/A' ) ~= 1
    fprintf(Fout, '
 %6s %9.3f
', PlanAtm_MEProp{i+1}, PlanAtm_MEProp{i+2} );
end
end

fclose(Fout);

D5. AC Field Sensors
% function AC_FIELD_SENSORS(PLACFIELD,UNITS,Print)
% AC Field Sensor Database
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
% This file contains AC Field Sensor values for the ISSPO program.
% Data here is loaded into the main program to determine the proper sensor
% to use for Missions based on the sensor requirements.
% Values are assigned to a variable name and saved to a MATLAB '.mat' file
% and loaded when calling the ISSPO program.
% Some of the values are based on the Unit system chosen and are loaded
% in separate sections.

% References
% http://www.magneticsciences.com/MSI95Gaussmeter.html
% http://meda-sam.meda.com/catalog/cat4_1.htm
% http://www.ssec.honeywell.com/magnetic/magnetometers.html

function AC_FIELD_SENSORS(PLACFIELD,UNITS,Print)

% Load Constants values into Local Program
load('Constants_DB.mat');
load('Planet_Data.mat');

% Planet Properties
PLTEMP = Planet_Properties{7};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% AC Field Sensor Data Base Created Based on un Unit System
%% 2 - D Matrix AC_FIELD_DB(ACJType, Property)

% AC Field Sensor Common Properties
% Properties [ SENSOR TOLERANCE VOLTAGE TYPE ]
% Units [ mV/mG V N/A ]
% AC_FIELD_SENSOR_DB1 = { 'Bell-4080' 3 0.1 511 25 1000 5 2 0.40
% 1.00 9 'DC' ;
% 'MSI-95' 3 0.1 50000 25 3000 3 1 0.40
% 'Bell-4180' 3 0.1 599 30 2000 2 1 0.50
% 'Bell-4190' 3 0.1 1999 30 2000 1 1 0.50
% 'Bell-5170ULR' 3 0.1 1000 0 20000 2 0 0.25
% 'Bell-5170LR' 3 0.1 100 200000 0 200000 2 0 0.25
% 'Bell-5170MR' 3 1000 2000000 0 200000 2 0 0.25
% 'Bell-5170HR' 3 10000 20000000 0 200000 2 0 0.25
% 'TAM - 1' 3 0.0 1000 0 60 1 0 0
% 'TAM - 2' 3 0.0 1000 0 60 1 0 0
% 'HM2300' 3 0.1 2000 0 0 0 0.52 0 0.0065
% ACFieldUnitLabelStrl = { 'Sensor' 'N/A' 'mG' 'mG' 'Hz' 'Hz' '%' 'N/A' 'sec' 'mV/mG' 'V' 'N/A' };
if strcmpi(UNITS,'SI') == 1
% Properties
% sensor_temp | electronics_temp | sensor_dimensions (cm) | elect_dimensions (cm) | sens
% units       |               |                        |                      |
grams        | Sensor temp  | Low deg C High | Low deg F High | L  | W  | H  | L  | W  | H  | grams
ACFIELD_SENSOR_DB2 = { 0 0 0 0 0 6.096 11.938 2.540 6.096 11.938 2.540 141.748
0 0 0 0 0 5.000 3.700 2.700 8.900 17.100 3.600 0
0 0 -10.0 50.0 -10.0 50.0 12.00 7.600 3.700 12.00 7.600 3.700 177.00
0 0 -10.0 50.0 -10.0 50.0 12.00 7.600 3.700 12.00 7.600 3.700 177.00
0 0 0 0 -25.0 70.0 14.047 1.811 1.811 17.526 9.906 3.658 0
0 0 0 0 -25.0 70.0 15.494 0.889 0.889 17.526 9.906 3.658 0
0 0 0 0 -25.0 70.0 15.494 0.889 0.889 17.526 9.906 3.658 0
0 0 -30.0 80.0 -30.0 71.0 4.76 6.68 11.9 5.23 14.02 16.38 310.0
0 0 -30.0 85.0 -30.0 85.0 10.67 3.81 2.23 0 0 0 98.0
ACFIELD_UNIT_LABEL2 = ['deg C' 'deg C' 'deg C' 'deg C' 'cm' 'cm' 'cm' 'cm' 'gram' 'gram']
end

if strcmpi(UNITS,'BRITISH') == 1
% Properties
% sensor_temp | electronics_temp | sensor_dimensions (in) | elect_dimensions (in) | sens
% oz          |               |                        |                      |
oz           | Sensor temp  | Low deg F High | Low deg C High | L  | W  | H  | L  | W  | H  | oz
ACFIELD_SENSOR_DB2 = { 0 0 0 0 0 2.40 4.70 1.00 2.40 4.70 1.00 5.00
0 0 0 0 0 2.00 1.50 1.00 3.50 6.75 1.40 0
0 0 14.0 122.0 14.0 122.0 4.70 3.00 1.75 4.70 3.00 1.75 6.24
0 0 14.0 122.0 14.0 122.0 4.70 3.00 1.75 4.70 3.00 1.75 6.24
0 0 32.0 122.0 32.0 158.0 5.53 0.71 0.71 6.90 3.90 1.44 0
0 0 32.0 122.0 32.0 158.0 6.10 0.35 0.35 6.90 3.90 1.44 0
0 0 32.0 122.0 32.0 158.0 6.10 0.35 0.35 6.90 3.90 1.44 0
0 0 32.0 122.0 32.0 158.0 6.10 0.35 0.35 6.90 3.90 1.44 0
0 0 32.0 122.0 32.0 158.0 6.10 0.35 0.35 6.90 3.90 1.44 0
0 0 32.0 122.0 32.0 158.0 6.10 0.35 0.35 6.90 3.90 1.44 0
ACFieldUnitLabelStr2 = ['deg F' 'deg F' 'deg F' 'deg F' 'in' 'in' 'in' 'in' 'oz' 'oz'];
end

% Comments on sensors - uses
AC_FIELD_SENSOR_DB3{1,1} = ' '; 
AC_FIELD_SENSOR_DB3{2,1} = ' '; 
AC_FIELD_SENSOR_DB3{3,1} = ' '; 
AC_FIELD_SENSOR_DB3{4,1} = ' '; 
AC_FIELD_SENSOR_DB3{5,1} = ' '; 
AC_FIELD_SENSOR_DB3{6,1} = ' '; 
AC_FIELD_SENSOR_DB3{7,1} = ' '; 
AC_FIELD_SENSOR_DB3{8,1} = ' '; 
AC_FIELD_SENSOR_DB3{9,1} = 'TAM-1 series parts meet NASA and military programs (MIL-STD-975). Fully Space Qualified'; 
AC_FIELD_SENSOR_DB3{10,1} = 'TAM-2 series parts meet NASA and military programs (MIL-STD-975). Fully Space Qualified. Single Analog Output per axis'; 
AC_FIELD_SENSOR_DB3{11,1} = 'Single Package design Sensor & Electronics self contained on PCB';

% Combine Data Arrays to a Single Matrix
AC_FIELD_SENSOR_DB = cat(2,AC_FIELD_SENSOR_DB1,AC_FIELD_SENSOR_DB2,AC_FIELD_SENSOR_DB3);
ACFieldUnitLabelStr = cat(2,ACFieldUnitLabelStr1,ACFieldUnitLabelStr2);

% Print Sensor Alert Warning if Planetary Magnetic Field is Zero
if PLACFIELD == 0
    if strcmpi(Print,'Y'! == 1
        fprintf('

 *** WARNING: PLANETARY MAGNETIC FIELD NOT DEFINED!! ***');
        fprintf('
 Sensor Design Based on other properties!');
    end
end

% Build Up Database of Possible Magnetic Field Sensors Based on Planetary Sensor Range
for d = 1:size(AC_FIELD_SENSOR_DB,1)
    if (PLACFIELD >= AC_FIELD_SENSOR_DB{d,3}) && (PLACFIELD <= AC_FIELD_SENSOR_DB{d,4})
        for a = 1:size(AC_FIELD_SENSOR_DB,2)
            AC_MATRIX{c,a} = AC_FIELD_SENSOR_DB{d,a};
        end
        c = c+1;
    end
end

% Copies MATRIX to MATRIX 2 if all Sensors Eliminated based on Temperature Range
if exist('AC_MATRIX') == 0
AC_MATRIX = AC_FIELDSENSOR_DB;
end

if size(AC_MATRIX,1) > 1
% Down Select a Single Sensor if Multiple Options Exist

% Select Via Temperature Operational Range

% Build Up Database of Possible AC Field Sensors Based on Temperature Range
for t = 1:size(AC_MATRIX,1)
    if (PLTEMP >= AC_MATRIX{t,13}) && (PLTEMP <= AC_MATRIX{t,14})
        for a = 1:size(AC_MATRIX,2)
            AC_MATRIX2{c,a} = AC_MATRIX{t,a};
        end
        c = c+1;
    end
end

% Copies AC_MATRIX to AC_MATRIX2 if all Sensors Eliminated Based on Temperature Range
if exist('AC_MATRIX2') == 0
    AC_MATRIX2 = AC_MATRIX;
end

if size(AC_MATRIX2,1) > 1
% Account for Fluctuations in Planetary Magnetic Field - Adds 20% to Sensor Max Range
PLACFIELD2 = PLACFIELD*0.2*PLACFIELD;
for d = 1:size(AC_MATRIX2,1)
    if (PLACFIELD2 >= AC_MATRIX2{d,3}) && (PLACFIELD2 <= AC_MATRIX2{d,4})
        for a = 1:size(AC_MATRIX2,2)
            AC_MATRIX3{c,a} = AC_MATRIX2{d,a};
        end
        c = c+1;
    end
end

% Copies AC_MATRIX2 to AC_MATRIX3 if all Sensors Eliminated Based on Fluctuation Range
if exist('AC_MATRIX3') == 0
    AC_MATRIX3 = AC_MATRIX2;
end

% Run Case if More than One Sensor is Viable
if size(AC_MATRIX3,1) > 1
% Sort Sensors by Max Operational Temperature Range
for t = 1:size(AC_MATRIX3,1)
    AC_TEMP(t) = AC_MATRIX3{t,14} - AC_MATRIX3{t,13};
end
[TempR, AC_Loc ] = max(AC_TEMP);

% Build Sensor Matrix Based on the Temp Range
k = 1;
for c = 1:length(AC_TEMP)
    if AC_TEMP(c) == TempR
        AC_MATRIX4{c,k} = AC_MATRIX3{AC_Loc,c} + PLACFIELD2;
    end
end

end
for j = 1:size(AC_MATRIX3,2)
    AC_MATRIX4{k,j} = AC_MATRIX3{c,j};
end
k = k + 1;
end
end

% Copies AC_MATRIX3 to AC_MATRIX4 if all Sensors Eliminated based on Fluctuation Range
if exist('AC_MATRIX4') == 0
    AC_MATRIX4 = AC_MATRIX3;
end

% Run Case if More than One Sensor is Viable
if size(AC_MATRIX4, 1) > 1
    % Create Mass / Volume Scaling Factor Optimization
    for mv = 1:size(AC_MATRIX4,1)
        AC_SCALE_FAC(mv) = AC_MATRIX4{mv,17} * AC_MATRIX4{mv,18} * AC_MATRIX4{mv,19} * AC_MATRIX4{mv,23} + ...
                        AC_MATRIX4{mv,20} * AC_MATRIX4{mv,21} * AC_MATRIX4{mv,22} * AC_MATRIX4{mv,24};
        if (AC_MATRIX4{mv,17} == 0) || (AC_MATRIX4{mv,18} == 0) || (AC_MATRIX4{mv,19} == 0) || (AC_MATRIX4{mv,23} == 0)
            AC_SCALE_FAC(mv) = 9.99E99;
        elseif (AC_MATRIX4{mv,20} == 0) || (AC_MATRIX4{mv,21} == 0) || (AC_MATRIX4{mv,22} == 0) && (AC_MATRIX4{mv,23} ~= 0)
            AC_SCALE_FAC(mv) = 9.99E99;
        elseif (AC_MATRIX4{mv,20} ~= 0) || (AC_MATRIX4{mv,21} ~= 0) || (AC_MATRIX4{mv,22} ~= 0) && (AC_MATRIX4{mv,23} == 0)
            AC_SCALE_FAC(mv) = 9.99E99;
        else
            AC_SCALE_FAC(mv) = AC_SCALE_FAC(mv) - 0.01;
        end
    end
    [AC_MIN_SCL, INDEX] = min(AC_SCALE_FAC);

    % Build Sensor Matrix Based on Scale Factors
    k = 1;
    for c = 1:length(AC_SCALE_FAC)
        if AC_SCALE_FAC(c) == AC_MIN_SCL
            for j = 1:size(AC_MATRIX4,2)
                AC_MATRIX5{k,j} = AC_MATRIX4{c,j};
            end
            k = k + 1;
        end
    end
end

% Copies AC_MATRIX4 to AC_MATRIX5 if all Sensors Eliminated based on Dimensions Range
if exist('AC_MATRIX5') == 0
    AC_MATRIX5 = AC_MATRIX4;
end
else
    % Assign Null values to Array if No Sensor is viable
    for ac = 1:size(AC_FIELD_SENSOR_DB,2)
        AC_MATRIX5{1,ac} = 0.0;
    end
end

AC_MATRIX5{1,1} = 'N/A';
AC_MATRIX5{1,12} = 'N/A';
AC_MATRIX5{1,25} = 'N/A';
end

% Save AC Field Sensor Properties to Data File
save('AC_FIELD_SENSOR_Data.mat');

save('AC_FIELDSENSOR_Final.mat', 'AC_MATRIX5', 'ACFieldUnitLabelStr');

% *** PRINT SUMMARY Output Section for 'Print' = 'Y'
if strcmpi(Print, 'Y') == 1
    fprintf('

 *** AC FIELD SENSOR SUMMARY RESULTS
');
    fprintf('

*** INPUTS ***
');
    fprintf('
 Magnetic Field Strength: %9.3f %s ' , PLACFIELD, MagFieldUnit);
    fprintf('
 Unit System: %s', UNITS);


};

% AC Field Sensor Properties
fprintf('

** Sensor Properties
');
fprintf(' %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s
 %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s
');
fprintf('• AC Field Sensor Type: %s', AC_MATRIX5{1,1});
fprintf('• Sensor Axis Number: %d
');
fprintf('• AC Field Sensing Range - Low: %11.2f %s', AC_MATRIX5{1,2}, ACFieldUnitLabelStr{2});
fprintf('• High: %11.2f %s', AC_MATRIX5{1,3}, ACFieldUnitLabelStr{3});
fprintf('• Operating Frequency Range - Low: %11.2f %s', AC_MATRIX5{1,4}, ACFieldUnitLabelStr{4});
fprintf('• High: %11.2f %s', AC_MATRIX5{1,5}, ACFieldUnitLabelStr{5});
fprintf('• Sensor Accuracy: %11.4f %s', AC_MATRIX5{1,6}, ACFieldUnitLabelStr{6});
fprintf('• Number of Digit Accuracy: %d %s', AC_MATRIX5{1,7}, ACFieldUnitLabelStr{7});
fprintf('• Sampling Time Interval: %11.3f %s', AC_MATRIX5{1,8}, ACFieldUnitLabelStr{8});
fprintf('• Sensor Tolerance: %11.2f %s', AC_MATRIX5{1,9}, ACFieldUnitLabelStr{9});
fprintf('• Environmental Properties
');
fprintf(' %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s
 %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s
');
fprintf('• Sensor Temperature Range - Low: %7.3f %s', AC_MATRIX5{1,10}, ACFieldUnitLabelStr{10});
fprintf('• High: %7.3f %s', AC_MATRIX5{1,11}, ACFieldUnitLabelStr{11});
fprintf('• Electronics Temperature Range - Low: %7.3f %s', AC_MATRIX5{1,12}, ACFieldUnitLabelStr{12});
fprintf('• High: %7.3f %s', AC_MATRIX5{1,13}, ACFieldUnitLabelStr{13});
fprintf('• Physical Properties
');
fprintf(' %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s
 %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s %11s
');
fprintf('• Sensor Dimensions - Length: %9.3f %s', AC_MATRIX5{1,14}, ACFieldUnitLabelStr{14});
fprintf('• Width: %9.3f %s', AC_MATRIX5{1,15}, ACFieldUnitLabelStr{15});
fprintf('• Height: %9.3f %s', AC_MATRIX5{1,16}, ACFieldUnitLabelStr{16});
fprintf('• Electronics Dimensions - Length: %9.3f %s', AC_MATRIX5{1,17}, ACFieldUnitLabelStr{17});
fprintf('• Width: %9.3f %s', AC_MATRIX5{1,18}, ACFieldUnitLabelStr{18});
fprintf('Sensor Mass: %9.3f %s', AC_MATRIX5{1,22}, ACFieldUnitLabelStr{22});
fprintf('Dimensions: %9.3f %s', AC_MATRIX5{1,24}, ACFieldUnitLabelStr{24});

fprintf('Power Requirements');
fprintf('Input Voltage: %9.3f %s', AC_MATRIX5{1,11}, ACFieldUnitLabelStr{11});
fprintf('Voltage Type: %s %s', AC_MATRIX5{1,12}, ACFieldUnitLabelStr{12});
fprintf('Sensor Comments: %s', AC_MATRIX5{1,25});

end

%******************************************************************************
% Writes a text file summary of the Planetary Properties
Fout = fopen('AC_FIELD_SENSOR_Summary.txt','w+');
fprintf(Fout, '


Magnetic Field Strength: %9.3f %s ', PLACFIELD, MagFieldUnit);
fprintf(Fout, '
Unit System: %s', UNITS);
fprintf(Fout, '

* ***********************************************************************

AC Field Sensor Properties
fprintf(Fout, '

** Sensor Properties **');
fprintf(Fout, '
AC Field Sensor Type: %s', AC_MATRIX5{1,1});
fprintf(Fout, '
Sensor Axis Number: %ld %s', AC_MATRIX5{1,2}, AC_FIELD_UNIT{2});
fprintf(Fout, '
AC Field Sensing Range - Low: %11.2f %s ', AC_MATRIX5{1,3}, ACFieldUnitLabelStr{3});
fprintf(Fout, '
High: %11.2f %s', AC_MATRIX5{1,4}, ACFieldUnitLabelStr{4});
fprintf(Fout, '
Operating Frequency Range - Low: %11.2f %s', AC_MATRIX5{1,5}, ACFieldUnitLabelStr{5});
fprintf(Fout, '
High: %11.2f %s', AC_MATRIX5{1,6}, ACFieldUnitLabelStr{6});
fprintf(Fout, '
Sensor Accuracy: %11.2f %s', AC_MATRIX5{1,7}, ACFieldUnitLabelStr{7});
fprintf(Fout, '
Number of Digit Accuracy: %ld %s', AC_MATRIX5{1,8}, AC_FIELD_UNIT{8});
fprintf(Fout, '
Sampling Time Interval: %11.4f %s ', AC_MATRIX5{1,9}, ACFieldUnitLabelStr{9});
fprintf(Fout, '
Sensor Tolerance: %11.2f %s ', AC_MATRIX5{1,10}, ACFieldUnitLabelStr{10});

Environmental Properties
fprintf(Fout, '

Environmental Properties');
fprintf(Fout, '
Sensor Temperature Range - Low: %7.3f %s', AC_MATRIX5{1,13}, ACFieldUnitLabelStr{13});
fprintf(Fout, '
High: %7.3f %s', AC_MATRIX5{1,14}, ACFieldUnitLabelStr{14});
fprintf(Fout, '
Electronics Temperature Range - Low: %7.3f %s', AC_MATRIX5{1,15}, ACFieldUnitLabelStr{15});
fprintf(Fout, '
High: %7.3f %s', AC_MATRIX5{1,16}, ACFieldUnitLabelStr{16});

Physical Properties
fprintf(Fout, '

Physical Properties');
fprintf(Fout, '
Sensor Dimensions - Length: %9.3f %s', AC_MATRIX5{1,17}, ACFieldUnitLabelStr{17});
fprintf(Fout, '
Width: %9.3f %s', AC_MATRIX5{1,18}, ACFieldUnitLabelStr{18});
fprintf(Fout, '
Height: %9.3f %s', AC_MATRIX5{1,19}, ACFieldUnitLabelStr{19});
fprintf(Fout, '
Sensor Mass: %9.3f %s', AC_MATRIX5{1,20}, ACFieldUnitLabelStr{20});
fprintf(Fout, '
Electronics Dimensions - Length: %9.3f %s', AC_MATRIX5{1,21}, ACFieldUnitLabelStr{21});
fprintf(Fout, '
Width: %9.3f %s', AC_MATRIX5{1,22}, ACFieldUnitLabelStr{22});
D6. Accelerometer Sensors

function ACCELEROMETER_SENSORS(ACC_TYPE, POW_TYPE, UNITS, Print)

% Accelerometers Sensor Database
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date:
% Fall 2007
% This file contains Accelerometers Sensor values for the ISSPO program.
% Data here is loaded into the main program to determine the proper sensor
% to use for Missions based on the Planetary Atmosphere composition.
% Values are assigned to a variable name and saved to a MatLAB '.mat' file
% and loaded when calling the ISSPO program.
% Some of the values are based on the Unit system chosen and are loaded
% in separate sections.
% References
% http://www.sensotec.com/catpages.asp

% Load Constants values into Local Program
load('Constants_DB.mat');
load('Planet_Data.mat');

% Define Local Temperature Variable
PLTEMP = Planet_Properties{7};

% Define Program Variables
if strcmpi(ACC_TYPE, 'BALLISTIC') == 1
    ACC_RANGE = 10000;
    TOL_LOWER = 0;
end
if strcmpi(ACC_TYPE, 'IMPACT') == 1
    ACC_RANGE = 10000;
    TOL_LOWER = 1001;
end
if strcmpi(ACC_TYPE, 'HIGH') == 1
    ACC_RANGE = 1000;
    TOL_LOWER = 101;
end
if strcmpi(ACC_TYPE, 'MEDIUM') == 1
    ACC_RANGE = 100;
    TOL_LOWER = 11;
end
if strcmpi(ACC_TYPE, 'SOFT') == 1
    ACC_RANGE = 10;
    TOL_LOWER = 0;
end

% Common Accelerator Properties
% Properties
% RESONANCE FREQ. [SENSOR DYNAMIC RANGE SHOCK LIMITS VIB LIMITS]
% INPUT CURRENT RANGE BIAS VOLTAGE ELECT NOISE LINEARITY
% Hz [ Type Low g's High Type Low mA High V Type Low Hz High
% Hz  g's mV/g Low mA]

| Hz  | g's | Hz  | g's | Hz  | g's | Hz  | g's | Hz  | g's | Hz  | g's | Hz  | g's | Hz  | g's | Hz  | g's | Hz  | g's |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 5000.00 | 10000.0 | 10000.0 |
| 'MA11' | -80.0 | 80.0 | 100.0 | 5.00 | 2.00 | 10000.00 | 1.00 |
| 'MA15' | 50.0 | 50.0 | 100.0 | 5.00 | 2.00 | 10000.00 | 1.00 |
| 'MA21' | 80.0 | 80.0 | 100.0 | 1.20 | 1.20 | 9000.00 | 1.00 |
| 'MA311' | -50.0 | 50.0 | 100.0 | 5.00 | 2.00 | 10000.00 | 1.00 |
| 'MA331' | -50.0 | 50.0 | 100.0 | 5.00 | 2.00 | 10000.00 | 1.00 |
| 'MA42' | -80.0 | 80.0 | 100.0 | 5.00 | 2.00 | 15000.00 | 1.00 |
| 'MAQ13' | -800.0 | 800.0 | 20.0 | 5.00 | 0.00 | 10000.00 | 1.00 |
| 'MAQ14' | 50.0 | 50.0 | 100.0 | 5.00 | 1.00 | 10000.00 | 1.00 |
| 'MAQ36' | -2000.0 | 2000.0 | 5.00 | 1.00 | 30000.00 | 1.00 |
| 'MAQ41' | -100.0 | 100.0 | 100.0 | 5.00 | 0.00 | 10000.00 | 1.00 |

% Common Accelerator Properties
end
AC MATRIX5{l,1} = 'N/A';
AC MATRIX5{l,2} = 'N/A';
AC MATRIX5{l,23} = 'N/A';
end

% Save AC Field Sensor Properties to Data File
save('AC_FIELD_SENSOR_Data.mat');
save('AC_FIELD_SENSOR_Final.mat', 'AC MATRIX5', 'ACFieldUnitLabelStr');

% *** PRINT SUMMARY Output Section for 'Print' = 'Y'
if strcmpi(Print, 'Y') == 1
fprintf('

 *** AC FIELD SENSOR SUMMARY RESULTS

 *** INPUTS ***
Magnetic Field Strength: %9.3f %s', PLACFIELD, MagFieldUnit);
Unit System: %s', UNITS);

** Sensor Properties
Sensor Type: %s', AC_MATRIX5{l,1});
Sensor Axis Number: %d', AC MATRIX5{l,2}, ACFieldUnitLabelStr{2});
AC Field Sensing Range - Low: %11.2f %s', AC MATRIX5{l,3}, ACFieldUnitLabelStr{3});
AC Field Sensing Range - High: %11.2f %s', AC MATRIX5{l,4}, ACFieldUnitLabelStr{4});
Operating Frequency Range - Low: %11.2f %s', AC MATRIX5{l,5}, ACFieldUnitLabelStr{5});
Operating Frequency Range - High: %11.2f %s', AC MATRIX5{l,6}, ACFieldUnitLabelStr{6});
Sensor Accuracy: %11.2f %s', AC MATRIX5{l,7}, ACFieldUnitLabelStr{7});
Number of Digit Accuracy: %11.4f %s', AC MATRIX5{l,8}, ACFieldUnitLabelStr{8});
Sampling Time Interval: %11.2f %s', AC MATRIX5{l,9}, ACFieldUnitLabelStr{9});
Sensor Tolerance: %11.2f %s', AC_MATRIX5{l,10}, ACFieldUnitLabelStr{10});

% Environmental Properties
Sensor Temperature Range - Low: %7.3f %s', AC MATRIX5{l,11}, ACFieldUnitLabelStr{11});
Sensor Temperature Range - High: %7.3f %s', AC MATRIX5{l,12}, ACFieldUnitLabelStr{12});
Electronics Temperature Range - Low: %7.3f %s', AC_MATRIX5{l,13}, ACFieldUnitLabelStr{13});
Electronics Temperature Range - High: %7.3f %s', AC_MATRIX5{l,14}, ACFieldUnitLabelStr{14});

% Physical Properties
Sensor Dimensions - Length: %9.3f %s', AC MATRIX5{l,17}, ACFieldUnitLabelStr{17});
Sensor Dimensions - Width: %9.3f %s', AC MATRIX5{l,18}, ACFieldUnitLabelStr{18});
Sensor Dimensions - Height: %9.3f %s', AC MATRIX5{l,19}, ACFieldUnitLabelStr{19});
Electronics Dimensions - Length: %9.3f %s', AC_MATRIX5{l,20}, ACFieldUnitLabelStr{20});
Electronics Dimensions - Width: %9.3f %s', AC_MATRIX5{l,21}, ACFieldUnitLabelStr{21});
fprintf('Sensor Mass: %9.3f %s', AC_MATRIX5{1,22}, ACFieldUnitLabelStr{22});
fprintf('Power Requirements
Input Voltage: %9.3f %s', AC_MATRIX5{1,11}, ACFieldUnitLabelStr{11});
fprintf('Voltage Type: %s %s', AC_MATRIX5{1,12}, ACFieldUnitLabelStr{12});
fprintf('Sensor Comments:
%s', AC_MATRIX5{1,25});
end

% Environmental Properties
fprintf(Fout, 'Sensor Temperature Range - Low: %11.2f %s', AC_MATRIX5{1,13}, ACFieldUnitLabelStr{13});
fprintf(Fout, 'High: %11.2f %s', AC_MATRIX5{1,14}, ACFieldUnitLabelStr{14});
fprintf(Fout, 'Electronics Temperature Range - Low: %11.2f %s', AC_MATRIX5{1,15}, ACFieldUnitLabelStr{15});
fprintf(Fout, 'High: %11.2f %s', AC_MATRIX5{1,16}, ACFieldUnitLabelStr{16});

% Physical Properties
fprintf(Fout, 'Sensor Dimensions - Length: %9.3f %s', AC_MATRIX5{1,17}, ACFieldUnitLabelStr{17});
fprintf(Fout, 'Width: %9.3f %s', AC_MATRIX5{1,18}, ACFieldUnitLabelStr{18});
fprintf(Fout, 'Height: %9.3f %s', AC_MATRIX5{1,19}, ACFieldUnitLabelStr{19});
fprintf(Fout, 'Electronics Dimensions - Length: %9.3f %s', AC_MATRIX5{1,20}, ACFieldUnitLabelStr{20});
fprintf(Fout, 'Width: %9.3f %s', AC_MATRIX5{1,21}, ACFieldUnitLabelStr{21});
fprintf(Fout, 'Height: %9.3f %s', AC_MATRIX5{1,22}, ACFieldUnitLabelStr{22});
D6. Accelerometer Sensors

function ACCELEROMETER_SENSORS(ACC_TYPE, POW_TYPE, UNITS, Print)
% Accelerometers Sensor Database
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
% This file contains Accelerometers Sensor values for the ISSPO program.
% Data here is loaded into the main program to determine the proper sensor
% to use for Missions based on the Planetary Atmosphere composition.
% Values are assigned to a variable name and saved to a MatLAB ‘.mat’ file
% and loaded when calling the ISSPO program.
% Some of the values are based on the Unit system chosen and are loaded
% in separate sections.
% References
% http://www.sensotec.com/catpages.asp
%
function ACCELEROMETER_SENSORS(ACC_TYPE, POW_TYPE, UNITS, Print)

% Load Constants values into Local Program
load('Constants_DB.mat');
load('Planet_Data.mat');
% Define Local Temperature Variable
PLTEMP = Planet_Properties{7};
% Define Program Variables
if strcmpi(ACC_TYPE, 'BALLISTIC') == 1
    ACC_RANGE = 10000;
    TOL_LOWER = 0;
end
if strcmpi(ACC_TYPE, 'IMPACT') == 1
    ACC_RANGE = 10000;
    TOL_LOWER = 1001;
end
if strcmpi(ACC_TYPE, 'HIGH') == 1
    ACC_RANGE = 1000;
    TOL_LOWER = 101;
end
if strcmpi(ACC_TYPE, 'MEDIUM') == 1
    ACC_RANGE = 100;
    TOL_LOWER = 11;
end
if strcmpi(ACC_TYPE, 'SOFT') == 1
    ACC_RANGE = 10;
    TOL_LOWER = 0;
end
% Common Accelerator Properties
% Properties [ SENSOR DYNAMIC RANGE SENSITIVITY TRANSVERSE SENS FREQ RANGE ELEC NOISE ]
% Units [ Hz Type Low g's High g's mA High V mA Type mG's Hz Hz ]
% RESONANCE FREQ. SHOCK LIMITS VIB LIMITS INPUT CURRENT RANGE BIAS VOLTAGE
ACCELEROMETER_SENSOR_DB1 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB2 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB3 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB4 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB5 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB6 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB7 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB8 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB9 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB10 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB11 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB12 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB13 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB14 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB15 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB16 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB17 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB18 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB19 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00}
ACCELEROMETER_SENSOR_DB20 = {'JTF-GPA', -500.0, 500.0, 0.08, 5.00, 0.00, 2400.00, 1.00
%\% Accelerometer Sensor Data Base Created Based on un Unit System
%\% 2 - D Matrix ACCELEROMETER_SENSOR_DB(A_Type, Property)

if strcmp(UNITS, 'SI') == 1
  % Properties
  % Units
  ACCelerometer SENSOR_DB2 = {
    [\% PROB e TEMP RANGE WEI GHT DIMENSIONS PROBE - cm TEMP SENS POWER ]
    [\% Low deg C High \% gr \% Length Width Height g's/deg C W ]
    [21.1 93 1 129.3 3.556 3.556 3.556 0 0 151
      -56.7 137.8 150.3 6.706 2.489 2.489 0.261 0 0 21
      -56.7 137.8 170.1 5.283 2.896 2.896 0.261 0 0 3
      -23.3 98.9 141.7 5.190 2.896 2.896 0.261 0.640 0 4
      -12.2 137.8 110.6 4.140 1.702 1.702 0.261 0 0 5
      -56.7 121.1 8.5 1.397 1.397 1.397 0.261 0 0 6
      -17.8 82.2 39.7 2.540 2.540 2.540 0.261 0.168 0 7
      -17.8 82.2 39.7 2.540 2.540 2.540 0.261 0 0 8
      -17.8 82.2 39.7 2.540 2.540 2.540 0.261 0 0 9
      -23.3 137.8 28.3 4.089 1.840 1.840 0.261 0 0 10
      -56.7 248.9 28.3 2.388 1.500 1.500 0.261 0 0 11
      -56.7 248.9 34.0 2.007 1.905 1.905 0.261 0 0 12
      -23.3 137.8 107.7 5.004 2.489 2.489 0.261 0 0 13
      0.0 71.1 85.0 4.064 2.540 2.540 0 0 0 14
      -56.7 120.0 10.0 2.030 2.030 2.030 0 0 0 15
      -20.0 54.0 2300.0 8.900 11.100 2.600 0 0 0 16
      -54.0 121.1 0.06 0.635 0.635 0.635 0.261 0 0 17
      -269.0 260.0 27.0 2.920 1.588 1.588 0 0 0 18
      -18.0 66.0 1.0 1.143 1.016 0.513 0.261 0 0 21
      -55.0 92.0 71.0 1.350 3.350 2.718 0.003 0.480 0 22
  ];

AccUnitLabelStr2 = { 'deg C' 'deg C' 'gr' 'cm' 'cm' 'cm' 'g's/deg C' 'W' };
if strcmpi(UNITS, 'British') == 1
% Properties
% Units
ACCELEROMETER_SENSOR_DB2 = { 
   70.0 200.0 4.560 1.400 1.400 1.400 0.0 0 0
   -70.0 280.0 5.300 2.640 0.980 0.980 0.145 0 0
   -70.0 280.0 6.000 2.080 1.140 1.000 0.145 0 0
   -10.0 210.0 5.000 2.240 1.140 1.140 0.145 2.186 0
   -10.0 180.0 3.900 1.630 0.670 0.670 0.145 0 0
   -70.0 250.0 0.330 0.550 0.550 0.550 0.145 0 0
   -70.0 280.0 1.000 1.610 0.740 0.740 0.145 0 0
   -10.0 280.0 1.000 0.940 0.590 0.590 0.145 0 0
   -70.0 480.0 1.200 0.790 0.750 0.750 0.145 0 0
   -70.0 480.0 0.250 0.470 0.330 0.330 0.145 0 0
   -70.0 480.0 3.500 1.650 1.260 1.260 0.145 0 0
   -10.0 280.0 3.800 1.970 0.980 0.980 0.145 0 0
   32.0 160.0 3.000 1.600 1.000 1.000 0 0 0
   -65.0 250.0 0.350 0.800 0.600 0.600 0 0 0
   -4.0 129.0 80.000 3.500 4.400 1.000 0 0 0
   -65.0 250.0 0.002 0.250 0.250 0.250 40.000 0 0
   -14.0 500.0 0.950 1.150 0.625 0.625 0 0 0
   0.0 150.0 0.035 0.400 0.400 0.202 1.200 0 0
   -67.0 203.0 2.504 1.320 1.320 1.070 0.00017 1.640 0
   
% Additional Comments on Sensor
ACCELEROMETER_SENSOR_DB1 = '3 JTF Units mounted on Triaxial Mounting block Records 3 axis data simultaneously.';
ACCELEROMETER_SENSOR_DB2 = 'General Purpose.';
ACCELEROMETER_SENSOR_DB3 = 'General Purpose.';
ACCELEROMETER_SENSOR_DB4 = 'General Purpose. 4 - 20 mA Proportional to G.';
ACCELEROMETER_SENSOR_DB5 = 'General Purpose. Miniature Size, Low Profile. Sensitivity up to 1000mV/G special order.';
ACCELEROMETER_SENSOR_DB6 = 'Miniature Size, Low Profile. Sensitivity down to 100mV/G special order.';
ACCELEROMETER_SENSOR_DB7 = 'DC response. Amplified Voltage Output. Range +/-1, 2, 5, 10, 20, 50 g''s available';
ACCELEROMETER_SENSOR_DB8 = 'DC response. 4-20 mA output proportional to G. Built in signal conditioning. 12 mA @ 0 g sensitivity scaled at full range. Range +/-1, 2, 5, 10, 20, 50 g''s available';
ACCELEROMETER_SENSOR_DB9 = 'DC response. 15-75 mV Full Scale Output Range. Use with Strain Gage Amps. Range +/-1, 2, 5, 10, 20, 50 g''s available';
ACCELEROMETER_SENSOR_DB10 = 'High Frequency. Sensitivity up to 1000mV/G special order.';
ACCELEROMETER_SENSOR_DB11 = 'General Purpose. Small Size. Charge Output format requires Charge Amplifier sets Frequency Range. Sensitivity in pC/G at 10, 20, 30 special order.';
ACCELEROMETER_SENSOR_DB12 = 'General Purpose. Small Size. Charge Output format requires Charge Amplifier. Sensitivity in pC/G at 10, 30, 50 special order.';
ACCELEROMETER_SENSOR_DB{13,26} = 'Sub-Miniature Size. Charge Output format requires Charge Amplifier. Sensitivity in pC/G at 0.5, 5, 50 special order.';

ACCELEROMETER_SENSOR_DB{14,26} = 'Enhanced Output. Charge Output format requires Charge Amplifier. Sensitivity in pC/G at 0.5, 5, 50 special order.';

ACCELEROMETER_SENSOR_DB{15,26} = 'Temperature and Vibration Output. Sensitivity in mV/G at 10, 30, 100, 1000 special order. Temp range 0 - 200 deg F.';

ACCELEROMETER_SENSOR_DB{16,26} = 'Rugged, High Impact. Enhanced Sensitivity mV/G in 1000, 500, 100, 50, 10, 5 at 5, 10, 50, 100, 500, 1000 G's. 13.5-16.5 VDC Source VDC Output.';

ACCELEROMETER_SENSOR_DB{17,26} = 'Triaxial Lightweight Accelerometer. Low Noise, Wide Frequency, Rugged Design.';

ACCELEROMETER_SENSOR_DB{18,26} = 'Temp, Humidity, Pressure, & Tri-Axial Shock Data Logger. Specific Operating Properties each. 60 day Battery Operation. 180 day start delay. PC Software Interface.';

ACCELEROMETER_SENSOR_DB{19,26} = 'Single Axis Accelerometer. Similar Model to type flown on Mars Deep Space 2 Mission. Three Axis model available with different ratings each direction.';

ACCELEROMETER_SENSOR_DB{20,26} = 'Single Axis, wide temperature range piezoelectric accelerometer for cryogenic applications. Self generating power device. Flight Proven - Huygens Lander SSP.';


% Select Acceleration Sensor by Input Operation Range Value

```matlab
for a = 1:size(ACCELEROMETER_SENSOR_DB,1)
    if strcmpi(ACC_TYPE,'BALLISTIC') == 1
        if (ACCELEROMETER_SENSOR_DB{a,3} > ACC_RANGE)
            for s = 1:size(ACCELEROMETER_SENSOR_DB,2)
                ACC_MATRIX{c,s} = ACCELEROMETER_SENSOR_DB{a,s};
                c = c + 1;
            end
        end
    end
    if strcmpi(ACC_TYPE,'IMPACT') == 1
        if (ACCELEROMETER_SENSOR_DB{a,3} <= ACC_RANGE) && (ACCELEROMETER_SENSOR_DB{a,3} >= TOL_LOWER)
            for s = 1:size(ACCELEROMETER_SENSOR_DB,2)
                ACC_MATRIX{c,s} = ACCELEROMETER_SENSOR_DB{a,s};
                c = c + 1;
            end
        end
    end
    if strcmpi(ACC_TYPE,'HIGH') == 1
        if (ACCELEROMETER_SENSOR_DB{a,3} <= ACC_RANGE) && (ACCELEROMETER_SENSOR_DB{a,3} >= TOL_LOWER)
            for s = 1:size(ACCELEROMETER_SENSOR_DB,2)
                ACC_MATRIX{c,s} = ACCELEROMETER_SENSOR_DB{a,s};
                c = c + 1;
            end
        end
    end
    if strcmpi(ACC_TYPE,'MEDIUM') == 1
        if (ACCELEROMETER_SENSOR_DB{a,3} <= ACC_RANGE) && (ACCELEROMETER_SENSOR_DB{a,3} >= TOL_LOWER)
            for s = 1:size(ACCELEROMETER_SENSOR_DB,2)
                ACC_MATRIX{c,s} = ACCELEROMETER_SENSOR_DB{a,s};
                c = c + 1;
            end
        end
    end
```
end

c = c + 1;
end
end
if strcmpi(ACC_TYPE, 'SOFT') == 1
    if (ACCELEROMETER_SENSOR_DB{a,3} <= ACC_RANGE)
        for s = 1:size(ACCELEROMETER_SENSOR_DB,2)
            ACC_MATRIX{c,s} = ACCELEROMETER_SENSOR_DB{a,s};
        end
    end
end

c = c + 1;
end
end

tmp = 1;
for a = 1:size(ACC_MATRIX,1)
    if strcmpi (ACC_MATRIX{a, 12} , POWJTYPE) == 1
        for s = 1:size(ACC_MATRIX,2)
            ACC_MATRIX2 {tmp, s } = ACC__MATRIX{a, s } ;
        end
        tmp = tmp + 1 ;
    end
end
end

% Continue Selection Process From Viable Options - If Multiple Sensor Options Exist

if size(ACC_MATRIX2,1) > 1

% Select Via Temperature Operational Range

c = 1;
% Build Up Database of Possible Acceleration Sensors Based on Temperature Range
for t = 1:size(ACC_MATRIX2,1)
    if (PLTEMP >= ACC_MATRIX2{t,18}) && ( PLTEMP <= ACC_MATRIX2 {t, 19 })
        for a = 1:size(ACC_MATRIX2,2)
            ACC_MATRIX3{c,a} = ACC_MATRIX2{t,a};
        end
        c = c+1;
    end
end
end

% Copies ACC_MATRIX2 to ACC_MATRIX3 if all Sensors Eliminated based on Temperature Range
if exist('ACC_MATRIX3') == 0
    ACC_MATRIX3 = ACC_MATRIX2;
end
if size(ACC_MATRIX3,1) >1

% Down Select by Min Electrical Noise
for d = 1:size(ACC_MATRIX3,1)
    NOISE_ACC_DELTA(d) = ACC_MATRIX3{d,24};
end
% Find the Minimum Acceleration Electrical Noise Sensor
MIN_NOISE_ACC_DELTA = min(NOISE_ACC_DELTA);
For Multiple Sensors with the Same Range Reduce Sensor Matrix to Those
with the Least Range

```matlab
k = 1;
for c = 1:length(NOISE_ACC_DELTA)
    if NOISE_ACC_DELTA(c) == MIN_NOISE_ACC_DELTA
        for j = 1:size(ACC_MATRIX3,2)
            ACC_MATRIX4{k, j'} = ACC_MATRIX3{c, j};
        end
        k = k + 1;
    end
end
```

Copies ACC_MATRIX3 to ACC_MATRIX4 if all Sensors Eliminated based on Temperature Range
```
if exist('ACC_MATRIX4') == 0
    ACC_MATRIX4 = ACC_MATRIX3;
end
```

Down Select by Max Temperature Range
```
if size(ACC_MATRIX4,1) > 1
    for d = 1:size(ACC_MATRIX4,1)
        TEMP_ACC_DELTA(d) = ACC_MATRIX4{d,19} - ACC_MATRIX4{d,18};
    end
end
```

Find the Maximum Accelerometer Temperature Range Sensor
```
MAX_TEMP_ACC_DELTA = max(TEMP_ACC_DELTA);
```

For Multiple Sensors with the Same Range Reduce Sensor Matrix to Those
with the Greatest Range
```
k = 1;
for c = 1:length(TEMP_ACC_DELTA)
    if TEMP_ACC_DELTA(c) == MAX_TEMP_ACC_DELTA
        for j = 1:size(ACC_MATRIX4,2)
            ACC_MATRIX5{k, j} = ACC_MATRIX4{c, j};
        end
        k = k + 1;
    end
end
```

Copies ACC_MATRIX4 to ACC_MATRIX5 if all Sensors Eliminated based on Temperature Range
```
if exist('ACC_MATRIX5') == 0
    ACC_MATRIX5 = ACC_MATRIX4;
end
```

Create Mass / Volume Scaling Factor Optimization
```
if size(ACC_MATRIX5,1) > 1
    for mv = 1:size(ACC_MATRIX5,1)
```

```matlab
end
```
ACC_SCALE_FAC(mv) = ACC_MATRIX5{mv, 20} * ACC_MATRIX5{mv, 21} * ACC_MATRIX5{mv, 22} * ACC_MATRIX5{mv, 23};
if (ACC_MATRIX5{mv, 20} == 0) || (ACC_MATRIX5{mv, 21} == 0) || (ACC_MATRIX5{mv, 22} == 0) || (ACC_MATRIX5{mv, 23} == 0)
ACC_SCALE_FAC(mv) = 9.99E99;
end

[ ACC_MIN_SCL, INDEX ] = min(ACC_SCALE_FAC);

% Build Sensor Matrix Based on Scale Factors
k = 1;
for c = 1:length(ACC_SCALE_FAC)
    if ACC_SCALE_FAC(c) == ACC_MIN_SCL
        for j = 1:size(ACC_MATRIX4,2)
            ACC_MATRIX6{k,j} = ACC_MATRIX5{c, j};
        end
        k = k + 1;
    end
end

% Copies ACC_MATRIX5 to ACC_MATRIX6 if all Sensors Eliminated based on Mass Volume Scale Factor
if exist('ACC_MATRIX6') == 0
    ACC_MATRIX6 = ACC_MATRIX5;
end

% Copies ACC_MATRIX2 to ACC_MATRIX6 if all Sensors Eliminated based on Mass Volume Scale Factor
if exist('ACC_MATRIX6') == 0
    ACC_MATRIX6 = ACC_MATRIX2;
end

% Save Accelerometer Sensor Properties to Data File
save('ACCELEROMETER_SENSOR_Data.mat');
save('ACCELEROMETER_SENSOR_Final.mat','ACC_MATRIX6','AccUnitLabelStr');

% *** PRINT SUMMARY Output Section for 'Print' = 'Y'
if strcmpi(Print, 'Y') == 1
    fprintf('

 *** ACCELEROMETER SENSOR SUMMARY RESULTS ***

 *** INPUTS ***

 Mission Acceleration Profile: %s', char(ACC_TYPE));
 fprintf('
 Accelerometer Power Type: %s', char(POW_TYPE));
 fprintf('
 Unit System: %s', UNITS);

 % Accelerometer Sensor Properties
** Sensor Properties **

Accelerometer Sensor Type:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Low</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensing Range</td>
<td>10.2f</td>
<td>10.2f</td>
</tr>
<tr>
<td>Sensor Sensitivity</td>
<td>10.2f</td>
<td>10.2f</td>
</tr>
<tr>
<td>Transverse Sensitivity</td>
<td>10.2f</td>
<td>10.2f</td>
</tr>
<tr>
<td>Operating Frequency Range - Low</td>
<td>10.2f</td>
<td>10.2f</td>
</tr>
<tr>
<td>Sensor Linearity</td>
<td>10.2f</td>
<td>10.2f</td>
</tr>
<tr>
<td>Resonance Frequency</td>
<td>10.2f</td>
<td>10.2f</td>
</tr>
<tr>
<td>Shock Limits</td>
<td>10.2f</td>
<td>10.2f</td>
</tr>
<tr>
<td>Vibration Limits</td>
<td>10.2f</td>
<td>10.2f</td>
</tr>
</tbody>
</table>

Power Requirements

Input Type:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Low</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Range - Low</td>
<td>10.2f</td>
<td>10.2f</td>
</tr>
<tr>
<td>Bias Voltage</td>
<td>10.2f</td>
<td>10.2f</td>
</tr>
<tr>
<td>Bias Voltage Type</td>
<td>10.2f</td>
<td>10.2f</td>
</tr>
<tr>
<td>Electrical Noise</td>
<td>10.2f</td>
<td>10.2f</td>
</tr>
<tr>
<td>Electrical Power</td>
<td>10.2f</td>
<td>10.2f</td>
</tr>
</tbody>
</table>

Environmental Properties

Sensor Temperature Range - Low:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature Sensitivity</td>
<td>10.3f</td>
</tr>
</tbody>
</table>

Physical Properties

Sensor Mass:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor Dimensions - Length</td>
<td>9.3f</td>
</tr>
<tr>
<td>Width</td>
<td>9.3f</td>
</tr>
<tr>
<td>Height</td>
<td>9.3f</td>
</tr>
</tbody>
</table>

Sensor Comments:

\n
**************************************************************************

*** ACCELEROMETER SENSOR SUMMARY RESULTS ***

**************************************************************************

Wrote a text file summary of the Planetary Properties

Fout = fopen('ACCELEROMETER_SENSOR_Summary.txt', 'w+');
Accelerometer Sensor Properties

% Power Requirements

% Environmental Properties

% Physical Properties

% Sensor Comments

fclose(Fout);
D7. Acoustic Sensors

% function ACOUSTIC_SENSORS(ACS_TYPE, UNITS, Print)
% Acoustic Sensor Database
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
% This file contains Acoustic Sensor values for the ISSPO program.
% Data here is loaded into the main program to determine the proper sensor
to use for Missions based on the sensor requirements.
% Values are assigned to a variable name and saved to a Matlab '.mat' file
% and loaded when calling the ISSPO program.
% Some of the values are based on the Unit system chosen and are loaded
% in separate sections.

% References
% http://www.pcb.com/Linked_Documents/Vibration/Mics_1006.pdf

function ACOUSTIC_SENSORS(ACS_TYPE, UNITS, Print)

% Load Constants values into Local Program
load('Constants_DB.mat');
load('Planet_Data.mat');

PLTEMP = Planet_Properties(7);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% ACOUSTIC Sensor Data Base Created Based on Unit System
%% 2 - D Matrix ACOUSTIC_DB(ACS_TYPE, Property)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Acoustic Sensor Common Properties
% Properties | SENSOR TYPE | INHERENT NOISE | CLIPPING LIMIT | VOLTAGE SUPPLY | NOM CURRENT | DYNAMIC RANGE | FREQUENCY RANGE | SENSITIVITY @250 Hz | RESONANCE
% Units | | Low dB | High V | mA | Low Hz | High Hz | % RH | OPERATIONAL Hz |
% dB | dB | | | | | | | | |

ACOUSTICSENSOR_DB = {
    'B&K Type 4948', 55, 150, 5, 20000, -57, -3, 2, 24000
    'B&K Type 4935', 30, 140, 100, 5000, 45, -3, 3, 0
    'B&K Type 4938', 30, 172, 4, 70000, 56, -3, 3, 60000
};
<table>
<thead>
<tr>
<th>Sensor ID</th>
<th>28</th>
<th>165</th>
<th>165</th>
<th>4</th>
<th>80000</th>
<th>0</th>
<th>-2</th>
<th>2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>'PCB 377B01'</td>
<td></td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>'PCB 377B02'</td>
<td>15</td>
<td>146</td>
<td>3.15</td>
<td>20000</td>
<td>0</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>'PCB 377B10'</td>
<td>30</td>
<td>170</td>
<td>4</td>
<td>70000</td>
<td>0</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>'PCB 377B11'</td>
<td></td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>'PCB 377B12'</td>
<td>15</td>
<td>146</td>
<td>3.15</td>
<td>10000</td>
<td>0</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>'PCB 377B20'</td>
<td></td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>'PCB 377B40'</td>
<td>20</td>
<td>160</td>
<td>4</td>
<td>40000</td>
<td>0</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>'PCB 377B41'</td>
<td>15</td>
<td>146</td>
<td>3.15</td>
<td>20000</td>
<td>0</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>'PCB 377A42'</td>
<td>10</td>
<td>146</td>
<td>2.60</td>
<td>20000</td>
<td>0</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>'PCB 377A50'</td>
<td>40</td>
<td>178</td>
<td>6.50</td>
<td>140000</td>
<td>0</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>'PCB 377A51'</td>
<td></td>
<td>31</td>
<td>192</td>
<td>10.0</td>
<td>20000</td>
<td>0</td>
<td>-2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>'PCB 377A53'</td>
<td>10</td>
<td>146</td>
<td>2.60</td>
<td>80000</td>
<td>0</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>'PCB 377A54'</td>
<td></td>
<td>15</td>
<td>146</td>
<td>3.15</td>
<td>10000</td>
<td>0</td>
<td>-2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>'PCB 377A60'</td>
<td></td>
<td>15</td>
<td>146</td>
<td>3.15</td>
<td>10000</td>
<td>0</td>
<td>-2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>'CT-190N'</td>
<td>56.8</td>
<td>135</td>
<td>10</td>
<td>20000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>'Array Type 4935'</td>
<td></td>
<td>135</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>'PU-Mini'</td>
<td>0</td>
<td>125</td>
<td>1.00</td>
<td>20000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>'USP-Mini'</td>
<td>0</td>
<td>135</td>
<td>1.00</td>
<td>20000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>'R80-Alpha'</td>
<td>30</td>
<td>62</td>
<td>0</td>
<td>200000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

AcousticUnitLabelStr1 = ['N/A' 'dB' 'Hz' 'Hz' 'dB' 'dB' 'dB' 'Hz' 'dB' 'dB' 'dB' 'V' 'V' 'mA' 'dB/yr' 'deg C' 'RH' 'N/A']

if strcmp('UNITS', 'SI') == 1
    % Properties
    if strcmp('UNITS', 'SI') == 1
        % WEIGHT
        % Units
        % grams
        ACOUSTICSENSOR_DB2 = [-50.0 100.0 -30.0 70.0 0.0130 -0.007 20 20 2.5
                             -10.0 55.0 -25.0 70.0 0 0 7 7 65.0
                             -40.0 300.0 -30.0 70.0 0.0030 -0.003 7 7 10.5
                             0 0 0 0 0 0 0 0 0];
AcousticUnitLabelStr2 = { 'deg C' 'deg C' 'deg C' 'deg C' 'dB/deg C' 'dB/kPa' 'mm' 'mm' 'mm' 'gram' };
<table>
<thead>
<tr>
<th>AcousticUnitLabelStr2</th>
<th>deg F</th>
<th>deg F</th>
<th>deg F</th>
<th>deg F</th>
<th>dB/deg F</th>
<th>dB/psi</th>
<th>in</th>
<th>in</th>
<th>in</th>
<th>oz</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.35</td>
<td>32.0</td>
<td>140.0</td>
<td>32.0</td>
<td>140.0</td>
<td>0</td>
<td>1.00</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.13</td>
<td>14.0</td>
<td>313.0</td>
<td>-13.0</td>
<td>158.0</td>
<td>0</td>
<td>2.6</td>
<td>0.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.88</td>
<td>32.0</td>
<td>140.0</td>
<td>32.0</td>
<td>140.0</td>
<td>0</td>
<td>2.44</td>
<td>0.50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.35</td>
<td>-85.0</td>
<td>347.0</td>
<td>-85.0</td>
<td>347.0</td>
<td>0</td>
<td>0.75</td>
<td>0.75</td>
<td>0.84</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

%Comments on sensors - uses
ACOUSTICSENSOR_DB3{1,1} = 'Measurement of sound pressure on surfaces, Acoustic-fatigue testing of airplanes, Wind-tunnel measurements, Medium- to high-level measurements, Measurement in confined spaces';
ACOUSTICSENSOR_DB3{2,1} = 'Simultaneous recording of time signals in medium to large microphone arrays, for example, simulated pass-by measurements, Spatial Transformation of Sound Fields (STSF) measurements';
ACOUSTICSENSOR_DB3{3,1} = 'For measurements in confined spaces and small cavities. High level, high frequency measurements. Optimised for flush mounting';
ACOUSTICSENSOR_DB3{4,1} = 'Free-field Precision Condenser Microphone Cartridges. Exceptional performance in high humidity.';
ACOUSTICSENSOR_DB3{5,1} = 'Free-field Precision Condenser Microphone Cartridges. Exceptional performance in high humidity.';
ACOUSTICSENSOR_DB3{6,1} = 'Pressure Precision Condenser Microphone Cartridges. Exceptional performance in high humidity.';
ACOUSTICSENSOR_DB3{7,1} = 'Pressure Precision Condenser Microphone Cartridges. Exceptional performance in high humidity.';
ACOUSTICSENSOR_DB3{8,1} = 'Pressure Precision Condenser Microphone Cartridges. Exceptional performance in high humidity.';
ACOUSTICSENSOR_DB3{9,1} = 'Random Incidence Precision Condenser Microphone Cartridges. Exceptional performance in high humidity.';
ACOUSTICSENSOR_DB3{10,1} = 'Free-Field Externally Polarized Precision Condenser Microphone Cartridges. Exceptional performance in high humidity.';
ACOUSTICSENSOR_DB3{11,1} = 'Free-Field Externally Polarized Precision Condenser Microphone Cartridges. Exceptional performance in high humidity.';
ACOUSTICSENSOR_DB3{12,1} = 'Free-Field Externally Polarized Precision Condenser Microphone Cartridges. Exceptional performance in high humidity.';
ACOUSTICSENSOR_DB3{13,1} = 'Pressure Externally Polarized Precision Condenser Microphone Cartridges. Exceptional performance in high humidity.';
ACOUSTICSENSOR_DB3{14,1} = 'Pressure Externally Polarized Precision Condenser Microphone Cartridges. Exceptional performance in high humidity.';
ACOUSTICSENSOR_DB3{15,1} = 'Pressure Externally Polarized Precision Condenser Microphone Cartridges. Exceptional performance in high humidity.';
ACOUSTICSENSOR_DB3{16,1} = 'Pressure Externally Polarized Precision Condenser Microphone Cartridges. Exceptional performance in high humidity.';
ACOUSTICSENSOR_DB3{17,1} = 'Random Incidence Externally Polarized Precision Condenser Microphone Cartridges. Exceptional performance in high humidity.';
ACOUSTICSENSOR_DB3{18,1} = 'Pressure Sensor. Excellent for extreme conditions - temperature, pressure. Not suited for quiet sounds. Flight Proven - Huygens Lander.';
ACOUSTICSENSOR_DB3{19,1} = 'Microflown Acoustic Sensor. Measures both particle velocity and sound pressure. Designed for the use in arrays.';
ACOUSTICSENSOR_DB3{20,1} = 'Microphone Array Sensor. Simultaneous recording of time signals in medium to large microphone arrays. Mass, volume, power reqs per each sensor.';
ACOUSTICSENSOR_DB3{21,1} = 'Microflown Acoustic Sensor. Measures both particle velocity and sound pressure. Designed for the use in arrays. Mass, volume, power reqs per each sensor.';
ACOUSTICSENSOR_DB3{22,1} = 'General Purpose Acoustic Sensor. High Frequency sensor. Robust, Reliable Sensor.';

% Combine Data Arrays to a Single Matrix
ACOUSTICSENSOR_DB = cat(2,ACOUSTICSENSOR_DB1,ACOUSTICSENSOR_DB2,ACOUSTICSENSOR_DB3);
AcousticUnitLabelStr = cat(2,AcousticUnitLabelStr1, AcousticUnitLabelStr2);

cmp = 1;
for a = 1:size(ACOUSTICSENSOR_DB,1)
    if strcmpi(ACOUSTICSENSOR_DB{a,18},ACS_TYPE) == 1
        for s = 1:size(ACOUSTICSENSOR_DB,2)
            ACS_MATRIX{tmp,s} = ACOUSTICSENSOR_DB{a,s};
        end
        tmp = tmp + 1;
    end
end

% Sort Sensors by Operational Temperature Range Based on Target Planetary Temperature
nc = 1;
for s = 1:size(ACS_MATRIX,1)
    if (PLTEMP >= ACS_MATRIX{s,19}) && (PLTEMP <= ACS_MATRIX{s,20})
        for s = 1:size(ACS_MATRIX,2)
            ACS_MATRIX2{c,s} = ACS_MATRIX{s,a};
        end
        c = c+1;
    end
end
if exist('ACS MATRIX2') == 0
fprintf('n
 *** WARNING ***

 NO VIABLE ACOUSTIC SENSORS EXISTS IN DATABASE BASED ON OPERATIONAL TEMPERATURE RANGE!

 Insulation methods must be used to insulate Acoustic Sensors.

ACS MATRIX2 = ACS MATRIX;
end

% Find the Max Operational Range
for d = 1:size(ACS MATRIX2,1)
    TEMPACS_DELTA(d) = ACS MATRIX2{d,20} - ACS MATRIX2{d,19};
end

% Find the Maximum Acoustic Temperature Range of the Sensor
MAX TEMPACS_DELTA = max(TEMPACS_DELTA);

% For Multiple Sensors with the Same Range Reduce Sensor Matrix to Those
% with the Greatest Range
k = 1;
for c = 1:length(TEMPACS_DELTA)
    if TEMPACS_DELTA(c) == MAX TEMPACS_DELTA
        for j = 1:size(ACS MATRIX2,2)
            ACS MATRIX3{k,j} = ACS MATRIX2{c,j};
        end
        k = k + 1;
    end
end

if size(ACS MATRIX3,1) > 1
% Create Volume Scaling Factor Optimization
for v = 1:size(ACS MATRIX2,1)
    ACS SCALE FAC(v) = ACS MATRIX3{v,25} * ACS MATRIX3{v,26} * ACS MATRIX3{v,27},
    if (ACS MATRIX3{v,25} == 0) || (ACS MATRIX3{v,26} == 0) || (ACS MATRIX3{v,27} == 0)
        ACS SCALE FAC(v) = 9.99E99;
    end
end

[ ACS MIN_SCL, INDEX ] = min(ACS SCALE FAC);
% Build Sensor Matrix Based on Scale Factors
k = 1;
for c = 1:length(ACS SCALE FAC)
    if ACS SCALE FAC(c) == ACS MIN_SCL
        for j = 1:size(ACS MATRIX3,2)
            ACS MATRIX4{k,j} = ACS MATRIX3{c,j};
        end
        k = k + 1;
    end
end
% Copies ACS_MATRIX3 to ACS_MATRIX4 if all Sensors Eliminated
if exist('ACS_MATRIX4') == 0
    ACS_MATRIX4 = ACS_MATRIX3;
end

% Save Acoustic Sensor Properties to Data File
save('ACOUSTIC_SENSOR_Data.mat');
save('ACOUSTIC_SENSOR_Final.mat', 'ACS_MATRIX4', 'AcousticUnitLabelStr');

% *** PRINT SUMMARY Output Section for 'Print' = 'Y'
if strcmpi(Print,'Y') == 1
    fprintf(' %n

*** ACOUSTIC SENSOR SUMMARY RESULTS ***

*** INPUTS ***

Unit System: %s

************************************************************************

Sensor Properties

Acoustic Sensor Type:
Dynamic Sensing Range - Low: %10.2f %s
High: %10.2f %s

Operating Frequency Range - Low: %10.2f %s
High: %10.2f %s

Sensitivity at 250 Hz Tolerance:
Low: %10.2f %s
High: %10.2f %s

Resonance Frequency:
Low: %10.2f %s
High: %10.2f %s

Inherent Noise:
Low: %10.2f %s
High: %10.2f %s

Clipping Limit:
Low: %10.2f %s
High: %10.2f %s

Stability Conditions - Time:
Low: %10.2f %s
High: %10.2f %s

Temperature:
Low: %10.2f %s
High: %10.2f %s

Relative Humidity:
Low: %10.2f %s
High: %10.2f %s

Power Requirements

Voltage Range - Min: %10.2f %s
Max: %10.2f %s
Nominal Current: %10.2f %s

Environmental Properties

Operational Temperature Range - Low: %10.2f %s
High: %10.2f %s
Storage Temperature Range - Low: %10.2f %s

% Acoustic Sensor Properties
fprintf('n\n\n Acoustic Sensor Type: %10.2f %s', ACS_MATRIX4{1,1});
fprintf('n\n Dynamic Sensing Range - Low: %10.2f %s', ACS_MATRIX4{1,2}, AcousticUnitLabelStr{2});
fprintf('n\n High: %10.2f %s', ACS_MATRIX4{1,3}, AcousticUnitLabelStr{2});
fprintf('n\n Operating Frequency Range - Low: %10.2f %s', ACS_MATRIX4{1,4}, AcousticUnitLabelStr{2});
fprintf('n\n High: %10.2f %s', ACS_MATRIX4{1,5}, AcousticUnitLabelStr{2});
fprintf('n\n Sensitivity at 250 Hz - Range: %10.2f %s', ACS_MATRIX4{1,6}, AcousticUnitLabelStr{2});
fprintf('n\n Tolerance - Low: %10.2f %s', ACS_MATRIX4{1,7}, AcousticUnitLabelStr{2});
fprintf('n\n High: %10.2f %s', ACS_MATRIX4{1,8}, AcousticUnitLabelStr{2});
fprintf('n\n Resonance Frequency: %10.2f %s', ACS_MATRIX4{1,9}, AcousticUnitLabelStr{2});
fprintf('n\n Inherent Noise: %10.2f %s', ACS_MATRIX4{1,10}, AcousticUnitLabelStr{2});
fprintf('n\n Clipping Limit: %10.2f %s', ACS_MATRIX4{1,11}, AcousticUnitLabelStr{2});
fprintf('n\n Stability Conditions - Time: %10.2f %s', ACS_MATRIX4{1,12}, AcousticUnitLabelStr{2});
fprintf('n\n Temperature: %10.2f %s', ACS_MATRIX4{1,13}, AcousticUnitLabelStr{2});
fprintf('n\n Relative Humidity: %10.2f %s', ACS_MATRIX4{1,14}, AcousticUnitLabelStr{2});

% Environmental Properties
fprintf('n\n\n Power Requirements');
fprintf('n\n Voltage Range - Min: %10.2f %s', ACS_MATRIX4{1,12}, AcousticUnitLabelStr{12});
fprintf('n\n Max: %10.2f %s', ACS_MATRIX4{1,13}, AcousticUnitLabelStr{13});
fprintf('n\n Nominal Current: %10.2f %s', ACS_MATRIX4{1,14}, AcousticUnitLabelStr{14});

% Environmental Properties
fprintf('n\n\n Physical Properties');
fprintf('n\n Sensor Dimensions - Length: %9.3f %s', ACS_MATRIX4{1,2});
fprintf('n\n Width: %9.3f %s', ACS_MATRIX4{1,3});
fprintf('n\n Height: %9.3f %s', ACS_MATRIX4{1,4});
fprintf('n\n Sensor Mass: %9.3f %s', ACS_MATRIX4{1,5});
fprintf('n\n Sensor Comments: %s', ACS_MATRIX4{1,15});

%**** Writes a text file summary of the Planetary Properties
Fout = fopen('ACOUSTIC_SENSOR_Summary.txt','w+');
fprintf(Fout, '*\n\n** ACOUSTIC SENSOR SUMMARY RESULTS ***');
fprintf(Fout, '*\n\n*** INPUTS ***');
fprintf(Fout, '*\n Unit System: %s', UNITS);
D8. Density Sensors

\% function DENSITY_SENSORS(DENS\_TYPE,PLDENSITY,UNITS,Print)
\% Density Sensor Database
\% Developed by: Keith Schreck
\% Mechanical and Aerospace Engineering
\% San Jose State University
\% Date: Fall 2007
\% This file contains Density Sensor values for the ISSPO program.
\% Data here is loaded into the main program to determine the proper sensor
\% to use for Missions based on the Planetary Atmosphere composition.
\% Values are assigned to a variable name and saved to a MatLAB '.mat' file
\% and loaded when calling the ISSPO program.
\% Some of the values are based on the Unit system chosen and are loaded
\% in separate sections.
\% References
\% function DENSITY_SENSORS(DENS\_TYPE,PLDENSITY,UNITS,Print)
\% Load Constants values into Local Program
load('Constants_DB.mat');
load('Planet_Data.mat');

% Assign Planetary Surface Properties to Local Variable
PLTEMP = Planet_Properties{7};
PLPRESSURE = Planet_Properties{8};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Density Sensor Data Base Created Based on un Unit System
%% 2 - D Matrix DENSITYSENSOR_DB(DS_Type, Property)
if strcmpi(UNITS,'SI') == 1
  DENSITY_CNVT = kgm3_to_gcm3;
  % Properties [ SENSOR RANGE ACCURACY REPEATABILITY TEMPERATURE RANGE TEST PRESSURE ]
  % Units [ kg's ]
  DENSITYSENSOR_DB = ['MMS 7826' 0 3 0.001 0.0001 -50.0 200.0 0];
  'LIQUID';
  'MMS 7826L' 0 3 0.001 0.0001 -50.0 200.0 0;
  'LIQUID';
  'MMS 7828' 0 3 0.001 0.0001 -50.0 200.0 0;
  'LIQUID';
  'MMS 7828L' 0 3 0.001 0.0001 -50.0 200.0 0;
  'LIQUID';
  'MMS 7815' 0 3 0.00015 0.00002 -50.0 110.0 22500;
  'LIQUID';
  'MMS 7815L' 0 3 0.00015 0.00002 -50.0 110.0 22500;
  'LIQUID';
  'EGL 7845' 0 3 0.005 0 -50.0 160.0 15000;
  'LIQUID';
  'EGL 7847' 0 3 0.005 0 -50.0 160.0 3000;
  'LIQUID';
  'MMS 7845' 0 3 0.00035 0.00005 -50.0 110.0 15000;
  'GAS';
  'MMS 7846' 0 3 0.0005 0.00005 -50.0 110.0 7500;
  'GAS';
  'MMS 7847' 0 3 0.00035 0.00005 -50.0 110.0 3000;
  'GAS';
  'MMS 7848' 0 3 0.0005 0.00005 -50.0 110.0 3000;
  'GAS';
  'MMS 7849' 0 3 0.00035 0.00005 -50.0 110.0 3000;
  'GAS';
  'MMS 7850' 0 3 0.0005 0.00005 -50.0 110.0 3000;
  'GAS';
  'MMS 7851' 0 3 0.00035 0.00005 -50.0 110.0 3000;
  'GAS';
  'MMS 7852' 0 3 0.0005 0.00005 -50.0 110.0 3000;
% Sort Database Sensors by GAS or Liquid Function Capability

\begin{verbatim}
   tmp = 1;
   for d = 1:size(DENSITYSENSOR_DB,1)
       if strcmpi(DENSITYSENSOR_DB{d,20},DENS_TYPE) == 1
           for s = 1:size(DENSITYSENSOR_DB,2)
               D_MATRIX{tmp,s} = DENSITYSENSOR_DB{d,s};
           end
           tmp = tmp + 1;
       end
   end

   c = 1;
   % Build Up Database of Possible Density Sensors Based on Density Range
   for d = 1:size(D_MATRIX,1)
       if (PLDENSITY*DENSITY_CNVT > D_MATRIX{d,2}) && (PLDENSITY*DENSITY_CNVT <= D_MATRIX{d,3})
           for a = 1:size(D_MATRIX,2)
               D_MATRIX2{c,a} = D_MATRIX{d,a};
           end
           c = c+1;
       end
\end{verbatim}
if exist('D_MATRIX2') == 0
fprintf('

 *** WARNING ***
NO VIABLE DENSITY SENSOR EXISTS IN DATABASE!

Other methods to determine Planetary Atmospheric Density must be used.

');
end

% Continue Selection Process From Viable Options - Bypass if None Exists
if exist('D_MATRIX2') == 1
% Select Via Temperature Operational Range
   c = 1;
   % Build Up Database of Possible Density Sensors Based on Temperature Range
   for t = 1:size(D_MATRIX2,1)
      if (PLTEMP >= D_MATRIX2{t,6}) && (PLTEMP <= D_MATRIX2{t,7})
         for a = 1:size(D_MATRIX2,2)
            D_MATRIX3{c,a} = D_MATRIX2{t,a};
         end
         c = c+1;
      end
   end
   % Copies D_MATRIX2 to D_MATRIX3 if all Sensors Eliminated based on Temperature Range
   if exist('D_MATRIX3') == 0
      D_MATRIX3 = D_MATRIX2;
   end
% Select Via Pressure Operational Range
   c = 1;
   % Build Up Database of Possible Density Sensors Based on Pressure Range
   for t = 1:size(D_MATRIX3,1)
      if (PLPRESSURE >= D_MATRIX3{t,6}) && (PLPRESSURE <= D_MATRIX3{t,7})
         for a = 1:size(D_MATRIX3,2)
            D_MATRIX4{c,a} = D_MATRIX3{t,a};
         end
         c = c+1;
      end
   end
   % Copies D_MATRIX3 to D_MATRIX4 if all Sensors Eliminated based on Pressure Range
   if exist('D_MATRIX4') == 0
      D_MATRIX4 = D_MATRIX3;
   end
% Create Mass / Volume Scaling Factor Optimization
   for mv = 1:size(D_MATRIX4,1)
      SCALE_FAC(mv) = D_MATRIX4{mv,10} * D_MATRIX4{mv,11} * D_MATRIX4{mv,12} * D_MATRIX4{mv,13};
      if (D_MATRIX4{mv,10} == 0) || (D_MATRIX4{mv,11} == 0) || (D_MATRIX4{mv,12} == 0) || (D_MATRIX4{mv,13} == 0)
         SCALE_FAC(mv) = 9.99E99;
      end
   end
[MIN_SCL, INDEX] = min(SCALE_FAC);

% Build Sensor Matrix Based on Scale Factors
k = 1;
for c = 1:length(SCALE_FAC)
    if SCALE_FAC(c) == MIN_SCL
        for j = 1:size(D_MATRIX4,2)
            D_MATRIX5{k,j} = D_MATRIX4{c,j};
        end
        k = k + 1;
    end
end

% Copies D_MATRIX4 to D_MATRIX5 if all Sensors Eliminated based on Pressure Range
if exist('D_MATRIX5') == 0
    D_MATRIX5 = D_MATRIX4;
end

if size(D_MATRIX5,1) > 1
    % Find the Min Accuracy Value
    for n = 1:size(D_MATRIX5,1)
        D_DELTA_RNG(n) = D_MATRIX5{n,4};
    end
    % Find the Minimum Refraction Operating Range of the Sensor
    MIN_D_DELTA_RNG = min(D_DELTA_RNG);

    % For Multiple Sensors with the Same Range Reduce Sensor Matrix to Those
    % with the Greatest Range
    k = 1;
    for c = 1:length(D_DELTA_RNG)
        if D_DELTA_RNG(c) == MIN_D_DELTA_RNG
            for j = 1:size(D_MATRIX5,2)
                D_MATRIX6{k,j} = D_MATRIX5{c,j};
            end
            k = k + 1;
        end
    end
end

% Copies D_MATRIX5 to D_MATRIX6 if all Sensors Eliminated
if exist('D_MATRIX6') == 0
    D_MATRIX6 = D_MATRIX5;
end

% If NO Viable Sensor Exists Load Zero Values into Data Matrix
if exist('D_MATRIX6') == 0
    D_MATRIX6{1,1} = 'N/A';
    for d = 2:size(DENSITYSENSOR_DB,2)-1
D_MATRIX6{1,1} = 0.0;
end
D_MATRIX6{1,20} = 'N/A';
D_MATRIX6{1,21} = 'N/A';
end

% Save Density Sensor Properties to Data File
save('DENSITY_SENSOR_Data.mat');
save('DENSITY_SENSOR_Final.mat','D_MATRIX6', 'DensityUnitLabelStr');

% *** PRINT SUMMARY Output Section for 'Print' = 'Y'
if strcmpi(Print, 'Y') == 1
    fprintf('

 *** DENSITY SENSOR SUMMARY RESULTS ***

*** INPUTS ***
Density Sensor Operating Type: %s', DENSITY_SENSOR.Final_Type);
Planetary Surface Density: %7.2f %s', PLDENSITY, DensityUnit);
Unit System: %s', UNITS);

% Density Sensor Properties
Density Sensor Type: %s', D_MATRIX6{1,1});
Density Sensing Range - Low: %9.3f %s', D_MATRIX6{1,2}, DensityUnitLabelStr{2});
Density Sensing Range - High: %9.3f %s', D_MATRIX6{1,3}, DensityUnitLabelStr{3});
Sensor Accuracy: %9.6f %s', D_MATRIX6{1,4}, DensityUnitLabelStr{4});
Sensor Repeatability: %10.7f %s', D_MATRIX6{1,5}, DensityUnitLabelStr{5});
Sensor Tolerance: %10.6f %s', D_MATRIX6{1,17}, DensityUnitLabelStr{17});
Sensor Viscosity: %9.3f %s', D_MATRIX6{1,18}, DensityUnitLabelStr{18});
Sensing Flowrates: %9.3f %s', D_MATRIX6{1,19}, DensityUnitLabelStr{19});

% Environmental Properties
Temperature Range - Low: %9.3f %s', D_MATRIX6{1,6}, DensityUnitLabelStr{6});
Temperature Range - High: %9.3f %s', D_MATRIX6{1,7}, DensityUnitLabelStr{7});
Test Pressure: %9.3f %s', D_MATRIX6{1,8}, DensityUnitLabelStr{8});
Max Operating Pressure: %9.3f %s', D_MATRIX6{1,9}, DensityUnitLabelStr{9});
Vibration Loads: %9.3f %s', D_MATRIX6{1,16}, DensityUnitLabelStr{16});

% Physical Properties
Sensor Mass: %9.3f %s', D_MATRIX6{1,10}, DensityUnitLabelStr{10});
Sensor Dimensions - Length: %9.3f %s', D_MATRIX6{1,11}, DensityUnitLabelStr{11});
Height: %9.3f %s', D_MATRIX6{1,13}, DensityUnitLabelStr{13});

% Power Requirements
Input Voltage: %9.3f %s', D_MATRIX6{1,14}, DensityUnitLabelStr{14});
Input Amperage: %9.3f %s', D_MATRIX6{1,15}, DensityUnitLabelStr{15});
% Prints a text file summary of the Planetary Properties

Fout = fopen('DENSITY_SENSOR_Summary.txt', 'w+');
fprintf(Fout, 'DENSITY SENSOR SUMMARY RESULTS ***

** INPUTS ***
Density Sensor Operating Type: %s', DENS_TYPE);
fprintf(Fout, 'Planetary Surface Density: %7.2f %s', PLDENSITY, DensityUnit);
fprintf(Fout, 'Unit System: %s', UNITS);
fprintf(Fout, '************************************************************************

** Sensor Properties **
Density Sensor Type: %s', D_MATRIX6{1,1});
Density Sensing Range - Low: %9.3f %s', D_MATRIX6{1,2}, DensityUnitLabelStr{2});
Density Sensing Range - High: %9.3f %s', D_MATRIX6{1,3}, DensityUnitLabelStr{3});
Sensor Accuracy: %9.6f %s', D_MATRIX6{1,4}, DensityUnitLabelStr{4});
Sensor Repeatability: %10.7f %s', D_MATRIX6{1,5}, DensityUnitLabelStr{5});
Sensor Tolerance: %10.6f %s', D_MATRIX6{1,6}, DensityUnitLabelStr{6});
Sensor Viscosity: %9.3f %s', D_MATRIX6{1,7}, DensityUnitLabelStr{7});
Sensing Flowrates: [Low: %9.3f %s', D_MATRIX6{1,8}, DensityUnitLabelStr{8}]
                [High: %9.3f %s', D_MATRIX6{1,9}, DensityUnitLabelStr{9}]);

** Environmental Properties **
Temperature Range - Low: %9.3f %s', D_MATRIX6{1,10}, DensityUnitLabelStr{10});
Temperature Range - High: %9.3f %s', D_MATRIX6{1,11}, DensityUnitLabelStr{11});
Test Pressure: %9.3f %s', D_MATRIX6{1,12}, DensityUnitLabelStr{12});
Max Operating Pressure: %9.3f %s', D_MATRIX6{1,13}, DensityUnitLabelStr{13});
Vibration Loads: %9.3f %s', D_MATRIX6{1,14}, DensityUnitLabelStr{14});
Physical Properties: %9.3f %s', D_MATRIX6{1,15}, DensityUnitLabelStr{15});

** Power Requirements **
Input Voltage: %9.3f %s', D_MATRIX6{1,16}, DensityUnitLabelStr{16});
Input Amperage: %9.3f %s', D_MATRIX6{1,17}, DensityUnitLabelStr{17});
Sensor Comments: %s', D_MATRIX6{1,18});

fclose(Fout);
D9. GCMS Sensors

% function GCMS_SENSORS(GCMS_TYPE,GCMS_RANGE,UNITS,Print)
% Density Sensor Database
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
% This file contains Gas Chromatograph Mass Spectrometer Sensor (GCMS)
% values for the ISSPO program. Data here is loaded into the main program
% to determine the proper sensor to use for Missions based on the Planetary
% Atmosphere composition. Values are assigned to a variable name and saved
% to a Matlab '.mat' file and loaded when calling the ISSPO program.
% Some of the values are based on the Unit system chosen and are loaded
% in separate sections.
% References
% http://www.hitachi-hta.com/pageloader.aspx?type=product&id=390&orgid=45&clid=CMCQxI2pepACFRpsagodVMeMxG
% http://www.spectrometer.com/
% http://www.thinksrs.com/products/RGA.htm
% http://www.restek.com/

function GCMS_SENSORS(GCMS_TYPE, GCMS_RANGE, UNITS, Print)
%
% Load Constants values into Local Program
load('Constants_DB.mat');
load('Planet Data.mat');

if strcmpi(GCMS_TYPE, 'WAVELENGTH') == 1
    % GCMS Sensor Data Base Created Based on Unit System
    % 2 - D Matrix GCMSSENSOR_DB(DB_Type, Property)
    if strcmpi(UNITS, 'SI') == 1
        Properties = [ SENSOR WAVELENGTH RANGE
            DIMENSIONS - mm POWER
            WEIGHT - kg Volts
            type
        ];
        GCMSSENSOR_DB = {'U-0080D' 190 1100 'AC' [ 1 2 5 10 20 50 ] 'Photodiode Array';
            'U-1900' 190 1100 'AC' [ 2 5 10 20 50 ] 'Photodiode Array';
            'U-2900' 190 1100 'AC' [ 5 25 50 ] 'Silicon photodiode';
            'U-3010' 190 900 'AC' [ 10 20 50 ] 'Silicon photodiode';
            'U-4100' 190 2600 'AC' [ 0 ] 'Single Photomultiplier';
            'U-5010' 190 2600 'AC' [ 0 ] 'Photomultiplier (UV-Vis region) Cooled'};
if strcmpi(UNITS,'British') == 1

% Properties
% SENSOR WAVELENGTH RANGE SPECTRAL WIDTH RESOLUTION SCAN TIME DETECTOR ARRAY
% WEIGHT WEIGHT WEIGHT
% Units [ TYPE Low 10^-7 in High 10^-7 in 10^-9 in sec # of Elements ]

22.05 16.54 11.57 4.72 [ 100 120 230 240 ] 'AC' [ 1 2 5 10 20 50 ] 'Photodiode Array' ;

59.5 22.5 15 12.3 [ 100 120 230 240 ] 'AC' [ 0 ] 'Silicon photodiode' ;

68 24 20 11.5 [ 100 120 230 240 ] 'AC' [ 5 25 50 ] 'Silicon photodiode' ;

110 26.8 27.2 13 [ 100 120 230 240 ] 'AC' [ 0 ] 'Single Photomultiplier' ;
264 'U-4100' 68.89 1023.62 'AC' 3.1496 0 60.63 1
PBS(NIR region) ;

77 'P-2500' 86.61 287.40 'AC' 0 10.20 1
'P-7000' 78.74 295.27 'AC' 7.8740 0 1

90.39 '2-2000' 71.80 354.33 'AC' 1.0236 0 0
'Z-2300' 74.80 354.33 'AC' 1.0236 0 0

416.67 '2-2700' 74.80 354.33 'AC' 1.0236 0 0

233.70 'F-2500' 86.61 287.40 'AC' 0 10.20 1
'F-7000' 77.84 295.27 'AC' 0 3.55 1

306.44 'Z-2000' 27.56 'N/A' 0 5.04 1
'Z-2300' 27.56 'N/A' 0 5.04 1
'Z-2700' 27.56 'N/A' 0 5.04 1

0.1969 '10^-7 in' '10^-7 in' '10^-9 in' 'sec' 'N/A' 'Ibm' 'in' 'in' 'in'

2.20 5.04 4.49 3.31 [ ] 'N/A' [ ] 'Monochromator' 0 0 2048 ;

0 0 0 0 [ ] 'N/A' [ ] 'Imaging Spectrograph' 0 55.12 0.01 3864 ;

1.5 7.0 4.73 2.25 [ ] 'N/A' [ ] 'Linear CCD 3864 pixels, Photo Diode Array 1024 pixels' ;

11.0 7.0 10.5 5.5 [ ] 'N/A' [ ] '2D CCD 3864 pixels, Photo Diode Array 1024 pixels' ;

7.05 13.35 3.31 3.31 [ ] 'N/A' [ ] 'Images Monochromator/Spectrograph' 0 19.69 1 1

45 16.4 16.6 7.6 [ ] 'N/A' [ ] 'Imaging Spectrometer' 0 0 2362 0.01 3864 ;

62 25.51 18.09 7.78 [ ] 'N/A' [ ] 'High resolution spectrometer' 0 5.0 1 1

143.3 29.13 13.78 13.78 [ ] 'N/A' [ ] 'High resolution spectrometer' 0 0.3937 1 1

0 32.99 12.24 12.99 [ ] 'N/A' [ ] 'High resolution spectrometer' 0 0 5.0 1

\%

% Comments on Features / Use
GCMSSENS0R_DB{1, 16} = { 'DNA/RNA measurement'
concentration (Nucleotide), Background correction (no correction, fixed wavelength, Tangent), Baseline correction, Input of
dilution magnification' ;
'Calculation of protein concentration' 'Concentration calculation by Warburg-Christian method,
Other quantitative analysis with reagent';
'Labeled DNA' 'Verification that a fluorescent dye coupled with a primer
before DNA sequencing' ;
'Wavelength scan' 'Baseline correction, Overlay of 20 channel data, Peak
detection, Recalculation, Printing' ;
'Time scan' 'Baseline correction, Overlay of 20 channel data, Auto
scale, Recalculation, Printing' ;

end

% Comments on Features / Use
GCMSSENS0R_DB{1, 16} = { 'DNA/RNA measurement'
concentration (Nucleotide), Background correction (no correction, fixed wavelength, Tangent), Baseline correction, Input of
dilution magnification' ;
'Calculation of protein concentration' 'Concentration calculation by Warburg-Christian method,
Other quantitative analysis with reagent';
'Labeled DNA' 'Verification that a fluorescent dye coupled with a primer
before DNA sequencing' ;
'Wavelength scan' 'Baseline correction, Overlay of 20 channel data, Peak
detection, Recalculation, Printing' ;
'Time scan' 'Baseline correction, Overlay of 20 channel data, Auto
scale, Recalculation, Printing' ;
'Quantitative calculation' 'Plotting of calibration curve, Statistical operation, One, two and three wavelength calculations, Saving and printing of calibration curve and measurement data' 'Other analysis' 'Auro-zero correction, Cell-length normalization, Input of sample name, comment and file name, constant display of measurement parameters, Data saving and retrieval' 'Resolution' 'Satisfies European Pharmacopoeia Bandpass'

GCMSSENSOR_DB{2,16} = { 'Modes of operation' 'Scanning, Time (Kinetics), Photometry and Ratio [DNA/RNA Analysis]' }
GCMSSENSOR_DB{3,16} = { 'Band pass' 'Variable 0.1, 0.5, 1, 2, 4, and 5 nm' 'Scan speed' '0.3, 3, 15, 30, 60, 120, 300, 600, 1200 and 1800 nm/min' 'Sample Compartment accommodates' 'automated analysis, specular and diffuse reflectance, turbid samples, thin films, long path cells, and continuous flow-through monitoring' 'Hitachi''s U-3010 Model' 'double-beam single-monochromator' 'Hitachi''s U-3310 Model' 'double-beam double-monochromator'}

GCMSSENSOR_DB{4,16} = { 'Band pass' 'NIR Region: Switching in 0.2 nm steps from 0.24 to 8.0 nm' 'Scan speed' '0.3 (0.75), 1 (7.5), 15 (37.5), 30 (75), 60 (150), 120 (300), 300 (750), 600 (1,500), 1200 (3,000) and 2400 (6,000) nm/min. Speeds in parenthesis for NIR region'}

GCMSSENSOR_DB{5,16} = { 'Band pass' 'Sample Compartment Dimensions' '300 x 930 x 980 mm Large Unit Dims'}

GCMSSENSOR_DB{6,16} = { 'Band pass' 'Monochromator system' 'prism/grating or grating/grating'}

GCMSSENSOR_DB{7,16} = { 'Band pass' 'Data Processing functions' 'Excitation side: 1, 2.5, 5, 10, 20 nm Emission side: 1, 30, 60, 240, 1200, 2400, 12000, 30000, and 60000 nm/min'; 'Response from 0 to 98% 0.002, 0.004, 0.01, 0.05, 0.1, 0.2, 4 s'

GCMSSENSOR_DB{8,16} = { 'Measurement Mode' 'Optics' 'Slit width' 'Sample Volume' 'Sample Number' 'Model Ranges' 'Optical Configuration' 'Slit width' 'Detector Array' 'pixels' 'Application' 'CCD - Linear 2D array <2048 pixels, PDA - NMOS & CMOS 128-1024 pixels' 'Low resolution absorption transmission and fluoresence' 'Multiple Models ranges from 190 to 2500 nm'; 'Fixed width, interchangeable. 3 pairs supplied: 0.5, 1 and 2 mm'}

GCMSSENSOR_DB{9,16} = { 'Measurement Mode' 'Optics' 'Slit width' 'Sample Volume' 'Sample Number' 'Model Ranges' 'Optical Configuration' 'Slit width' 'Detector Array' 'pixels' 'Application' 'Aberration-corrected concave holographic diffraction grating'}

GCMSSENSOR_DB{10,16} = { 'Measurement Mode' 'Optics' 'Slit width' 'Sample Volume' 'Sample Number' 'Model Ranges' 'Optical Configuration' 'Slit width' 'Detector Array' 'pixels' 'Application' 'Aberration-corrected concave holographic diffraction grating'}

GCMSSENSOR_DB{11,16} = { 'Model Ranges' '400 - 1600' 'Optical Configuration' 'Slit width' 'Application' 'Multiple Models ranges from 190 to 2500 nm'}

GCMSSENSOR_DB{12,16} = { 'Models' 'Detector Array' 'pixels' 'Application' 'Aberration-corrected concave holographic diffraction grating'}

"Hitachi''s U-3010 Model" 'double-beam single-monochromator' ;
"Hitachi''s U-3310 Model" 'double-beam double-monochromator'}
% Define Sensing Operational Wavelengths based on Atmospheric Composition
% Spectral Data From: http://laserstars.org/data/elements/
% NIST ASD: http://physics.nist.gov/PhysRefData/ASD/lines_form.html
k = 1;
for L = 1:2:length(PlanAtm_Prop)
switch PlanAtm_Prop(L)
case 'H'
    if strcmpi(UNITS, 'SI') == 1;
        Lambdas(k) = 91.13;
        Lambdas(k+1) = 364.51;
        Lambdas(k+2) = 824.14;
        Lambdas(k+3) = 1458.17;
        Lambdas(k+4) = 2278.17;
        Lambdas(k+5) = 3280.56;
        k = k+6;
    end
    if strcmpi(UNITS, 'British') == 1;
        Lambdas(k) = 35.877; % 10^-7 inches
        Lambdas(k+1) = 143.507; % 10^-7 inches
        Lambdas(k+2) = 322.889; % 10^-7 inches
        Lambdas(k+3) = 574.083; % 10^-7 inches
        Lambdas(k+4) = 896.917; % 10^-7 inches
        Lambdas(k+5) = 1291.56; % 10^-7 inches
        k = k+6;
    end
    case 'H2'
    if strcmpi(UNITS, 'SI') == 1;
        Lambdas(k) = 397.007;
        Lambdas(k+1) = 410.174;
        Lambdas(k+2) = 434.047;
        Lambdas(k+3) = 486.133;
        Lambdas(k+4) = 656.285;
        k = k+5;
    end
    if strcmpi(UNITS, 'British') == 1;
        Lambdas(k) = 156.302; % 10^-7 inches
        Lambdas(k+1) = 161.486; % 10^-7 inches
        Lambdas(k+2) = 170.885; % 10^-7 inches
        Lambdas(k+3) = 258.380; % 10^-7 inches
        k = k+5;
    end
    case 'CO2'
    if strcmpi(UNITS, 'SI') == 1;
        Lambdas(k) = 2600.00;
        Lambdas(k+1) = 4000.00;
        Lambdas(k+2) = 13000.00;
        k = k+3;
    end
    if strcmpi(UNITS, 'British') == 1;
        Lambdas(k) = 1023.62; % 10^-7 inches
        Lambdas(k+1) = 1574.80; % 10^-7 inches
        Lambdas(k+2) = 5118.09; % 10^-7 inches
        k = k+3;
end
```c
x = k

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
    end

k = 11

end

if (k) == (IS, 'SLINN')
    case 20
```
case 'CH4'
    if strcmpi(UNITS, 'SI') == 1;
        Lambdas(k) = 350.000;
        Lambdas(k+1) = 800.000;
        k = k+2;
    end
    if strcmpi(UNITS, 'British') == 1;
        Lambdas(k) = 117.795; % 10^-7 inches
        Lambdas(k+1) = 314.960; % 10^-7 inches
        k = k+2;
    end
    case 'He'
    if strcmpi(UNITS, 'SI') == 1;
        Lambdas(k) = 396.474;
        Lambdas(k+1) = 447.149;
        Lambdas(k+2) = 501.569;
        Lambdas(k+3) = 507.562;
        Lambdas(k+4) = 667.814;
        Lambdas(k+5) = 706.518;
        k = k+6;
    end
    if strcmpi(UNITS, 'British') == 1;
        Lambdas(k) = 156.092; % 10^-7 inches
        Lambdas(k+1) = 176.043; % 10^-7 inches
        Lambdas(k+2) = 197.468; % 10^-7 inches
        Lambdas(k+3) = 231.324; % 10^-7 inches
        Lambdas(k+4) = 262.919; % 10^-7 inches
        Lambdas(k+5) = 278.157; % 10^-7 inches
        k = k+6;
    end
    case 'Na'
    if strcmpi(UNITS, 'SI') == 1;
        Lambdas(k) = 371.107;
        Lambdas(k+1) = 431.370;
        Lambdas(k+2) = 430.881;
        Lambdas(k+3) = 439.281;
        Lambdas(k+4) = 444.670;
        Lambdas(k+5) = 453.332;
        Lambdas(k+6) = 455.153;
        Lambdas(k+7) = 519.165;
        Lambdas(k+8) = 654.575;
        Lambdas(k+9) = 780.978;
        k = k+10;
    end
    if strcmpi(UNITS, 'British') == 1;
        Lambdas(k) = 156.092; % 10^-7 inches
        Lambdas(k+1) = 161.957; % 10^-7 inches
        Lambdas(k+2) = 169.636; % 10^-7 inches
        Lambdas(k+3) = 172.945; % 10^-7 inches
        Lambdas(k+4) = 175.067; % 10^-7 inches
        Lambdas(k+5) = 178.477; % 10^-7 inches
        Lambdas(k+6) = 179.194; % 10^-7 inches
        Lambdas(k+7) = 204.396; % 10^-7 inches
        k = k+10;
Lambdas(k+8) = 257.707; \% 10^-7 inches
Lambdas(k+9) = 307.472; \% 10^-7 inches
k = k+10;
end

case 'H2O'
if strcmpi(UNITS, 'SI') == 1;
Lambdas(k) = 730.00;
Lambdas(k+1) = 820.00;
Lambdas(k+2) = 930.00;
Lambdas(k+3) = 970.00;
Lambdas(k+4) = 1200.00;
Lambdas(k+5) = 1450.00;
Lambdas(k+6) = 1920.00;
Lambdas(k+7) = 2500.00;
k = k+8;
end
if strcmpi(UNITS, 'British') == 1;
Lambdas(k) = 287.402; \% 10^-7 inches
Lambdas(k+1) = 322.835; \% 10^-7 inches
Lambdas(k+2) = 366.142; \% 10^-7 inches
Lambdas(k+3) = 381.690; \% 10^-7 inches
Lambdas(k+4) = 472.441; \% 10^-7 inches
Lambdas(k+5) = 570.866; \% 10^-7 inches
Lambdas(k+6) = 767.717; \% 10^-7 inches
Lambdas(k+7) = 984.252; \% 10^-7 inches
k = k+8;
end
end

case 'Aire'
if strcmpi(UNITS, 'SI') == 1;
Lambdas(k) = 392.572,
Lambdas(k+1) = 407.201;
Lambdas(k+2) = 413.172;
Lambdas(k+3) = 415.859;
Lambdas(k+4) = 420.067;
Lambdas(k+5) = 427.753;
Lambdas(k+6) = 434.806;
Lambdas(k+7) = 442.600;
Lambdas(k+8) = 454.505;
Lambdas(k+9) = 460.957;
Lambdas(k+10) = 472.687;
Lambdas(k+11) = 487.086;
Lambdas(k+12) = 696.543;
Lambdas(k+13) = 706.722;
Lambdas(k+14) = 714.704;
k = k+15;
end
if strcmpi(UNITS, 'British') == 1;
Lambdas(k) = 154.556; \% 10^-7 inches
Lambdas(k+1) = 160.315; \% 10^-7 inches
Lambdas(k+2) = 162.666; \% 10^-7 inches
Lambdas(k+3) = 163.724; \% 10^-7 inches
Lambdas(k+4) = 165.381; \% 10^-7 inches
Lambdas(k+5) = 168.407; \% 10^-7 inches
Lambdas(k+6) = 171.183; \% 10^-7 inches
% Based on Atmospheric Contents Determine Range Of Wavelengths of Atmosphere
Min_Lambda = min(Lambdas);
Max_Lambda = max(Lambdas);

% Build Up Sensor Database of Sensors whose Range Exceeds Atmospheric Components Range
\texttt{c} = 1;
\texttt{for g = 1:size(GCMSSENSOR_DB,1)}
    \texttt{if (Min_Lambda >= GCMSSENSOR_DB\{g,2\}) && (Max_Lambda <= GCMSSENSOR_DB\{g,3\})}
        \texttt{for a = 1:size(GCMSSENSOR_DB,2)}
            \texttt{GCMS_MATRIX\{c,a\} = GCMSSENSOR_DB\{g,a\};}
        \texttt{end}
        \texttt{c = c+1;}
    \texttt{end}
\texttt{end}

% Down Select a Single Sensor If Multiple Options Exist in the Database
\texttt{if size(GCMS_MATRIX,1) >1}
% Create Mass / Volume Scaling Factor Optimization
\texttt{for mv = 1:size(GCMS_MATRIX,1)}
    \texttt{GCMS_SCALE_FAC\{mv\} = GCMS_MATRIX\{mv,8\} * GCMS_MATRIX\{mv,9\} * GCMS_MATRIX\{mv,10\} * GCMS_MATRIX\{mv,11\};}
    \texttt{if (GCMS_MATRIX\{mv,8\} == 0) || (GCMS_MATRIX\{mv,9\} == 0) || (GCMS_MATRIX\{mv,10\} == 0) || (GCMS_MATRIX\{mv,11\} == 0) GCMS_SCALE_FAC\{mv\} = 9.99E99;}
\texttt{end}
\texttt{[ GCMS_MIN_SCL, INDEX ] = min(GCMS_SCALE_FAC);}
% Build Sensor Matrix Based on Scale Factors
\texttt{k = 1;}
\texttt{for c = 1:length(GCMS_SCALE_FAC)}
    \texttt{if GCMS_SCALE_FAC\{c\} == GCMS_MIN_SCL}
        \texttt{for j = 1:size(GCMS_MATRIX,2)}
            \texttt{GCMS_MATRIX2\{k,j\} = GCMS_MATRIX\{c,j\};}
        \texttt{end}
        \texttt{k = k + 1;}
    \texttt{end}
\texttt{end}
% Copies GCMS_MATRIX to GCMS_MATRIX2 if all Sensors Eliminated based on Mass Volume Scale Factor
\texttt{if exist(\texttt{\'GCMS_MATRIX2\texttt{\') == 0 \texttt{GCMS_MATRIX2 = GCMS_MATRIX;}})
% Down Select Sensor Based On Smallest Resolution Range
\texttt{if size(GCMS_MATRIX2,1) >1}
    \texttt{for r = 1:size(GCMS_MATRIX2,1)}
        \texttt{GCMS_RES\{r\} = GCMS_MATRIX2\{r,5\};}
        \texttt{if GCMS_RES\{r\} == 0 GCMS_RES\{r\} = 9.99E99;}
end
end

[ Min_GCMS_RES, INDEX ] = min(GCMS_RES);

% Build Sensor Matrix Based on Volumes
k = 1;
for c = 1:length(GCMS_RES)
    if GCMS_RES(c) == Min_GCMS_RES
        for j = 1:size(GCMS_MATRIX2,2)
            GCMS_MATRIX3{k,j} = GCMS_MATRIX2{c,j};
        end
        k = k + 1;
    end
end

% Copies GCMS_MATRIX2 to GCMS_MATRIX3 if all Sensors Eliminated based on Minimum Resolution
if exist('GCMS_MATRIX3') == 0
    GCMS_MATRIX3 = GCMS_MATRIX2;
end

% Copies GCMS_MATRIX to GCMS_MATRIX3 if a single Sensor meets Operational Range
if exist('GCMS_MATRIX3') == 0
    GCMS_MATRIX3 = GCMS_MATRIX;
end

% Save Density Sensor Properties to Data File
save('GCMS_SENSOR_Data.mat');
save('GCMS_SENSOR_Final.mat','GCMS_MATRIX3','GCMSUnitLabelStr');

% *****************************************************
% % *** PRINT SUMMARY Output Section for 'Print' = 'Y'
if strcmpi(Print,'Y') == 1
    fprintf('

 *** GCMS SENSOR SUMMARY RESULTS ***
');
    fprintf('INPUTS ');
    fprintf('Unit System: %s', UNITS);
    fprintf('GCMS Sensor Properties ');
    fprintf('GCMS Sensor Type: %s', GCMS_MATRIX3{1,1});
    fprintf('Sensing Wavelength Range - Low: %10.2f %s', GCMS_MATRIX3{1,2}, GCMSUnitLabelStr{2});
    fprintf('High: %10.2f %s', GCMS_MATRIX3{1,3}, GCMSUnitLabelStr{3});
    fprintf('Spectral Width: %10.4f %s', GCMS_MATRIX3{1,4}, GCMSUnitLabelStr{4});

% *****************************************************
fprintf('Resolution: %10.4f %s', GCMS_MATRIX3{1,5}, GCMSUnitLabelStr{5});
fprintf('Scan Time: %10.2f %s', GCMS_MATRIX3{1,6}, GCMSUnitLabelStr{6});
fprintf('Sample Volume: %d', GCMS_MATRIX3{1,14}(1,s));
end
fprintf('Detector Array # of Elements: %d %s', GCMS_MATRIX3{1,7}, GCMSUnitLabelStr{7});
fprintf('Detector Type: %s', GCMS_MATRIX3{1,15});

fprintf('Power Requirements:
Voltage Source: %d %s', GCMS_MATRIX3{1,12}(1,v));
end
fprintf('Voltage Type: %s %s', GCMS_MATRIX3{1,13}, GCMSUnitLabelStr{13});

fprintf('Physical Properties:
Sensor Mass: %10.2f %s', GCMS_MATRIX3{1,8}, GCMSUnitLabelStr{8});
fprintf('Sensor Dimensions - Length: %10.2f %s', GCMS_MATRIX3{1,9}, GCMSUnitLabelStr{9});
fprintf('Width: %10.2f %s', GCMS_MATRIX3{1,10}, GCMSUnitLabelStr{10});
fprintf('Height: %10.2f %s', GCMS_MATRIX3{1,11}, GCMSUnitLabelStr{11});

fprintf('Sensor Comments:
Sensor Type: %s', GCMS_MATRIX3{1,16}{n,1});
fprintf('Sensing Wavelength Range:
Low: %10.2f %s', GCMS_MATRIX3{1,17}, GCMSUnitLabelStr{17});
fprintf('High: %10.2f %s', GCMS_MATRIX3{1,18}, GCMSUnitLabelStr{18});
fprintf('Spectral Width: %10.2f %s', GCMS_MATRIX3{1,19}, GCMSUnitLabelStr{19});
fprintf('Resolution: %10.2f %s', GCMS_MATRIX3{1,20}, GCMSUnitLabelStr{20});
fprintf('Scan Time: %10.2f %s', GCMS_MATRIX3{1,21}, GCMSUnitLabelStr{21});

% *** Writes a text file summary of the Planetary Properties
Fout = fopen('GCMS_SENSOR_Summary.txt','w+');
fprintf(Fout, 'GCMS SENSOR SUMMARY

*** INPUTS ***
Unit System: %s', UNITS);
fprintf(Fout, 'INPUTS ***

% GCMS Sensor Properties
fprintf(Fout, 'Sensor Properties ***
');
fprintf(Fout, 'Sensor Type: %s', GCMS_MATRIX3{1,1});
fprintf(Fout, 'Sensing Wavelength Range - Low: %10.2f %s', GCMS_MATRIX3{1,2}, GCMSUnitLabelStr{2});
fprintf(Fout, 'High: %10.2f %s', GCMS_MATRIX3{1,3}, GCMSUnitLabelStr{3});
fprintf(Fout, 'Spectral Width: %10.2f %s', GCMS_MATRIX3{1,4}, GCMSUnitLabelStr{4});
fprintf(Fout, 'Resolution: %10.2f %s', GCMS_MATRIX3{1,5}, GCMSUnitLabelStr{5});
fprintf(Fout, 'Scan Time: %10.2f %s', GCMS_MATRIX3{1,6}, GCMSUnitLabelStr{6});

fprintf(Fout, ' 
 Sample Volume: 
');
for s = 1:length(GCMS_MATRIX3{1,14})
fprintf(Fout, ' %3d', GCMS_MATRIX3{1,14}(1,s));
end
fprintf(Fout, ' %s', GCMSUnitLabelStr{14});
fprintf(Fout, '
 Detector Array # of Elements: 
');
fprintf(Fout, ' %d %s', GCMS_MATRIX3{1,7}, GCMSUnitLabelStr{7});
fprintf(Fout, '
 Detector Type: 
');
fprintf(Fout, ' %s %s', GCMS_MATRIX3{1,15}, GCMSUnitLabelStr{15});
fprintf(Fout, '

 Power Requirements');
fprintf(Fout, '
 Voltage Source: 
');
for v = 1:length(GCMS_MATRIX3{1,12})
fprintf(Fout, ' %4d', GCMS_MATRIX3{1,12}(1,v));
end
fprintf(Fout, ' %s', GCMSUnitLabelStr{12});
fprintf(Fout, '
 Voltage Type: 
');
fprintf(Fout, ' %s %s', GCMS_MATRIX3{1,13}, GCMSUnitLabelStr{13});
fprintf(Fout, '

 Physical Properties');
fprintf(Fout, '
 Sensor Mass: 
');
fprintf(Fout, ' %10.2f %s', GCMS_MATRIX3{1,8}, GCMSUnitLabelStr{8});
fprintf(Fout, '
 Sensor Dimensions - Length: 
');
fprintf(Fout, ' %10.2f %s', GCMS_MATRIX3{1,9}, GCMSUnitLabelStr{9});
fprintf(Fout, '
 Width: 
');
fprintf(Fout, ' %10.2f %s', GCMS_MATRIX3{1,10}, GCMSUnitLabelStr{10});
fprintf(Fout, '
 Height: 
');
fprintf(Fout, ' %10.2f %s', GCMS_MATRIX3{1,11}, GCMSUnitLabelStr{11});

fprintf(Fout, '

 Sensor Comments: 
');
for n = 1:length(GCMS_MATRIX3{1,16})
fprintf(Fout, '
 %40s %s', GCMS_MATRIX3{1,16}(n,1), GCMS_MATRIX3{1,16}(n,2));
end
fclose(Fout);

if strcmpi(GCMS_TYPE, 'MASS-CHARGE') == 1

% DEFINE MASS-CHARGE VARIABLE LIMITS INTERNAL
if strcmpi(GCMS_RANGE, 'HIGH') == 1
GCMS_RANGE_NUM = 300;
G_TOL_LOWER = 99;
end
if strcmpi(GCMS_RANGE, 'MEDIUM') == 1
GCMS_RANGE_NUM = 200;
G_TOL_LOWER = 99;
end
if strcmpi(GCMS_RANGE, 'LOW') == 1
GCMS_RANGE_NUM = 100;
G_TOL_LOWER = 99;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%?
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
GCMS Sensor Data Base Created Based on un Unit System

%%% GCMS Sensor Data Base Created Based on un Unit System
```matlab
if strcmpi(UNITS,'SI') == 1
    % Properties
    SCAN RANGE
    BAKEOUT TEMP
    % Units
    of Magnitude
    deg C
    GCMSSENSOR_DB1 = {
        'RGA100' 1 100 'Quadrupole' {'Faraday Cup' 'Electron Multiplier'}
        0.5        [ 2.6E-4 200 ] [ 5.0E-11 5.0E-14 ] [ 1.0E-4 1.0E-6 ] 70.00
    300.00;
    'RGA200' 1 200 'Quadrupole' {'Faraday Cup' 'Electron Multiplier'}
    0.5        [ 2.6E-4 200 ] [ 5.0E-11 5.0E-14 ] [ 1.0E-4 1.0E-6 ] 70.00
    'RGA300' 1 300 'Quadrupole' {'Faraday Cup' 'Electron Multiplier'}
    0.5        [ 2.6E-4 200 ] [ 5.0E-11 5.0E-14 ] [ 1.0E-4 1.0E-6 ] 70.00
    'Aeolos' 1 100 'Quadrupole' {'Electron Multiplier'}
    0.5        [ 0 ] [ 1.9E-14 ] [ 0 ] 300.00
    'GCMS-Huygens' 2 141 'Quadrupole' {'Electron Multiplier'}
    1E-6        [ 0 ] [ 1.275E-16 ] [ 7.50E-4 ] 100.00
    0.00;
}

    GCMSUnitLabelStr1 = {'N/A' 'amu' 'amu' 'N/A' 'N/A' 'N/A' 'amu' 'A/Torr' 'Torr' 'Torr' 'Torr' 'deg C'};

% Properties
% ION ENERGY
% Units
% Min Volts Max
% FOCUS VOLTAGE
% #
% Min mA Max
% ELECTRON CURRENT
% ELECTRON Type
% Low W Degas High
% FIELD RANGE
% ELECTRON
% POWER-TYP
% POWER-AVE
% POWER-MAX
% POWER REQ
% CURRENT
% MASS
% Dimensions - cm
% MASS STABILITY
% Types
% amu min V Type A kg
% bits/sec
% Units
% Length Width Height
% amu min V Type A kg
% GCMSSENSOR_DB3 = {
    46.61 10.42 7.88 0.30 24 'DC' 2.5 2.72
    0 0 0
```
### Properties

<table>
<thead>
<tr>
<th>SCAN RANGE</th>
<th>RESOLUTION</th>
<th>SENSOR</th>
<th>MASS RANGE</th>
<th>MASS FILTER</th>
<th>DETECTOR</th>
<th>DYNAMIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>deg F</td>
<td>psi</td>
<td>Low amu</td>
<td>High</td>
<td>psi</td>
<td>TYPE</td>
<td>Orders</td>
</tr>
</tbody>
</table>

### Units

- amu
- A/psi
- psi
- deg F

### GCMS SENSOR DB1

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Mass Range</th>
<th>Mass Filter</th>
<th>Detector</th>
</tr>
</thead>
<tbody>
<tr>
<td>'Quadrupole'</td>
<td>0.00 150</td>
<td>100</td>
<td>'Faraday Cup' 'Electron Multiplier'</td>
</tr>
</tbody>
</table>

### GCMS SENSOR DB2

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Mass Range</th>
<th>Mass Filter</th>
<th>Detector</th>
</tr>
</thead>
<tbody>
<tr>
<td>'Quadrupole'</td>
<td>0.00 150</td>
<td>100</td>
<td>'Faraday Cup' 'Electron Multiplier'</td>
</tr>
</tbody>
</table>

### Field Range

- Low W Degas
- High

### Electron

- Type
- Type
- Low W Degas
- High
- Min Volts
- Max Volts

---

```
% Properties

<table>
<thead>
<tr>
<th>DATA RATE</th>
<th>POWER-TYP</th>
<th>POWER-AVE</th>
<th>POWER-MAX</th>
<th>MASS STABILITY</th>
<th>POWER REQ</th>
<th>CURRENT</th>
<th>MASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>bits/sec</td>
<td>Units</td>
<td>[BTUs/hr]</td>
<td>[BTUs/hr]</td>
<td></td>
<td>[V]</td>
<td>[A]</td>
<td>[lbm]</td>
</tr>
<tr>
<td>28800</td>
<td>GCMSSENSOR_DB3 =</td>
<td>{18.35 4.10 3.10</td>
<td>0.1 30</td>
<td>24 'DC'</td>
<td>2.5 6.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28800</td>
<td>0 0 0 0 0 0</td>
<td>18.35 4.10 3.10</td>
<td>0.1 30</td>
<td>24 'DC'</td>
<td>2.5 6.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28800</td>
<td>0 0 0 0 0 0</td>
<td>18.35 4.10 3.10</td>
<td>0.1 30</td>
<td>24 'DC'</td>
<td>2.5 6.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>960 95.63 140.02 242.48</td>
<td></td>
<td>18.50 7.80 7.80 0 36</td>
<td>28 'DC'</td>
<td>1.68 38.14</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

% Comments on Features / Use

GCMSSENSOR_DB4{1,1} = {'Gas Chromatograph Residual GasAnalyzer';
    'Probe design consists of an ionizer, quadrupole mass filter, and detector';
    'Built-in degassing feature';
    'Faraday Cup or Electron Multiplier option';
    'RS-232c Computer Interface'};

GCMSSENSOR_DB4{2,1} = {'Gas Chromatograph Residual Gas Analyzer';
    'Probe design consists of an ionizer, quadrupole mass filter, and detector';
    'Faraday Cup or Electron Multiplier option';
    'RS-323c Computer Interface'};

GCMSSENSOR_DB4{3,1} = {'Gas Chromatograph Residual Gas Analyzer';
    'Probe design consists of an ionizer, quadrupole mass filter, and detector';
    'Built-in degassing feature';
    'Faraday Cup or Electron Multiplier option';
    'RS-232c Computer Interface'};

GCMSSENSOR_DB4{4,1} = {'Thermogravimetry and Mass Spectrometry';
    'Multiple Components Coupled Together';
    'Loss-free gas transfer to the QMS';
    'Faraday Cup or Electron Multiplier option';
    'RS-232c Computer Interface'};

GCMSSENSOR_DB4{5,1} = {'Combined Gas Chromatograph Mass Spectrometer';
    'Probe design consists of a 5 ion sources, quadrupole mass filter, and 2 Electron Multipliers';
    'Leak Management Holes';
    'Hydrogen gas used as Transport medium in sampling chambers';
    'Huygens Probe GCMS System - Custom Flight Proven configuration'};

% Combine Data Arrays into a Single Data Matrix
% Build Up Sensor Database of Sensors whose Range Meets Input Operational Range

\begin{verbatim}
c = 1;
for g = 1:size(GCMSSENSOR_DB,1)
    if (GCMSSENSOR_DB{g,3} <= GCMS_RANGE_NUM &
        (GCMSSENSOR_DB{g,3} > (GCMS_RANGE_NUM-G_TOL_LOWER))
        for a = 1:size(GCMSSENSOR_DB,2)
            GCMS_MATRIX{c,a} = GCMSSENSOR_DB{g,a};
        end
        c = c+1;
    end
end
\end{verbatim}

% Down Select from database if Multiple Sensors Exist
if size(GCMS_MATRIX,1) > 1
    % Down Select Sensor by Max Dynamic Range (Orders Of Magnitude)
    for g = 1:size(GCMS_MATRIX,1)
        GCMS_DynRANGE(g) = GCMS_MATRIX{g,6};
    end
    % Find the Max Dynamic Range Sensors
    [ MAX_GCMS_RANGE, INDEX ] = max(GCMS_DynRANGE);
    c = 1;
    for g = 1:size(GCMS_MATRIX,1)
        if GCMS_MATRIX{g,6} == MAX_GCMS_RANGE
            for a = 1:size(GCMS_MATRIX,2)
                GCMS_MATRIX2{c,a} = GCMS_MATRIX{g,a};
            end
            c = c + 1;
        end
    end
if exist('GCMS_MATRIX2') == 0
    GCMS_MATRIX2 = GCMS_MATRIX;
end
if size(GCMS_MATRIX2,1) > 1
    % Down Select a Single Sensor If Multiple Options Exist in the Database
    % Create Mass / Volume Scaling Factor Optimization
    for mv = 1:size(GCMS_MATRIX2,1)
        GCMS_SCALE_FAC(mv) = GCMS_MATRIX2{mv,25} * GCMS_MATRIX2{mv,26} * GCMS_MATRIX2{mv,27} * GCMS_MATRIX2{mv,33};
        if (GCMS_MATRIX{mv,25} == 0) || (GCMS_MATRIX{mv,26} == 0) || (GCMS_MATRIX{mv,27} == 0) || (GCMS_MATRIX{mv,33} == 0)
            GCMS_SCALE_FAC(mv) = 9.99E99;
        end
    end
    [ GCMS_MIN_SCL, INDEX ] = min(GCMS_SCALE_FAC);
% Build Sensor Matrix Based on Scale Factors
k = 1;
for c = 1:length(GCMS_SCALE_FAC)
    if GCMS_SCALE_FAC(c) == GCMS_MIN_SCL
        for j = 1:size(GCMS_MATRIX2,2)
            GCMS_MATRIX3{k, j} = GCMS_MATRIX2{c,j};
        end
        k = k + 1;
    end
end
end

% Copies GCMS_MATRIX2 to GCMS_MATRIX3 if all Sensors Eliminated based on Mass Volume Scale Factor
if exist('GCMS_MATRIX3') == 0
    GCMS_MATRIX3 = GCMS_MATRIX2;
end
end
if exist('GCMS_MATRIX3') == 0
    GCMS_MATRIX3 = GCMS_MATRIX;
end

% Save Density Sensor Properties to Data File
save('GCMS_SENSOR_Data.mat');
save('GCMS_SENSOR_Final.mat', 'GCMS_MATRIX3', 'GCMSUnitLabelStr', 'GCMS_UNIT');

%-----------------------------------------------------
% *** PRINT SUMMARY Output Section for 'Print' = 'Y'
if strcmpi(Print, 'Y') == 1
    fprintf('

*** GCMS SENSOR SUMMARY RESULTS ***

*** INPUTS ***
GCMS Operational Type: %s', GCMS_TYPE);
GCMS Operational Range: %s', GCMS_RANGE);
Unit System: %s', UNITS);

GCMS Sensor Properties
Sensor Properties
Sensor Properties
GCMS Sensor Type: %s', GCMS_MATRIX3{1,1});
Sensing Mass/Charge Range - Low: %10.2f %s', GCMS_MATRIX3{1,2}, GCMSUnitLabelStr{2});
High: %10.2f %s', GCMS_MATRIX3{1,3}, GCMSUnitLabelStr{3});
Mass Filter Type: %s', GCMS_MATRIX3{1,4});
Detector Operational Type: %s', GCMS_MATRIX3{1,5});
for r = 1:length(GCMS_MATRIX3{1,5})
    fprintf(' %s', 'GCMS_MATRIX3{1,5}[r]);

%-----------------------------------------------------
fprintf('
 Dynamic Mass-Charge Scanning Range: %10.2f %s', GCMS_MATRIX3{1,6}, GCMSUnitLabelStr{6});
fprintf('
 Scanning Resolution: %10.2f %s', GCMS_MATRIX3{1,7}, GCMSUnitLabelStr{7});
for r = 1:length(GCMS_MATRIX3{1,6})
fprintf(' %5.2f, GCMS_MATRIX3 {1, 8 } (r) ) ,•
fprintf(' %s', GCMSUnitLabelStr{8} ) ;
end
fprintf('
 Minimum Detected Partial Pressure: %10.2f %s', GCMS_MATRIX3{1,9}, GCMSUnitLabelStr{9});
for r = 1:length(GCMS_MATRIX3{1,9})
fprintf (' %5.2f, GCMS_MATRIX3 {1, 10} (r) ) ;
end
fprintf(' %s', GCMSUnitLabelStr{10} );
fprintf('
 Operating Pressure: %10.2f %s', GCMS_MATRIX3{1,10}, GCMSUnitLabelStr{10});
for r = 1:length(GCMS_MATRIX3{1,10})
fprintf (' %5.2f, GCMS_MATRIX3 {1, 11} (r) ) ;
end
fprintf(' %s', GCMSUnitLabelStr{11} );
fprintf('
 Maximum Operating Temperature: %10.2f %s', GCMS_MATRIX3{1,11}, GCMSUnitLabelStr{11});
fprintf('
 Bakeout Temperature: %10.2f %s', GCMS_MATRIX3{1,12}, GCMSUnitLabelStr{12});
fprintf('
 Charge Field Range - Low: %10.2f %s', GCMS_MATRIX3{1,16}, GCMSUnitLabelStr{16});
fprintf('
 High: %10.2f %s', GCMS_MATRIX3{1,17}, GCMSUnitLabelStr{17});
fprintf('
 Electron Source Voltage - Low: %10.2f %s', GCMS_MATRIX3{1,18}, GCMSUnitLabelStr{18});
fprintf('
 High: %10.2f %s', GCMS_MATRIX3{1,19}, GCMSUnitLabelStr{19});
for r = 1:length(GCMS_MATRIX3{1,20})
fprintf (' %5.2f, GCMS_MATRIX3 {1, 21} (r) ) ;
end
fprintf(' %s', GCMSUnitLabelStr{21});
fprintf('
 Focus Voltage - Low: %10.2f %s', GCMS_MATRIX3{1,22}, GCMSUnitLabelStr{22});
fprintf('
 High: %10.2f %s', GCMS_MATRIX3{1,23}, GCMSUnitLabelStr{23});
fprintf('
 Warm Up Period Mass Stability: %10.2f %s', GCMS_MATRIX3{1,24}, GCMSUnitLabelStr{24});
fprintf('
 Time: %10.2f %s', GCMS_MATRIX3{1,25}, GCMSUnitLabelStr{25});
fprintf('
 Sensor Return Data Rate: %10.2f %s', GCMS_MATRIX3{1,26}, GCMSUnitLabelStr{26});
fprintf('

 Physical Properties');
fprintf('
 Sensor Mass: %10.2f %s', GCMS_MATRIX3{1,33}, GCMSUnitLabelStr{33});
fprintf('
 Sensor Dimensions - Length: %10.2f %s', GCMS_MATRIX3{1,34}, GCMSUnitLabelStr{34});
fprintf('
 Width: %10.2f %s', GCMS_MATRIX3{1,35}, GCMSUnitLabelStr{35});
fprintf('
 Height: %10.2f %s', GCMS_MATRIX3{1,36}, GCMSUnitLabelStr{36});
fprintf('
 Power Source Voltage: %10.2f %s', GCMS_MATRIX3{1,37}, GCMSUnitLabelStr{37});
fprintf('
 Power Source Voltage Type: %10.2f %s', GCMS_MATRIX3{1,38}, GCMSUnitLabelStr{38});
fprintf('
 Power Source Requirements - Typical: %10.2f %s', GCMS_MATRIX3{1,39}, GCMSUnitLabelStr{39});
fprintf('
 Average: %10.2f %s', GCMS_MATRIX3{1,40}, GCMSUnitLabelStr{40});
fprintf('
 Peak: %10.2f %s', GCMS_MATRIX3{1,41}, GCMSUnitLabelStr{41});
fprintf('Sensor Comments:

for n = 1:length(GCMS_MATRIX3{1,38})
    fprintf(C
        %s', GCMS_MATRIX3{1,38}{n} )
    end
end

%*** Writes a text file summary of the Planetary Properties
Fout = fopen('GCMS_SENSOR_Summary.txt', 'w+');

fprintf(Fout, 'GCMS SENSOR SUMMARY RESULTS ***
*** INPUTS ***
GCMS Operational Type: %s', GCMS_TYPE);
fprintf(Fout, ' GCMS Operational Range: %s', GCMS_RANGE);
fprintf(Fout, ' Unit System: %s', UNITS);

% GCMS Sensor Properties
fprintf(Fout, 'Sensor Properties ***
GCMS Sensor Type:
Sensing Mass/Charge Range - Low: %10.2f %s', GCMS_MATRIX3{1,2}, GCMSUnitLabelStr{2} );
fprintf(Fout, ' High: %10.2f %s', GCMS_MATRIX3{1,3}, GCMSUnitLabelStr{3} );
fprintf(Fout, ' Mass Filter Type: %s', GCMS_MATRIX3{1,4} );
fprintf(Fout, ' Detector Operational Type: %s', GCMSUnitLabelStr{4} );
for r = 1:length(GCMS_MATRIX3{1,5})
    fprintf(Fout, ' %s', GCMS_MATRIX3{1,5}{r} );
end
fprintf(Fout, ' Dynamic Mass-Charge Scanning Range: %10d %s', GCMS_MATRIX3{1,6}, GCMSUnitLabelStr{6} );
fprintf(Fout, ' Resolution: %10.2f %s', GCMS_MATRIX3{1,7}, GCMSUnitLabelStr{7} );
for r = 1:length(GCMS_MATRIX3{1,8})
    fprintf(Fout, ' %5.2f, %s', GCMS_MATRIX3{1,8}{r} );
end
fprintf(Fout, ' Minimum Detected Partial Pressure: %10.2f %s', GCMS_MATRIX3{1,9}, GCMSUnitLabelStr{9} );
fprintf(Fout, ' Operating Pressure: %10.2f %s', GCMS_MATRIX3{1,10}, GCMSUnitLabelStr{10} );
for r = 1:length(GCMS_MATRIX3{1,10})
    fprintf(Fout, ' %5.2f, %s', GCMS_MATRIX3{1,10}{r} );
end
fprintf(Fout, ' Maximum Operating Temperature: %10.2f %s', GCMS_MATRIX3{1,11}, GCMSUnitLabelStr{11} );
fprintf(Fout, ' Bakeout Temperature: %10.2f %s', GCMS_MATRIX3{1,12}, GCMSUnitLabelStr{12} );
fprintf(Fout, ' Number of Ion Sources: %10d %s', GCMS_MATRIX3{1,13}, GCMSUnitLabelStr{13} );
fprintf(Fout, ' Construction Material: %s', GCMS_MATRIX3{1,14} );
fprintf(Fout, ' Filament Material: %s', GCMS_MATRIX3{1,15} );
fprintf(Fout, ' Charge Field Range - Low: %10.2f %s', GCMS_MATRIX3{1,16}, GCMSUnitLabelStr{16} );
fprintf(Fout, '\n High: %10.2f %s', GCMS_MATRIX3{1,17}, GCMSUnitLabelStr{17});
fprintf(Fout, '\n Electron Source Voltage - Low: %10.2f %s', GCMS_MATRIX3{1,18}, GCMSUnitLabelStr{18});
fprintf(Fout, '\n High: %10.2f %s', GCMS_MATRIX3{1,19}, GCMSUnitLabelStr{19});
fprintf(Fout, '\n Ion Energy: %s', GCMS_MATRIX3{1,20});
for r = 1:length(GCMS_MATRIX3{1,20})
    fprintf(Fout, ' %5.2f, GCMS_MATRIX3{l,20}{r} );
end
fprintf(Fout, '\n Focus Voltage - Low: %10.2f %s', GCMS_MATRIX3{1,21}, GCMSUnitLabelStr{21});
fprintf(Fout, '\n High: %10.2f %s', GCMS_MATRIX3{1,22}, GCMSUnitLabelStr{22});
fprintf(Fout, '\n Electron Source Current - Low: %10.2f %s', GCMS_MATRIX3{1,23}, GCMSUnitLabelStr{23});
fprintf(Fout, '\n High: %10.2f %s', GCMS_MATRIX3{1,24}, GCMSUnitLabelStr{24});
fprintf(Fout, '\n Warm Up Period Mass Stability: %10.2f %s', GCMS_MATRIX3{1,25}, GCMSUnitLabelStr{25});
fprintf(Fout, '\n Time: %10.2f %s', GCMS_MATRIX3{1,26}, GCMSUnitLabelStr{26});
fprintf(Fout, '\n Sensor Return Data Rate: %10.2f %s', GCMS_MATRIX3{1,27}, GCMSUnitLabelStr{27});
fprintf(Fout, '\n\n Physical Properties');
fprintf(Fout, '\n Sensor Mass: %10.2f %s', GCMS_MATRIX3{1,33}, GCMSUnitLabelStr{33});
fprintf(Fout, '\n Sensor Dimensions - Length: %10.2f %s', GCMS_MATRIX3{1,25}, GCMSUnitLabelStr{25});
fprintf(Fout, '\n Width: %10.2f %s', GCMS_MATRIX3{1,26}, GCMSUnitLabelStr{26});
fprintf(Fout, '\n Height: %10.2f %s', GCMS_MATRIX3{1,27}, GCMSUnitLabelStr{27});
fprintf(Fout, '\n Power Source Voltage: %10.2f %s', GCMS_MATRIX3{1,30}, GCMSUnitLabelStr{30});
fprintf(Fout, '\n Power Source Voltage Type: %s', GCMS_MATRIX3{1,31} );
fprintf(Fout, '\n Power Source Current: %10.2f %s', GCMS_MATRIX3{1,32}, GCMSUnitLabelStr{32});
fprintf(Fout, '\n Power Requirements - Typical: %10.2f %s', GCMS_MATRIX3{1,35}, GCMSUnitLabelStr{35});
fprintf(Fout, '\n Average: %10.2f %s', GCMS_MATRIX3{1,36}, GCMSUnitLabelStr{36});
fprintf(Fout, '\n Peak: %10.2f %s', GCMS_MATRIX3{1,37}, GCMSUnitLabelStr{37});

for n = 1:length(GCMS_MATRIX3{1,38})
    fprintf(Fout, '\n %s', GCMS_MATRIX3{1,38}{n} );
end
fclose(Fout);

end

D10. Humidity Sensors
% function HUMIDITY_SENSORS(UNITS,Print)
% Humidity Sensor Database
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
% This file contains Humidity Sensor values for the ISSPO program.
% Data here is loaded into the main program to determine the proper sensor
% to use for Missions based on the sensor requirements.
% Values are assigned to a variable name and saved to a MatLAB '.mat' file
% and loaded when calling the ISSPO program.

% Some of the values are based on the Unit system chosen and are loaded
% in separate sections.

% References
% http://content.honeywell.com/sensing/prodinfo/humiditymoisture/

function HUMIDITY_SENSORS(UNITS,Print)

% Load Constants values into Local Program
load('Constants_DB.mat');
load('Planet_Data.mat');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Humidity Sensor Data Base Created Based on un Unit System
%% 2 - D Matrix HUMIDITYSENSOR_DB(H_Type, Property)
if strcmpi(UNITS,'SI') == 1
    % Properties 
    % Units
    % Stability Temp Range
    % Humidity Range
    % Capacitance at 55% RH
    % Sensitivity
    % Hysteresis
    % Response Time
    % RH/yr
    % Min C Max
    % kHz
    % Max
    % mm
    % hr
    % RH
    % %RH/%RH
    % sen
    % nity
    % ysersis
    % Response Time
    % HumUnitLabelStr = 
    % in
    % 'N/A'
    % 'RH'
    % 'RF'
    % 'pF'
    % 'pF/RH'
    % '%RH/yr'
    % 'deg C'
    % 'kHz'
    % 'in'
    % 'in'
    % 'oz'

end

if strcmpi(UNITS,'BRITISH') == 1
    % Properties 
    % Units
    % Stability Temp Range
    % Humidity Range
    % Capacitance at 55% RH
    % Sensitivity
    % Hysteresis
    % Response Time
    % RH/yr
    % Min C Max
    % kHz
    % Max
    % mm
    % hr
    % RH
    % %RH/%RH
    % sen
    % nity
    % ysersis
    % Response Time
    % HumUnitLabelStr = 
    % in
    % 'N/A'
    % 'RH'
    % 'RF'
    % 'pF'
    % 'pF/RH'
    % '%RH/yr'
    % 'deg F'
    % 'kHz'
    % 'in'
    % 'in'
    % 'oz'
end
end

% Find the Max Operational Range
for h = 1:size(HUMIDITYSENSOR_DB,1)
    DELTA(h) = HUMIDITYSENSOR_DB{h,3} - HUMIDITYSENSOR_DB{h,2};
end

% Find the Maximum Humidity Range of the Sensor
MAX_DELTA = max(DELTA);

% For Multiple Sensors with the Same Range Reduce Sensor Matrix to Those
% with the Greatest Range
k = 1;
for c = 1:length(DELTA)
    if DELTA(c) == MAX_DELTA
        for j = 1:size(HUMIDITYSENSOR_DB,2)
            H_MATRIX{k,j} = HUMIDITYSENSOR_DB{c,j};
        end
        k = k + 1;
    end
end

% Run Case if More than One Sensor is Viable
if size(H_MATRIX,1) > 1
    % Sort Sensors by Max Operational Range
    for h = 1:size(H_MATRIX,1)
        H_TEMP(h) = H_MATRIX{h,12} - H_MATRIX{h,11};
    end
    [Temp, H_Loc ] = max(H_TEMP);
    % Build Sensor Matrix Based on the Temp Range
    k = 1;
    for c = 1:length(H_TEMP)
        if H_TEMP(c) == Temp
            for j = 1:size(HUMIDITYSENSOR_DB,2)
                H_MATRIX2{h,j} = H_MATRIX{c,j};
            end
        end
        k = k + 1;
    end
    clear H_MATRIX;
end
H_MATRIX = H_MATRIX2;

% Sort Sensors by Volume if there is more than one sensor
if size(H_MATRIX,1) > 1
    % Sort Sensors by Min Volume
    for h = 1:size(H_MATRIX,1)
        H_VOL(h) = H_MATRIX{h,15} * H_MATRIX{h,16} * H_MATRIX{h,17};
    end
[Vol, H_Loc ] = min(H_Vol);

% Build Sensor Matrix Based on Volumes
k = 1;
for c = 1:length(H_Vol)
    if H_Vol(c) == Vol
        for j = 1:size(HUMIDITYSENSOR_DB,2)
            H_MATRIX3{k,j} = H_MATRIX{c,j};
        end
    end
    k = k + 1;
end

clear H_MATRIX;
H_MATRIX = H_MATRIX3;

H20_FLG = 0;

% Search Planetary Atmosphere for Water Component
for w = 1:2:length(PlanAtm_Prop)
    if strcmpi(PlanAtm_Prop{1,w},'H2O') == 1
        PlanAtm_Prop{1,w};
        H20_FLG = 1;
        break;
    end
end

if H20_FLG == 0;
    if strcmpi(Print,'Y') == 1
        fprintf('

 *** WARNING IN HUMIDITY SENSOR PROGRAM ***

 Humidity Sensor Not Required In Sensor Package Design!!

 Water Component not found in Selected Planetary Atmosphere.

 ***

end

H_MATRIX{1,1} = 'N/A';
for w = 2:size(HUMIDITYSENSOR_DB,2)
    H_MATRIX{1,w} = 0.0;
end

end

% Save Humidity Sensor Properties to Data File
save('HUMIDITY_SENSOR_Data.mat');
save('HUMIDITY_SENSOR_Final.mat','H_MATRIX','HumUnitLabelStr');
fprintf('

*** HUMIDITY SENSOR SUMMARY RESULTS ***
');
fprintf('

*** INPUTS ***
');
fprintf('

Unit System: %s', 'UNITS');

if H2O_FLG == 0;
    fprintf('

*** WARNING IN HUMIDITY SENSOR PROGRAM ***
');
    fprintf('

Humidity Sensor Not Required In Sensor Package Design!!!
');
    fprintf('

Water Component not found in Selected Planetary Atmosphere.
');
end

fprintf('

% Humidity Sensor Properties
');
fprintf('

Sensor Properties ***

Humidity Sensor Type: %s', 'H_MATRIX[1,1]');
fprintf('

Humidity Sensing Range - Low: %9.3f %s', 'H_MATRIX[1,2], HumUnitLabelStr[2]');
fprintf('

Humidity Sensing Range - High: %9.3f %s', 'H_MATRIX[1,3], HumUnitLabelStr[3]');
fprintf('

Capacitance at 55% RH - Low: %9.3f %s', 'H_MATRIX[1,4], HumUnitLabelStr[4]');
fprintf('

Typical: %9.3f %s', 'H_MATRIX[1,5], HumUnitLabelStr[5]');
fprintf('

Max: %9.3f %s', 'H_MATRIX[1,6], HumUnitLabelStr[6]');
fprintf('

Sensor Sensivity: %9.3f %s', 'H_MATRIX[1,7], HumUnitLabelStr[7]');
fprintf('

Sensor Hysteresis: %9.3f %s', 'H_MATRIX[1,8], HumUnitLabelStr[8]');
fprintf('

Response Time: %9.3f %s', 'H_MATRIX[1,9], HumUnitLabelStr[9]');
fprintf('

Sensor Stability: %9.3f %s', 'H_MATRIX[1,10], HumUnitLabelStr[10]');

% Environmental Properties

Environmental Properties

Temperature Range - Low: %9.3f %s', 'H_MATRIX[1,11], HumUnitLabelStr[11]');
fprintf('

Temperature Range - High: %9.3f %s', 'H_MATRIX[1,12], HumUnitLabelStr[12]');
fprintf('

Sensing Frequency Range - Min: %9.3f %s', 'H_MATRIX[1,13], HumUnitLabelStr[13]');
fprintf('

Max: %9.3f %s', 'H_MATRIX[1,14], HumUnitLabelStr[14]');

Physical Properties

Sensor Dimensions - Length: %9.3f %s', 'H_MATRIX[1,15], HumUnitLabelStr[15]');
fprintf('

Width: %9.3f %s', 'H_MATRIX[1,16], HumUnitLabelStr[16]');
fprintf('

Height: %9.3f %s', 'H_MATRIX[1,17], HumUnitLabelStr[17]');
fprintf('

Sensor Mass: %9.3f %s', 'H_MATRIX[1,18], HumUnitLabelStr[18]');
end

%******************************************************************************

% Writes a text file summary of the Planetary Properties

Fout = fopen('HUMIDITY_SENSOR_Summary.txt','w+');

fprintf(Fout, '

*** HUMIDITY SENSOR SUMMARY RESULTS ***
');
fprintf(Fout, '

*** INPUTS ***
');
fprintf(Fout, '

Unit System: %s', 'UNITS');

if H2O_FLG == 0;
    fprintf(Fout, '

*** WARNING IN HUMIDITY SENSOR PROGRAM ***
');
end
fprintf(Fout, '\n Humidity Sensor Not Required in Sensor Package Design!!');
fprintf(Fout, '\n Water Component not found in Selected Planetary Atmosphere.');
end

fprintf(Fout, '\n
% Humidity Sensor Properties
fprintf(Fout, '\n Humidity Sensor Type: %s', H_MATRIX{1,1});
fprintf(Fout, '\n Humidity Sensing Range - Low: %9.3f %s', H_MATRIX{1,2}, HumUnitLabelStr{2});
fprintf(Fout, '\n Humidity Sensing Range - High: %9.3f %s', H_MATRIX{1,3}, HumUnitLabelStr{3});
fprintf(Fout, '\n Capacitance at 55% RH - Low: %9.3f %s', H_MATRIX{1,4}, HumUnitLabelStr{4});
fprintf(Fout, '\n Typical: %9.3f %s', H_MATRIX{1,5}, HumUnitLabelStr{5});
fprintf(Fout, '\n Max: %9.3f %s', H_MATRIX{1,6}, HumUnitLabelStr{6});
fprintf(Fout, '\n Sensor Sensitivity: %9.3f %s', H_MATRIX{1,7}, HumUnitLabelStr{7});
fprintf(Fout, '\n Sensor Hysteresis: %9.3f %s', H_MATRIX{1,8}, HumUnitLabelStr{8});
fprintf(Fout, '\n Response Time: %9.3f %s', H_MATRIX{1,9}, HumUnitLabelStr{9});
fprintf(Fout, '\n Sensor Stability: %9.3f %s', H_MATRIX{1,10}, HumUnitLabelStr{10});

% Environmental Properties
fprintf(Fout, '\n Temperature Range - Low: %9.3f %s', H_MATRIX{1,11}, HumUnitLabelStr{11});
fprintf(Fout, '\n High: %9.3f %s', H_MATRIX{1,12}, HumUnitLabelStr{12});
fprintf(Fout, '\n Sensing Frequency Range - Min: %9.3f %s', H_MATRIX{1,13}, HumUnitLabelStr{13});
fprintf(Fout, '\n Max: %9.3f %s', H_MATRIX{1,14}, HumUnitLabelStr{14});

% Physical Properties
fprintf(Fout, '\n Sensor Dimensions - Length: %9.3f %s', H_MATRIX{1,15}, HumUnitLabelStr{15});
fprintf(Fout, '\n Width: %9.3f %s', H_MATRIX{1,16}, HumUnitLabelStr{16});
fprintf(Fout, '\n Height: %9.3f %s', H_MATRIX{1,17}, HumUnitLabelStr{17});
fprintf(Fout, '\n Sensor Mass: %9.3f %s', H_MATRIX{1,18}, HumUnitLabelStr{18});
fclose(Fout);

D11. Imaging Sensors

function IMAGING_SENSORS(SYS_TYPE, RESOLUTION, IMG_TYPE, UNITS, Print)

% Imaging Sensor Database
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
% This file contains Imaging Sensor values for the ISSPO program.
% Data here is loaded into the main program to determine the proper sensor
% to use for Missions based on the sensor requirements.
Values are assigned to a variable name and saved to a MatLAB '.mat' file and loaded when calling the ISSPO program.

Some of the values are based on the Unit system chosen and are loaded in separate sections.

References
http://www.fairchildimaging.com/applications/aerospace.htm
http://www.fairchildimaging.com/library/

function IMAGING_SENSORS(SYS_TYPE,RESOLUTION,IMG_TYPE,UNITS,Print)

% Load Constants values into Local Program
load('Constants_DB.mat' ) ;
load('Planet_Data.mat') ;

% Define Local Variables
RESOLUTION = OPT_DATA{2} ;
IMG_TYPE = OPT_DATA{3} ;
IMG_TYPE = cellstr(IMG_TYPE);

% IMAGING Sensor Data Base Created Based on un Unit System
% 2 - D Matrix IMAGINGSENSOR_DB(IS_Type, Property)

if strcmpi(UNITS, 'SI' ) ^= 1

% Properties [ SENSOR IMAGE AREA READ NOISE FULL WELL CAPACITY IMAGE TYPE GAIN LINEARITY ARRAY DIMENSIONS ]
PIXEL SIZE Width Units length mm Width @ 1MHz e- @ 250kHz Pixel ke- Register e-/ADU Length % # pixels
13 61.44 13 'Condor 486:90' 'CAMERA' 800 1.5 1.0 4096 4096
15 92.16 15 'Condor 486:135' 'CAMERA' 800 1.5 1.0 4096 4096
15 135.10 15 'Harrier 447' 'CAMERA' 800 1.5 1.0 4096 4096
15 31.00 15 'Peregrine 3041' 'CAMERA' 800 1.5 1.0 4096 4096
15 30.72 15 'Peregrine 486' 'CAMERA' 800 1.5 1.0 4096 4096
15 61.40 15 'Peregrine 486SX' 'CAMERA' 800 1.5 1.0 4096 4096
15 61.40 15 'Macrofire - MC' 'CAMERA' 800 1.5 1.0 4096 4096
7.4 15.20 15 'Macrofire-MC 1600x1200P' 'CAMERA' 800 1.5 1.0 4096 4096
7.4 11.84 15 'QuantumFire XI' 'CAMERA' 800 1.5 1.0 4096 4096
7.4 15.20 15 'CAMERA' 800 1.5 1.0 4096 4096
30.72

30.72

61.20

15

15

15

0

0

0

0

7

0

0

0

{
Low deg C High
deg C
fps
Length
Width
{
15.0
30.0
-60.0
0.55 ] [ 0.15 0.52 0.85 1.8 ]

'mm'

0
'MICRO'}
0
'MICRO'}
3.0
'MICRO'}
0
'MICRO'}
0
'MICRO'}
1.0
'MICRO'}
0
'MICRO'}
0
'MICRO'}
0

2.0

}

V TYPE

1.6 ]
0
;

'e-/ADU'

2.5
0

[4

2.1

2.1

2.1

2.5
0

[4

2.1

2.1

[ 0 ]

2.5
0

[4

[ 4 ]
;
%4
[ 2 1 ]
0

2.5
0

2.5
0

MHz

CURRENT

[4

0

512

256

4096

2048

2048

1024

1024

2048

582

1024

READOUT

•ke- '

%20

%19

%18

%17

%16

%15

%14

%13

N/A
mA
mW
' 16-bit'
[4
0
'N/A'

VOLTAGE

%11
%12

'ke- •

;
512
;
1024
;
2048
;
2048
;
4096
;l
;l
};

2048

;
752

1392

ADC DYNAMIC RANGE

' e-' ' e - '

WEIGHT

mm'

'NIR'

'NIR'

'NIR'

'NIR'

'NIR'

'NIR'

'NIR'

'NIR'

}

N/A
Height
kg
N/A
Thermoelectric w/Chiiled Water'
152
145
266
7.7

COOLING METHOD
DIMENSIONS (mm)

'micro-m'

0

{ 'UV' •VISUAL' 'NIR'
0
'VISUAL'
{
0
0
{ 'X-RAY' 'UV' 'VISUAL
750
0
{ ' X - R A Y0 ' ' U V ' 'VISUAL
0
{ 'X-RAY' 'UV' 'VISUAL
500
0
'UV' 'VISUAL
{ 'X-RAY'
200
0
f
I ' X - R A Y ' ' U V ' 'VISUAL'
100
2.5
1
{ 'X-RAY' 'UV 'VISUAL'
750
0
{ ' X - RAY0 ' ' UV ''VISUAL
0
{ 'X-RAY'0 ' U V 'VISUAL
0

'# Pix
eU '

'CAMERA'
18
' CAMERA'
0
1
ARRAY'
0
1
ARRAY'
85
'ARRAY'
0
'ARRAY'
0
'ARRAY'
80
'ARRAY'
0
'LINEAR'
0
'LINEAR'
0

'# Pixel s'

0

0

[
OPERATING TEMP
COOLING TEMP
TIME
FRAME RATE

'N/A' 'N/A' 'N/A•

'Pixelfly Q E '
0
6.6C
'Sony DXC-990P'
0
4.80
CCD 3041'
30.72
7
'CCD 412'
15.36
45
'CCD 424'
0
21.50
CCD 442A'
30.72
0
'CCD 447'
30.72
12
'CCD 485'
12
61.20
'CCD 111'
0
3.33
'CCD 153A'
6.66
0

15.0
30.0
-60.0
'Thermoelectric w/Cryo Cooler'
'16-bit'
[ 6.5 1.90 1.18 0.55 ] [ 0.15 0.52 0.85 1.8 ]
165
131
395
16.4
0
'N/A'
%2
1
15.0
30.0
-60.0
'Thermoelectric w/Cryo Cooler
• 16-bit'
1.6 ]
[ 6.5 1.90 1.18 0.55 ] [ 0.15 0.52 0.85 1.8 ]
0
'N/A'
0
0
0
29.5
0
; %3
0
0
0.0
'N/A'
'14-bit'
[ 0 ]
[0 i
0
0
0
0
0
'N/A'
0
15.0
30.0
-25.0
'Thermoelectric w/Forced Air'
'16-bit'
[ 1.1 0.364 0.180]
[ 0.9 2.7 5.6 ]
131
112
180
4.5
0
•N/A'
0
%5
1
15.0
30.0
-60.0
'Thermoelectric w/Chilled Water
'16-bit'
1.6 ]
[ 6.5 1.90 1.18 0.55 ] [ 0.15 0.52 0.85 1.8 ]
152
131
239
9.3
0
'N/A'
0
; %6
15.0
30.0
-60.0
'Thermoelectric w/'Chilled Water'
'16-bit'
1.6 ]
[ 6.5 1.90 1.18 0.55 ] [ 0.15 0.52 0.85 1.8 ]
152
131
239
9.3
0
'N/A'
0
; %7
0
0
0.0
' N/A'
'12-bit'
[ 0 60 ]
[ 5 9 12 15 18 38 ]
134.6 134.6 58.4
0
'N/A'

% Properties
RATES
READOUT
POWER
]
%
Units
sec
IMAGINGSENSOR_DB2 =
1.6 ]
[ 6.5 1.90 1.18
0
; %1

ImgUnj tLabelStrl == {

0.013

21.50

21

13

15.36

15

0 .013

30.72

15

13

6.40

8.98

8

6.45


% Properties
% Units

<table>
<thead>
<tr>
<th>SENSOR IMAGING TYPE IMAGE AREA READ NOISE FULL WELL CAPACITY IMAGE TYPE</th>
<th>PIXEL SIZE</th>
<th>IMAGE</th>
<th>GAIN</th>
<th>LINEARITY</th>
<th>ARRAY DIMENSIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMAGINGSENSOR_DB1 = ['Condor 486:90', 'CAMERA']</td>
<td>2.419</td>
<td>'condor 486:90'</td>
<td>'camerA'</td>
<td>'x-ray'</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>2.419</td>
<td>'condor 486:113'</td>
<td>'camerA'</td>
<td>'x-ray'</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>3.628</td>
<td>'condor 486:200'</td>
<td>'camerA'</td>
<td>'x-ray'</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>5.319</td>
<td>'condor 486:200'</td>
<td>'camerA'</td>
<td>'x-ray'</td>
<td>10</td>
</tr>
<tr>
<td>Image Label</td>
<td>Camera Name</td>
<td>Sensor Type</td>
<td>Coolant Method</td>
<td>Cooling Temp</td>
<td>Resolution</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>----------------</td>
<td>---------------</td>
<td>------------</td>
</tr>
<tr>
<td>'Harrier 447'</td>
<td>'Peregrine 468'</td>
<td>'Micro'</td>
<td>'Thermoelectric w/Cryo Cooler'</td>
<td>-76.0</td>
<td>1200</td>
</tr>
<tr>
<td>'Peregrine 468'</td>
<td>'Micro'</td>
<td>'Thermoelectric w/Cryo Cooler'</td>
<td>-76.0</td>
<td>1200</td>
<td>6.5 fps</td>
</tr>
</tbody>
</table>

...Continued...
IMAGINGSENSOR_DB3[8,1] = {

'Scientific precision and accuracy';
'Software-controlled binning & windowing';
'Optimize speed versus resolution'}

IMAGINGSENSOR_DB3[9,1] = {

'Microscope Monochrome Camera';
'500% increase in Field-of-View';
'Progressive Scan Interline CCD Array';
'Optional Automated RGB 3-Pass Color Filter';
'Software image editing tools';
'IEEE 1394 FireWire 6-pin Computer Interface';
'F-Mount Optical Interface';
'Electronic Shutter: Integration time to 60 seconds';
'IS0 9001:2000 CERTIFIED'}

IMAGINGSENSOR_DB3[10,1] = {

'Microscope Monochrome Camera';
'500% increase in Field-of-View';
'Visual to Near Infra-Red Spectrum Range';
'Optional Automated RGB 3-Pass Color Filter';
'Software image editing tools';
'IEEE 1394 FireWire 6-pin Computer Interface';
'F-Mount Optical Interface';
'Electronic Shutter: Integration time to 20 minutes';
'IS0 9001:2000 CERTIFIED'}

IMAGINGSENSOR_DB3[11,1] = {

'Optional Color CCD Sensor';
'Readout noise: 7e^- rms typical';
'Spectral Range: 290 - 1100 nm';
'12 W Power Consumption';
'Software camware and software development kit included';
'High speed serial LVDS shielded ethernet patch cable RJ45 connector computer interface';
'Hardened against high magnetic fields'}

IMAGINGSENSOR_DB3[12,1] = {

'3-CCD color video camera';
'Bayonet mount adapts various high quality, professional lenses';
'850 TV line Horizontal resolution';
'Approx. 6 W Power Consumption';
'6 Automatic Exposure Modes';
'+30 dB sensitivity gain for low light conditions';
'RGB, component, Y/C and composite video outputs, RS-232C controllable';
'Applications for microscopy, industrial, inspection and remote camera systems'}

IMAGINGSENSOR_DB3[13,1] = {

'Back-illuminated CCD Array';
'Highest quantum efficiency';
'Multiple output image formats';
'Four-port readout';
'Optimal design for speed and sensitivity';
'Dual Digitizer';
'Separate, optimized readout channels for lowest noise and highest speed';
'Space Qualified - Production Company';
'Software Gain & Binning'}
IMAGINGSENSOR_DB3{14,1} = {'Scientific precision and accuracy' };  
  'Photosite CCD Array' ;  
  'Frame Transfer Architecture' ;  
  'Multiple output image formats';  
  'Four-port readout' ;  
  'Space Qualified - Production Company' ;  
  'Software Gain & Binning' ;  
  'Scientific precision and accuracy' );

IMAGINGSENSOR_DB3{15,1} = {'Split Frame Transfer CCD Array';  
  '2k - 1024 x 530 pixel Storage Areas' ;  
  'Rapid image capture capability' ;  
  'Multiple output image formats';  
  'Four-port readout' ;  
  'Low Read Noise' ;  
  'Space Qualified - Deep Impact Mission' ;  
  'Software Gain & Binning' ;  
  'Scientific precision and accuracy' );

IMAGINGSENSOR_DB3{16,1} = {'Photosite CCD Array' ;  
  'Multi Pinned Phase Option' ;  
  'High Dynamic Range' ;  
  'Space Qualified - Production Company' ;  
  'Scientific precision and accuracy' );

IMAGINGSENSOR_DB3{17,1} = {'Full Frame CCD Array' ;  
  'Full frame 3-phase with MPP integration mode' ;  
  'Multi Pinned Phase Option' ;  
  'Stand Alone CCD or integrated with Camera' ;  
  '2 Low Noise Amplifiers' ;  
  '2 High Speed Amplifiers' ;  
  'Space Qualified - Production Company' ;  
  'Software Gain & Binning' );

IMAGINGSENSOR_DB3{18,1} = {'Full Frame CCD Array' ;  
  'Multi Pinned Phase Option' ;  
  'Low Read Noise, Wide Dynamic Range' ;  
  'Space Qualified - Production Company' ;  
  'Software Gain & Binning' );

IMAGINGSENSOR_DB3{19,1} = {'Line Scan Image Sensor' ;  
  'On-chip video and compensation amplifiers' ;  
  'Low Power Requirements' ;  
  'Low Noise Equivalent Exposure' ;  
  'Dimensionally Precise Photosite Spacing' ;  
  'Space Qualified - Production Company' );

IMAGINGSENSOR_DB3{20,1} = {'High Speed Linear Image Sensor' ;  
  'Enhanced spectral response' ;  
  'Low dark signal' ;  
  'High Responsivity' ;  
  'On-chip Clock Drivers' ;  
  'Dynamic Range Typical 5000:1' ;  
  'Dimensionally Precise Photosite Spacing' ;  
  'Space Qualified - Production Company' );

% Combine Data Arrays into a Single Data Matrix
IMAGINGSENSOR_DB = cat(2,IMAGINGSENSOR_DB1,IMAGINGSENSOR_DB2,IMAGINGSENSOR_DB3);
IMG.UnitLabelStr = cat(2, IMG.UnitLabelStr1, IMG.UnitLabelStr2);

IMG_MATRIX = IMAGINGSENSOR_DB;

% Select Sensors By System Type - 'CAMERA' or 'ARRAY' or 'LINEAR'

for typ = 1:size(IMG_MATRIX,1)
    if strcmpi(IMG_MATRIX{typ,2}, SYS_TYPE) == 1
        for s = 1:size(IMG_MATRIX,2)
            IMG_MATRIX{tmp,s} = IMG_MATRIX{typ,s};
            tmp = tmp + 1;
        end
    end
end

% Down Select Censors that Match Imaging Types

for loc = 1:length(IMG_TYPE)
    STR1 = IMG_TYPE{loc};
    if strcmpi(STR1, 'X-RAY') == 1
        c = 1;
        FLAG = 0;
        for img = 1:size(IMG_MATRIX2,1)
            for pos = 1:length(IMG_MATRIX2{img,3})
                STR2 = IMG_MATRIX2{img,3}{pos};
                if strcmpi(STR1, STR2) == 1
                    for s = 1:size(IMG_MATRIX2,2)
                        IMG_MATRIX_TMP{c,s} = IMG_MATRIX2{img,s};
                    end
                    FLAG = 1;
                end
            end
            if FLAG == 1
                c = c + 1;
            end
            FLAG = 0;
        end
        if exist('IMG_MATRIX_TMP') == 0
            if strcmpi(Print, 'Y') == 1
                fprintf('

*** ERROR: NULL DATABASE ***

No Sensor Exist in Database which Meet Image Type: %s', STR1);
                IMG_MATRIX2 =IMG_MATRIX2 ;
            end
        end
        clear IMG_MATRIX2;
        IMG_MATRIX2 = IMG_MATRIX_TMP;
        clear IMG_MATRIX_TMP;
    end
end

if strcmpi(STR1, 'VISUAL') == 1
    c = 1;
    FLAG = 0;
for img = 1:size(IMG_MATRIX2,1)
    for pos = 1:length(IMG_MATRIX2{img,3})
        STR2 = IMG_MATRIX2{img,3}{pos};
        if strcmpi(STR1, STR2) == 1
            for s = 1:size(IMG_MATRIX2,2)
                IMG_MATRIX_TMP{c,s} = IMG_MATRIX2{img,s};
            end
            FLAG = 1;
        end
        if FLAG == 1
            c = c + 1;
        end
    end
    if exist('IMG_MATRIX_TMP') == 0
        if strcmpi(Print, 'Y') == 1
            fprintf('

 *** ERROR: NULL DATABASE ***

 No Sensor Exist in Database which Meet Imaging Requirements!

 Image Type: %s', STR1);
        end
    end
    clear IMG_MATRIX2;
    IMG_MATRIX2 = IMG_MATRIX_TMP;
end
if strcmpi(STR1, 'UV') == 1
    c = 1;
    FLAG = 0;
    for img = 1:size(IMG_MATRIX2,1)
        for pos = 1:length(IMG_MATRIX2{img,3})
            STR2 = IMG_MATRIX2{img,3}{pos};
            if strcmpi(STR1, STR2) == 1
                for s = 1:size(IMG_MATRIX2,2)
                    IMG_MATRIX_TMP{c,s} = IMG_MATRIX2{img,s};
                end
                FLAG = 1;
            end
            if FLAG == 1
                c = c + 1;
            end
        end
        if exist('IMG_MATRIX_TMP') == 0
            if strcmpi(Print, 'Y') == 1
                fprintf('

 *** ERROR: NULL DATABASE ***

 No Sensor Exist in Database which Meet Imaging Requirements!

 Image Type: %s', STR1);
            end
        end
    end
end
IMG_MATRIX_TMP = IMG_MATRIX2;
end
end

clear IMG_MATRIX2;
IMG_MATRIX2 = IMG_MATRIX_TMP;
clear IMG_MATRIX_TMP;
end

if strcmpi(STR1, 'MICRO') == 1
  c = 1;
  FLAG = 0;
  for img = 1:size(IMG_MATRIX2,1)
    for pos = 1:length(IMG_MATRIX2{img,3})
      STR2 = IMG_MATRIX2{img,3}{pos};
      if strcmpi(STR1,STR2) == 1
        for s = 1:size(IMG_MATRIX2,2)
          IMG_MATRIX_TMP(c,s) = IMG_MATRIX2{img,s};
          FLAG = 1;
        end
      end
    end
    if FLAG == 1
      C = C + 1;
    end
    FLAG = 0;
  end
  if exist('IMG_MATRIX_TMP') == 0
    if strcmpi(Print,'Y') == 1
      fprintf('

 *** ERROR: NULL DATABASE ***

 No Sensor Exist in Database which Meet Imaging Requirements!

 Image Type: %s', STR1);
    IMG_MATRIX_TMP = IMG_MATRIX2;
  end
  end
  clear IMG_MATRIX2;
  IMG_MATRIX2 = IMG_MATRIX_TMP;
  clear IMG_MATRIX_TMP;
end

% Down Select from Multiple Sensor Options
if size(IMG_MATRIX2,1) > 1

  if strcmpi(RESOLUTION,'HIGH') == 1
    MPixels = 8;
    TOL_Low = 1;
    TOL_High = 100;
  end
  if strcmpi(RESOLUTION,'MEDIUM') == 1
    MPixels = 4;
  end
TOL_Low = 2;
TOL_High = 4;
end
if strcmpi(RESOLUTION, 'LOW') == 1
MPixels = 1;
TOL_Low = 1;
TOL_High = 0;
end
for img = 1:size(IMG_MATRIX2, 1)
% Convert Number of Pixels to Megapixel resolutions
IMG_SIZE(img) = ( (IMG_MATRIX2(img,4) * IMG_MATRIX2(img,5))/1024) / 1024;
end
% Sort Sensors by Resolution Scale
k = 1;
for img = 1:size(IMG_MATRIX2, 1)
if (IMG_SIZE(img) >= (MPixels-TOL_Low) && IMG_SIZE(img) <= (MPixels+TOL_High))
for j = 1:size(IMG_MATRIX2,2)
IMG_MATRIX3{k,j} = IMG_MATRIX2{img,j};
end
k = k + 1;
end
end
% Copies IMG_MATRIX2 to IMG_MATRIX3 if all Sensors Eliminated
if exist('IMG_MATRIX3') == 0
if strcmpi(Print, 'Y') == 1
fprintf('*** WARNING: Optical Sensors Eliminated based on Chosen Resolution!');
end
IMG_MATRIX3 = IMG_MATRIX2;
end
if size(IMG_MATRIX3, 1) > 1
% Create Mass / Volume Scaling Factor Optimization
for mv = 1:size(IMG_MATRIX3,1)
IMG_SCALE_FAC(mv) = IMG_MATRIX3{mv,23} * IMG_MATRIX3{mv,24} * IMG_MATRIX3{mv,25} * IMG_MATRIX3{mv,26};
if [IMG_MATRIX3{mv,23} == 0] || [IMG_MATRIX3{mv,24} == 0] || [IMG_MATRIX3{mv,25} == 0] || [IMG_MATRIX3{mv,26} == 0]
IMG_SCALE_FAC(mv) = 9.99E99;
end
end
[ IMG_MIN_SCL, INDEX ] = min(IMG_SCALE_FAC);
% Build Sensor Matrix Based on Scale Factors
k = 1;
for c = 1:length(IMG_SCALE_FAC)
if IMG_SCALE_FAC(c) == IMG_MIN_SCL
for j = 1:size(IMG_MATRIX3,2)
IMG_MATRIX4{k,j} = IMG_MATRIX3{c,j};
end
k = k + 1;
end
% Copies IMG_MATRIX3 to IMG_MATRIX4 if all Sensors Eliminated
if exist('IMG_MATRIX4') == 0
    IMG_MATRIX4 = IMG_MATRIX3;
end

if size(IMG_MATRIX4,1) > 1
    for we = 1:size(IMG_MATRIX4,1)
        Max_WE_Des(we) = IMG_MATRIX4{we,12};
    end

    Max_WE_Sens = max(Max_WE_Des);
    k = 1;
    for we = 1:size(IMG_MATRIX4,1)
        if IMG_MATRIX4{we,12} == Max_WE_Sens
            for j = 1:size(IMG_MATRIX4,2)
                IMG_MATRIX5{k, j} = IMG_MATRIX4{we,j};
            end
            k = k + 1;
        end
    end
end

% Copies IMG_MATRIX4 to IMG_MATRIX5 if all Sensors Eliminated
if exist('IMG_MATRIX5') == 0
    IMG_MATRIX5 = IMG_MATRIX4;
end

% Copies IMG_MATRIX2 to IMG_MATRIX5 if all Sensors Eliminated
end

% Save Density Sensor Properties to Data File
save('IMAGING_SENSOR_Data.mat');
save('IMAGING_SENSOR_Final.mat','IMG_MATRIX5','ImgUnitLabelStr');

% *** PRINT SUMMARY Output Section for 'Print' = 'Y'
if strcmpi(Print,'Y') == 1
    fprintf('

 *** IMAGING SENSOR SUMMARY RESULTS ***
');
    fprintf('

 *** INPUTS ***
');
fprintf('System Type: %s', SYSJTYPE);
fprintf('Resolution: %s', RESOLUTION);
fprintf('Image Type: %s', IMG_TYPE{r});
for r = 1:length(IMG_TYPE)
fprintf('%s', IMG_TYPE{r});
end
fprintf('Unit System: %s', UNITS);

* * * * * * * * * ***********************

/* Imaging Sensor Properties */
fprintf('Sensor Properties **');
fprintf('Imaging Sensor Type: %s', IMG_MATRIX{1,1});
fprintf('Imaging Spectrum: %s', IMG_MATRIX{1,3});
for r = 1:length(IMG_MATRIX{1,3})
fprintf('%s', IMG_MATRIX{1,3}{r});
end
fprintf('
Array Dimensions - Length: %10d %s', IMG_MATRIX{1,4}, IMGUnitLabelStr{4});
fprintf('
  Width: %10d %s', IMG_MATRIX{1,5}, IMGUnitLabelStr{5});
fprintf('
Sensor Pixel Size: %10.4f %s', IMG_MATRIX{1,6}, IMGUnitLabelStr{6});
fprintf('Imaging Array Size - Length: %10.3f %s', IMG_MATRIX{1,7}, IMGUnitLabelStr{7});
fprintf('
  Width: %10.3f %s', IMG_MATRIX{1,8}, IMGUnitLabelStr{8});
fprintf('
Read Noise at 1 MHz: %10d %s', IMG_MATRIX{1,9}, IMGUnitLabelStr{9});
fprintf('250 kHz: %10d %s', IMG_MATRIX{1,10}, IMGUnitLabelStr{10});
fprintf('
Full Well Capacity - Pixel: %10d %s', IMG_MATRIX{1,11}, IMGUnitLabelStr{11});
fprintf('Register: %10d %s', IMG_MATRIX{1,12}, IMGUnitLabelStr{12});
fprintf('Sensor Gain: %10.2f %s', IMG_MATRIX{1,13}, IMGUnitLabelStr{13});
fprintf('Sensor Linearity: %10.2f %s', IMG_MATRIX{1,14}, IMGUnitLabelStr{14});
fprintf('ADC Dynamic Range: %d %s', IMG_MATRIX{1,19}, IMGUnitLabelStr{19});
fprintf('Readout Rates: %s', IMG_MATRIX{1,20});
for l = 1:length(IMG_MATRIX{1,20})
fprintf('%7.4f', IMG_MATRIX{1,20}{l});
end
fprintf('Readout Time: %s', IMGUnitLabelStr{20});
fprintf('Readout Time: %s', IMGUnitLabelStr{21});
for l = 1:length(IMG_MATRIX{1,21})
fprintf('%7.4f', IMG_MATRIX{1,21}{l});
end
fprintf('Frame Rates: %s', IMGUnitLabelStr{21});
fprintf('Frame Rates: %s', IMGUnitLabelStr{22});
for l = 1:length(IMG_MATRIX{1,22})
fprintf('%7.4f', IMG_MATRIX{1,22}{l});
end
fprintf('Environment Properties');
fprintf('Operating Temperature Range - Low: %10.2f %s', IMG_MATRIX{1,15}, IMGUnitLabelStr{15});
fprintf('High: %10.2f %s', IMG_MATRIX{1,16}, IMGUnitLabelStr{16});
fprintf('Cooled Temperature: %10.2f %s', IMG_MATRIX{1,17}, IMGUnitLabelStr{17});
fprintf('Cooling Method: %s %s', IMG_MATRIX{1,18}, IMGUnitLabelStr{18});
fprintf('Physical Properties');
fprintf('Dimensions - Length: %10.2f %s', IMG_MATRIX{1,23}, IMGUnitLabelStr{23});
fprintf(’\n
Sensor Mass:
Width:  %10.2f %s’, IMG_MATRIX5{1,24}, ImgUnitLabelStr{24} );
fprintf(’\n Height:  %10.2f %s’, IMG_MATRIX5{1,25}, ImgUnitLabelStr{25} );
fprintf(’\n Sensor Mass:  %10.3f %s’, IMG_MATRIX5{1,26}, ImgUnitLabelStr{26} );

fprintf(’\n
功率要求：
输入电压：  %9.3f %s’, IMG_MATRIX5{l,27}, ImgUnitLabelStr{27} ) ;
fprintf(’\n 输入电压类型：  %9.3f %s’, IMG_MATRIX5{l,28}, ImgUnitLabelStr{28} ) ;
fprintf(’\n 输入安培数：  %9.3f %s’, IMG_MATRIX5{l,29}, ImgUnitLabelStr{29} ) ;

fprintf(’\n
Sensor Comments:
for r = 1:length(IMG_MATRIX5{1,31})
fprintf(’\n %s ’ , IMG_MATRIX5{1,31}{r} ) ;
end

echo

%*******************************************
*****************************
*** 写一个总结文件的行星属性
Fout = fopen(’IMAGING_SENSOR_Summary.txt’, ’w+’);

fprintf(Fout, ’

*** IMAGING SENSOR SUMMARY ***’);
fprintf(Fout, ’

*** INPUTS ***’);
fprintf(Fout, ’
 System Type: %s’;
fprintf(Fout, ’
 Resolution: %s’;
fprintf(Fout, ’
 Image Type: ’);
for r = 1:length(IMG_TYPE)
fprintf(Fout, ’ %s’, IMG_MATRIX5{1,2} );
end
fprintf(Fout, ’
 Unit System: %s’);

% Imaging Sensor Properties
fprintf(Fout, ’

Sensor Properties ***’);
fprintf(Fout, ’
 Imaging Sensor Type: ’);
for r = 1:length(IMG_MATRIX5{1,3})
fprintf(Fout, ’ %s’, IMG_MATRIX5{1,3}{r} ) ;
end
fprintf(Fout, ’
 Array Dimensions - Length:  %10d %s’, IMG_MATRIX5{1,4}, ImgUnitLabelStr{4} );
fprintf(Fout, ’
 Width:  %10d %s’, IMG_MATRIX5{1,5}, ImgUnitLabelStr{5} );
fprintf(Fout, ’
 Sensor Pixel Size:  %10.3f %s’, IMG_MATRIX5{1,6}, ImgUnitLabelStr{6} );
fprintf(Fout, ’
 Imaging Array Size - Length:  %10.3f %s’, IMG_MATRIX5{1,7}, ImgUnitLabelStr{7} );
fprintf(Fout, ’
 Width:  %10.3f %s’, IMG_MATRIX5{1,8}, ImgUnitLabelStr{8} );
fprintf(Fout, ’
 Read Noise at 1 MHz:  %10d %s’, IMG_MATRIX5{1,9}, ImgUnitLabelStr{9} );
fprintf(Fout, ’
 Full Well Capacity - Pixel:  %10d %s’, IMG_MATRIX5{1,10}, ImgUnitLabelStr{10} );
fprintf(Fout, ’
 Register:  %10d %s’, IMG_MATRIX5{1,11}, ImgUnitLabelStr{11} );
fprintf(Fout, ’
 Sensor Gain:  %10.2f %s’, IMG_MATRIX5{1,12}, ImgUnitLabelStr{12} );
fprintf(Fout, ’
 Sensor Linearity:  %10.2f %s’, IMG_MATRIX5{1,13}, ImgUnitLabelStr{13} );
fprintf(Fout, ’
 ADC Dynamic Range:  %s %s’, IMG_MATRIX5{1,19}, ImgUnitLabelStr{19} );

fprintf(Fout, ’

** Sensor Properties **’);

fprintf(Fout, ’
 Imaging Sensor Type: ’);
for r = 1:length(IMG_MATRIX5{1,3})
fprintf(Fout, ’ %s’, IMG_MATRIX5{1,3}{r} ) ;
end
fprintf(Fout, ’
 Array Dimensions - Length:  %10d %s’, IMG_MATRIX5{1,4}, ImgUnitLabelStr{4} );
fprintf(Fout, ’
 Width:  %10d %s’, IMG_MATRIX5{1,5}, ImgUnitLabelStr{5} );
fprintf(Fout, ’
 Sensor Pixel Size:  %10.3f %s’, IMG_MATRIX5{1,6}, ImgUnitLabelStr{6} );
fprintf(Fout, ’
 Imaging Array Size - Length:  %10.3f %s’, IMG_MATRIX5{1,7}, ImgUnitLabelStr{7} );
fprintf(Fout, ’
 Width:  %10.3f %s’, IMG_MATRIX5{1,8}, ImgUnitLabelStr{8} );
fprintf(Fout, ’
 Read Noise at 1 MHz:  %10d %s’, IMG_MATRIX5{1,9}, ImgUnitLabelStr{9} );
fprintf(Fout, ’
 Full Well Capacity - Pixel:  %10d %s’, IMG_MATRIX5{1,10}, ImgUnitLabelStr{10} );
fprintf(Fout, ’
 Register:  %10d %s’, IMG_MATRIX5{1,11}, ImgUnitLabelStr{11} );
fprintf(Fout, ’
 Sensor Gain:  %10.2f %s’, IMG_MATRIX5{1,12}, ImgUnitLabelStr{12} );
fprintf(Fout, ’
 Sensor Linearity:  %10.2f %s’, IMG_MATRIX5{1,13}, ImgUnitLabelStr{13} );
fprintf(Fout, ’
 ADC Dynamic Range:  %s %s’, IMG_MATRIX5{1,19}, ImgUnitLabelStr{19} );
% Environmental Properties
fprintf(Fout, '

 Environmental Properties
 Operating Temperature Range - Low: %10.2f %s', IMG_MATRIX5{1,15}, ImgUnitLabelStr{15} );
fprintf(Fout, ' High: %10.2f %s', IMG_MATRIX5{1,16}, ImgUnitLabelStr{16} );
fprintf(Fout, ' Cooled Temperature: %10.2f %s', IMG_MATRIX5{1,17}, ImgUnitLabelStr{17} );
fprintf(Fout, ' Cooling Method: %s', IMG_MATRIX5{1,18}, ImgUnitLabelStr{18} );

% Physical Properties
fprintf(Fout, '

 Physical Properties
 Dimensions - Length: %10.2f %s', IMG_MATRIX5{1,23}, ImgUnitLabelStr{23} );
fprintf(Fout, ' Width: %10.2f %s', IMG_MATRIX5{1,24}, ImgUnitLabelStr{24} );
fprintf(Fout, ' Height: %10.2f %s', IMG_MATRIX5{1,25}, ImgUnitLabelStr{25} );
fprintf(Fout, ' Sensor Mass: %10.3f %s', IMG_MATRIX5{1,26}, ImgUnitLabelStr{26} );

% Power Requirements
fprintf(Fout, '

 Power Requirements
 Input Voltage: %9.3f %s', IMG_MATRIX5{1,27}, ImgUnitLabelStr{27} );
fprintf(Fout, ' Input Voltage Type: %9.3f %s', IMG_MATRIX5{1,28}, ImgUnitLabelStr{28} );
fprintf(Fout, ' Input Amperage: %9.3f %s', IMG_MATRIX5{1,29}, ImgUnitLabelStr{29} );
fprintf(Fout, ' Power Requirement: %9.3f %s', IMG_MATRIX5{1,30}, ImgUnitLabelStr{30} );

% Sensor Comments
for r = 1:length(IMG_MATRIX5{1,31})
    fprintf(Fout, '
 %s', IMG_MATRIX5{1,31}{r} );
end
fclose(Fout);
D12. Inclinometer Sensors

% function INCLINOMETER_SENSORS(INCR,UNITS,Print)
% Inclinometer Sensor Database
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
% This file contains Inclinometer Sensor values for the ISSPO program.
% Data here is loaded into the main program to determine the proper sensor
% to use for Missions based on the sensor requirements.
% Values are assigned to a variable name and saved to a MatLAB '.mat' file
% and loaded when calling the ISSPO program.
% Some of the values are based on the Unit system chosen and are loaded
% in separate sections.
% References
% http://www.riekerinc.com/Electronidnclinometers.htm
% http://www.jewellinstruments.com/inclinometer.htm
% http://www.spectronsensors.com/tilt.html

function INCLINOMETER_SENSORS(INCR,UNITS,Print)

% Load Constants values into Local Program
load('Constants_DB.mat');
load('Planet_Data.mat');

PLTEMP = Planet_Properties{7};

% Define Program Variables For Max Inclination Range
if strcmpi(INCR,'HIGH') == 1
    INC_ANGLE = 90;
    TOL_LOWER = 46;
end
if strcmpi(INCR,'MEDIUM') == 1
    INC_ANGLE = 45;
    TOL_LOWER = 30;
end
if strcmpi(INCR,'LOW') == 1
    INC_ANGLE = 30;
    TOL_LOWER = 0;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% INCLINOMETER Sensor Data Base Common Properties
%% 2 - D Matrix INCLINOMETERSSENSOR_DB(IS_Type, Property)

% Properties
% VOLTAGE_TYPE VOLTAGE RESPONSE_TIME BANDWIDTH OUTPUT_DATA RESOLUTION SENSITIVITY NON-LINEARIITY
% INCLINATION RANGE
<table>
<thead>
<tr>
<th>% Units</th>
<th>Volts</th>
<th>[ TYPE</th>
<th>Low</th>
<th>Deg</th>
<th>High</th>
<th>% N/A</th>
<th>deg</th>
<th>mV/deg</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>0.30</td>
<td>'H4A1-30'</td>
<td>-30.00</td>
<td>30.00</td>
<td>'Analog 0-5V'</td>
<td>0.02</td>
<td>67.70</td>
<td>0.50</td>
<td></td>
</tr>
<tr>
<td>'DC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ 8 30 ]</td>
<td>0.30</td>
<td>'H4A1-45'</td>
<td>-45.00</td>
<td>45.00</td>
<td>'Analog 0-5V'</td>
<td>0.03</td>
<td>44.40</td>
<td>0.50</td>
<td></td>
</tr>
<tr>
<td>'DC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ 8 30 ]</td>
<td>0.30</td>
<td>'H4A1-70'</td>
<td>-70.00</td>
<td>70.00</td>
<td>'Analog 0-5V'</td>
<td>0.04</td>
<td>28.60</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>'DC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ 12 30 ]</td>
<td>0.30</td>
<td>'H5A1-30'</td>
<td>-30.00</td>
<td>30.00</td>
<td>'Analog 0-5V, 0-10V'</td>
<td>0.02</td>
<td>67.70</td>
<td>0.50</td>
<td></td>
</tr>
<tr>
<td>'DC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ 12 30 ]</td>
<td>0.30</td>
<td>'H5A1-45'</td>
<td>-45.00</td>
<td>45.00</td>
<td>'Analog 0-5V, 0-10V'</td>
<td>0.03</td>
<td>44.40</td>
<td>0.50</td>
<td></td>
</tr>
<tr>
<td>'DC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ 12 30 ]</td>
<td>0.30</td>
<td>'H5A1-60'</td>
<td>-60.00</td>
<td>60.00</td>
<td>'Analog 0-5V, 0-10V'</td>
<td>0.03</td>
<td>33.30</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>'DC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ 12 30 ]</td>
<td>0.30</td>
<td>'N2'</td>
<td>-10.00</td>
<td>10.00</td>
<td>'Analog mV'</td>
<td>0.002</td>
<td>12.00</td>
<td>0.50</td>
<td></td>
</tr>
<tr>
<td>'DC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ 3 6 ]</td>
<td>0.30</td>
<td>'N3'</td>
<td>-30.00</td>
<td>30.00</td>
<td>'Analog mV'</td>
<td>0.005</td>
<td>5.75</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>'DC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ 3 6 ]</td>
<td>0.30</td>
<td>'N4'</td>
<td>-70.00</td>
<td>70.00</td>
<td>'Analog mV'</td>
<td>0.010</td>
<td>3.65</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>'DC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ 8 30 ]</td>
<td>0.30</td>
<td>'NG2U'</td>
<td>-10.00</td>
<td>10.00</td>
<td>'Analog 0-5V'</td>
<td>0.001</td>
<td>200.00</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>'DC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ 8 30 ]</td>
<td>0.30</td>
<td>'NS3U'</td>
<td>-30.00</td>
<td>30.00</td>
<td>'Analog 0-5V'</td>
<td>0.003</td>
<td>67.00</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>'DC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ 8 30 ]</td>
<td>0.30</td>
<td>'NG4U'</td>
<td>-80.00</td>
<td>80.00</td>
<td>'Analog 0-5V'</td>
<td>0.010</td>
<td>25.00</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>'DC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ 8 30 ]</td>
<td>0.30</td>
<td>'LCP 196'</td>
<td>-90.00</td>
<td>90.00</td>
<td>'Analog 0-5V'</td>
<td>0.00017</td>
<td>0</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>'DC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ 12 18 ]</td>
<td>0.00</td>
<td>'LSRP-90'</td>
<td>-90.00</td>
<td>90.00</td>
<td>'Analog 0-5V'</td>
<td>0.0000573</td>
<td>0</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>'DC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ 12 18 ]</td>
<td>0.00</td>
<td>'L-210'</td>
<td>-80.00</td>
<td>80.00</td>
<td>'Analog mV'</td>
<td>0.0083</td>
<td>100.00</td>
<td>1.25</td>
<td></td>
</tr>
<tr>
<td>'AC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ 0 ]</td>
<td>0.00</td>
<td>'L-211U'</td>
<td>-60.00</td>
<td>60.00</td>
<td>'Analog mV'</td>
<td>0.0083</td>
<td>100.00</td>
<td>3.33</td>
<td></td>
</tr>
<tr>
<td>'AC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ 0 ]</td>
<td>0.00</td>
<td>'L-212T'</td>
<td>-45.00</td>
<td>45.00</td>
<td>'Analog mV'</td>
<td>0.0056</td>
<td>150.00</td>
<td>2.22</td>
<td></td>
</tr>
</tbody>
</table>

IncUnitLabelStr1 = {'N/A' 'Deg' 'Deg' 'N/A' 'Deg' 'mV/deg' 'V' 'N/A' 'V' 'sec' 'Hz'};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% INCLINOMETER Sensor Data Base Created Based on a Unit System
%% 2 - B Matrix INCLINOMETERSENSOR_DB(IS_Type, Property)
if strcmpi(UNITS,'SI') == 1
  % Units
  % Properties [ OPERATING TEMP STORAGE TEMP DIMENSIONS (mm) WEIGHT ]
  % INCLINOMETERSENSOR_DB2 = {
  -40.0 85.0 -45.0 90.0 66.0 66.0 28.0 227 ; 
  -40.0 85.0 -45.0 90.0 66.0 66.0 28.0 227 ; 
  -40.0 85.0 -45.0 90.0 66.0 66.0 28.0 227 ; 
  -40.0 85.0 -45.0 90.0 64.0 64.0 28.0 212.6 ;
end

if strcmp(UNITS,'British') == 1

% Properties

% Units
INCLINOMETERSSENSOR_DB2 = {
-40.0  85.0  -45.0  90.0  64.0  64.0  28.0  212.6 ;
-40.0  85.0  -45.0  90.0  64.0  64.0  28.0  212.6 ;
-40.0  85.0  -45.0  90.0  64.0  64.0  28.0  212.6 ;
-40.0  85.0  -45.0  90.0  64.0  64.0  28.0  212.6 ;
-40.0  85.0  -45.0  90.0  37.0  37.0  11.7  18.5 ;
-40.0  85.0  -45.0  90.0  37.0  37.0  11.7  18.5 ;
-40.0  85.0  -45.0  90.0  37.0  37.0  11.7  18.5 ;
-40.0  85.0  -45.0  90.0  37.0  37.0  11.7  18.5 ;
-40.0  85.0  -45.0  90.0  48.8  48.8  21.6  110 ;
-40.0  85.0  -45.0  90.0  48.8  48.8  21.6  110 ;
-40.0  110.0  -45.0  90.0  48.8  48.8  21.6  110 ;
-40.0  110.0  -45.0  90.0  48.8  48.8  21.6  110 ;
-40.0  110.0  -60.0  90.0  239.3  22.2  22.2  311.9 ;
-18.0  71.0  -60.0  90.0  40.64  36.32  36.32  113.4 ;
-54.0  124.0  -54.0  124.0  41.0  18.4  15.9  0 ;
-54.0  124.0  -54.0  124.0  41.0  18.4  15.9  0 ;
-54.0  124.0  -54.0  124.0  41.0  18.4  15.9  0 ;

IncUnitLabelStr2 = {'deg C' 'deg C' 'deg C' 'deg C' 'mm' 'mm' 'mm' 'gram'} ;
end

IncUnitLabelStr2 = {'deg F' 'deg F' 'deg F' 'deg F' 'in' 'in' 'in' 'oz'} ;
end

% Comments on Features / Use
INCLINOMETERSSENSOR_DB3[1.1] = ['Single Axis Angle Measurement' ;
'2 Levels of Temperature Compensation' ;
'Analog Voltage 0-5V Output' ;
'Serial Digital RS232 Output' ;
'Vibration and shock resistant' ;
'Environmentally sealed to IP66' ;
'Rugged die-cast Zinc housing' ;
INCLINOMETERSENSOR_DB3{2,1} = {'Single Axis Angle Measurement' ; '2 Levels of Temperature Compensation' ; 'Analog Voltage 0-5V Output' ; 'Serial Digital RS232 Output' ; 'Vibration and shock resistant' ; 'Environmentally sealed to IP66' ; 'Rugged die-cast Zinc housing' ; 'EMC protected' ; 'Mechanical Zero adjustment' ; 'EMC protected' ; 'Mechanical Zero adjustment' ;
'Single Axis Angle Measurement' ; '2 Levels of Temperature Compensation' ; 'Analog Voltage 0-5V Output' ; 'Serial Digital RS232 Output' ; 'Vibration and shock resistant' ; 'Environmentally sealed to IP66' ; 'Rugged die-cast Zinc housing' ; 'EMC protected' ; 'Mechanical Zero adjustment' ;
'Single Axis Angle Measurement' ; 'Temperature Compensation' ; '2 Analog Outputs: 0-5V, 0-10V' ; 'Digital Serial RS232 Output' ; 'Vibration and Shock Resistant' ; 'Environmentally sealed to IP66' ; 'Rugged Machined Aluminum Housing' ; 'EMC protected' ; 'Mechanical Zero adjustment' ;
'Single Axis Angle Measurement' ; 'Temperature Compensation' ; '2 Analog Outputs: 0-5V, 0-10V' ; 'Digital Serial RS232 Output' ; 'Vibration and Shock Resistant' ; 'Environmentally sealed to IP66' ; 'Rugged Machined Aluminum Housing' ; 'EMC protected' ; 'Mechanical Zero adjustment' ;
'Single Axis Angle Measurement' ; 'Temperature Compensation' ; '2 Analog Outputs: 0-5V, 0-10V' ; 'Digital Serial RS232 Output' ; 'Vibration and Shock Resistant' ; 'Environmentally sealed to IP66' ; 'Rugged Machined Aluminum Housing' ; 'EMC protected' ; 'Mechanical Zero adjustment' ;
'Single Axis Angle Measurement' ; 'Temperature Compensation' ; '2 Analog Outputs: 0-5V, 0-10V' ; 'Digital Serial RS232 Output' ; 'Vibration and Shock Resistant' ; 'Environmentally sealed to IP66' ; 'Rugged Machined Aluminum Housing' ;
'}
INCLINOMETERSENSOR_DB3{8,1} = { 'EMC protected', 'Mechanical Zero adjustment', 'Linear output characteristics', 'Minimal zero offset drift', 'Hysteresis free output signal', 'High measurement accuracy', 'Very low relative linearity errors', 'Integrated sensor electronics', 'Long-term stability', 'Low power consumption', 'Analog mV output signal', 'Hermetically sealed to IP65', 'Zero offset mechanically adjustable through 360 within mounting ring', 'No interference by ambient electromagnetic fields', 'Shockproof to 10,000g - no moving mechanical parts', 'Sensor electrically isolated from point of measurement using high quality PBT plastic housing', - no ground connections' },

INCLINOMETERSENSOR_DB3{9,1} = { 'Linear output characteristics', 'Minimal zero offset drift', 'Hysteresis free output signal', 'High measurement accuracy', 'Very low relative linearity errors', 'Integrated sensor electronics', 'Long-term stability', 'Low power consumption', 'Analog mV output signal', 'Hermetically sealed to IP65', 'Zero offset mechanically adjustable through 360 within mounting ring', 'No interference by ambient electromagnetic fields', 'Shockproof to 10,000g - no moving mechanical parts', 'Sensor electrically isolated from point of measurement using high quality PBT plastic housing', - no ground connections' },

INCLINOMETERSENSOR_DB3{10,1} = { 'Linear output characteristics', 'Minimal zero offset drift', 'Hysteresis free output signal', 'High measurement accuracy', 'Very low relative linearity errors', 'Integrated sensor electronics', 'Long-term stability', 'Low power consumption', 'Analog mV output signal', 'Hermetically sealed to IP65', 'Zero offset mechanically adjustable through 360 within mounting ring', 'No interference by ambient electromagnetic fields', 'Shockproof to 10,000g - no moving mechanical parts', 'Sensor electrically isolated from point of measurement using high quality PBT plastic housing', - no ground connections' },

INCLINOMETERSENSOR_DB3{11,1} = { 'Liquid capacitive based inclinometers', 'Analog DC Output', 'Single Supply Input', 'Low Power Consumption', 'IP65 Packaging', 'Shock Resistant', 'Hysteresis Free' };
INCLINOMETERSENSOR_DB3\{12,1\} = \{ 'Liquid capacitive based inclinometers' ; 'Analog DC Output' ; 'Single Supply Input' ; 'Low Power Consumption' ; 'IP65 Packaging' ; 'Shock Resistant' ; 'Hysteresis Free' \};

INCLINOMETERSENSOR_DB3\{13,1\} = \{ 'Liquid capacitive based inclinometers' ; 'Analog DC Output' ; 'Single Supply Input' ; 'Low Power Consumption' ; 'IP65 Packaging' ; 'Shock Resistant' ; 'Hysteresis Free' \};

INCLINOMETERSENSOR_DB3\{14,1\} = \{ 'Biaxial Inclinometer - two-axis tilt sensor' ; 'Excellent turn-on-repeatability' ; 'Very low hysteresis' \};

INCLINOMETERSENSOR_DB3\{15,1\} = \{ 'Compact, cylindrical shape' ; 'Stacking Feature allows for several unit use for multi-axis measurement' ; 'Precise readings with high outputs at lower range frequencies' ; 'Applications - Heavy Construction, Grading, Ship & Barge Leveling, Deviation Surveys, Continuous Casting, Weapons Platform Leveling' ; 'Seals meet MIL-STD 202, Method 112 Requirements' \};

INCLINOMETERSENSOR_DB3\{16,1\} = \{ 'Two Single Axis Sensors Mounted on Single Unit' ; 'Wide angular range, high accuracy, dynamic output attenuation' ; 'L Series incorporates a specially designed dampering orifice' ; 'Includes hermetic sealing, compact size, and are available in a variety of housing configurations' ; 'Applications - Industrial, Aerospace, Military, Photonics, Geotechnical, Oceanographic, Construction.' ; 'Flight Rated Production Company' \};

INCLINOMETERSENSOR_DB3\{17,1\} = \{ 'Two Single Axis Sensors Mounted on Single Unit' ; 'Wide angular range, high accuracy, dynamic output attenuation' ; 'L Series incorporates a specially designed dampering orifice' ; 'Flight Proven Product - Cassini-Huygens Mission Huygens Probe' ; 'Includes hermetic sealing, compact size, and are available in a variety of housing configurations' ; 'Applications - Industrial, Aerospace, Military, Photonics, Geotechnical, Oceanographic, Construction.' ; 'Flight Rated Production Company' \};

INCLINOMETERSENSOR_DB3\{18,1\} = \{ 'Two Single Axis Sensors Mounted on Single Unit' ; 'Wide angular range, high accuracy, dynamic output attenuation' ; 'L Series incorporates a specially designed dampering orifice' ; 'Includes hermetic sealing, compact size, and are available in a variety of housing configurations' ; 'Applications - Industrial, Aerospace, Military, Photonics, Geotechnical, Oceanographic, Construction.' ; 'Flight Rated Production Company' \};

\% Combine Data Arrays into a Single Data Matrix
INCLINOMETERSENSOR_DB = cat(2,INCLINOMETERSENSOR_DB1,INCLINOMETERSENSOR_DB2,INCLINOMETERSENSOR_DB3);
IncUnitLabelStr = cat(2,IncUnitLabelStr1,IncUnitLabelStr2);
% Select Inclination Sensor by Input Operation Range Value

\[
c = 1;
\]

\[
\text{for } i = 1:\text{size(INCLINOMETERSENSOR_DB,1)}
\]
\[
\text{if } (\text{INCLINOMETERSENSOR_DB}{i,3} < \text{INC_ANGLE}) \&\& (\text{INCLINOMETERSENSOR_DB}{i,3} >= \text{TOL_LOWER})
\]
\[
\text{for } s = 1:\text{size(INCLINOMETERSENSOR_DB,2)}
\]
\[
\text{INCL_MATRIX}{c,s} = \text{INCLINOMETERSENSOR_DB}{i,s};
\]
\[
c = c + 1;
\]
\[
\text{end}
\]
\[
\text{end}
\]

% Continue Selection Process From Viable Options - If Multiple Sensor Options Exist

\[
\text{if size(INCL_MATRIX,1) > 1}
\]

% Select Via Temperature Operational Range

\[
c = 1;
\]

% Build Up Database of Possible Inclinometer Sensors Based on Temperature Range

\[
\text{for } t = 1:\text{size(INCL_MATRIX,1)}
\]
\[
\text{if } (\text{PLTEMP} >= \text{INCL_MATRIX}{t,12}) \&\& (\text{PLTEMP} <= \text{INCL_MATRIX}{t,13})
\]
\[
\text{for } a = 1:\text{size(INCL_MATRIX,2)}
\]
\[
\text{INCL_MATRIX2}{c,a} = \text{INCL_MATRIX}{t,a};
\]
\[
c = c + 1;
\]
\[
\text{end}
\]
\[
\text{end}
\]

% Copies INCL_MATRIX to INCL_MATRIX2 if all Sensors Eliminated based on Temperature Range

\[
\text{if exist('INCL_MATRIX2') == 0}
\]
\[
\text{INCL_MATRIX2 = INCL_MATRIX};
\]
\[
\text{end}
\]

% If size(INCL_MATRIX2,1) > 1

% Down Select by Max Temperature Range

\[
\text{for } d = 1:\text{size(INCL MATRIX2,1)}
\]
\[
\text{TEMP_INCL_DELTA}{d} = \text{INCL MATRIX2}{d,13} - \text{INCL MATRIX2}{d,12};
\]
\[
\text{end}
\]

% Find the Maximum Inclinometer Temperature Range Sensor

\[
\text{MAX_TEMP_INCL_DELTA} = \max(\text{TEMP_INCL_DELTA});
\]

% For Multiple Sensors with the Same Range Reduce Sensor Matrix to Those
% with the Greatest Range

\[
k = 1;
\]

\[
\text{for } c = 1:\text{length(TEMP_INCL_DELTA)}
\]
\[
\text{if } \text{TEMP_INCL_DELTA}{c} == \text{MAX_TEMP_INCL_DELTA}
\]
\[
\text{for } j = 1:\text{size(INCL MATRIX2,2)}
\]
\[
\text{INCL_MATRIX3}{k,j} = \text{INCL MATRIX2}{c,j};
\]
\[
k = k + 1;
\]
\[
\text{end}
\]
\[
\text{end}
\]
\[
\text{end}
\]
% Copies INCL_MATRIX2 to INCL_MATRIX3 if all Sensors Eliminated based on Temperature Range
if exist('INCL_MATRIX3') == 0
    INCL_MATRIX3 = INCL_MATRIX2;
end

if size(INCL_MATRIX3,1) >1
    % Create Mass / Volume Scaling Factor Optimization
    for mv = 1:size(INCL_MATRIX3,1)
        INC_SCALE_FAC(mv) = INCL_MATRIX3{mv,10} * INCL_MATRIX3{mv,11} * INCL_MATRIX3{mv,12} * INCL_MATRIX3{mv,13};
        if (INCL_MATRIX3{mv,10} == 0) || (INCL_MATRIX3{mv,11} == 0) || (INCL_MATRIX3{mv,12} == 0) || (INCL_MATRIX3{mv,13} == 0)
            INC_SCALE_FAC(mv) = 9.99E99;
        end
    end

    [ INC_MIN_SCL, INDEX ] = min(INC_SCALE_FAC);

    % Build Sensor Matrix Based on Scale Factors
    k = 1;
    for c = 1:length(INC_SCALE_FAC)
        if INC_SCALE_FAC(c) == INC_MIN_SCL
            for j = 1:size(INCL_MATRIX3,2)
                INCL_MATRIX4{k,j} = INCL_MATRIX3{c,j};
            end
            k = k + 1;
        end
    end

end

% Copies INCL_MATRIX3 to INCL_MATRIX4 if all Sensors Eliminated based on Mass Volume Scale Factor
if exist('INCL_MATRIX4') == 0
    INCL_MATRIX4 = INCL_MATRIX3;
end

if size(INCL_MATRIX4,1) >1
    % Search for Flight Rated Components
    k = 1;
    for c = 1:size(INCL_MATRIX4,1)
        for pos = 1:length(INCL_MATRIX4{c,20})
            if findstr(INCL_MATRIX4{c,20}{pos},'Flight Proven Product') == 1
                for loc = 1:size(INCL_MATRIX4,2)
                    INCL_MATRIX5{k,loc} = INCL_MATRIX4{c,loc};
                end
                k = k + 1;
            end
        end
    end
end
end
% Copies INCL_MATRIX4 to INCL_MATRIX5 if all Sensors Eliminated based on Temperature Range
if exist('INCL_MATRIX5') == 0
  INCL_MATRIX5 = INCL_MATRIX4;
end

% Save Density Sensor Properties to Data File
save('INCLINOMETER_SENSOR_Data.mat');
save('INCLINOMETER SENSOR Final.mat','INCL MATRIX5','IncUnitLabelStr');

% PRINT SUMMARY Output Section for 'Print'
if strcmpi(Print,'Y') == 1
  fprintf('

*** INCLINOMETER SENSOR SUMMARY RESULTS ***

** Sensor Properties **

Inclinometer Sensor Type: %s
Sensing Inclination Range:
  Low: %10.3f %s
  High: %10.3f %s
Output Data Format: %s
Sensor Resolution: %10.7f %s
Sensor Sensitivity: %10.3f %s
Sensor Non-Linearity: %10.4f %s
Response Time: %10.2f %s
Sensor Bandwidth: %s

% Environmental Properties
Environmental Characteristics:
Operating Temperature Range - Low: %9.3f %s
  High: %9.3f %s
Storage Temperature Range - Low: %9.3f %s
  High: %9.3f %s
Power Requirements:
Input Voltage Type: %s
Input Voltage: %10.7f %s

for i = 1:length(INCL_MATRIX5{1,9})
  fprintf('%d', INCL_MATRIX5{l,9}(i) );
end
fprintf(' %s', INCL_MATRIX5{1,8}, IncUnitLabelStr{8})

% PRINT SUMMARY Output Section for 'Print'
if strcmpi(Print,'Y') == 1
  fprintf('

*** INPUTS ***
Inclination Angle Range: %s
Unit System: %s

');
end
fprintf('

Physical Properties');
fprintf('
  Dimensions - Length: %10.3f %s', INCL_MATRIX5{1,16}, IncUnitLabelStr{16});
fprintf('
            Width: %10.3f %s', INCL_MATRIX5{1,17}, IncUnitLabelStr{17});
fprintf('
            Height: %10.3f %s', INCL_MATRIX5{1,18}, IncUnitLabelStr{18});
fprintf('
  Sensor Mass: %10.3f %s', INCL_MATRIX5{1,19}, IncUnitLabelStr{19});

fprintf('

Sensor Comments: ');
for r = 1:length(INCL_MATRIX5{1,20})
  fprintf('
            %s', INCL_MATRIX5{l,20}{r});
end

***************

Fout = fopen('INCLINOMETER_SENSOR_Summary.txt','w+');

fprintf(Fout, '

*** INCLINOMETER SENSOR SUMMARY RESULTS ***');
fprintf(Fout, '

**» INPUTS ***');
fprintf(Fout, ' 
 Inclination Angle Range: %s', INCR{l});
fprintf(Fout, ' 
 Unit System: %s ' , UNITS);

fprintf(Fout, '

******* SUMMARY RESULTS **********
')

% Inclinometer Sensor Properties
fprintf(Fout, '

% Inclometer Sensor Type: %s', INCL_MATRIX5{1,1});
fprintf(Fout, ' 
 Sensing Inclination Range - Low: %10.3f %s', INCL_MATRIX5{1,2}, IncUnitLabelStr{2});
fprintf(Fout, ' 
                          High: %10.3f %s', INCL_MATRIX5{1,3}, IncUnitLabelStr{3});
fprintf(Fout, ' 
 Output Data Format: %10.3f %s', INCL_MATRIX5{1,4}, IncUnitLabelStr{4});
fprintf(Fout, ' 
 Sensor Resolution: %10.7f %s', INCL_MATRIX5{1,5}, IncUnitLabelStr{5});
fprintf(Fout, ' 
 Sensor Sensitivity: %10.3f %s', INCL_MATRIX5{1,6}, IncUnitLabelStr{6});
fprintf(Fout, ' 
 Sensor Non-Linearity: %10.4f %s', INCL_MATRIX5{1,7}, IncUnitLabelStr{7});
fprintf(Fout, ' 
 Response Time: %10.3f %s', INCL_MATRIX5{1,10}, IncUnitLabelStr{10});
fprintf(Fout, ' 
 Sensor Bandwidth: %10.2f %s', INCL_MATRIX5{1,11}, IncUnitLabelStr{11});

% Environmental Properties
fprintf(Fout, '

% Operating Temperature Range - Low: %9.3f %s', INCL_MATRIX5{1,12}, IncUnitLabelStr{12});
fprintf(Fout, ' 
                          High: %9.3f %s', INCL_MATRIX5{1,13}, IncUnitLabelStr{13});
fprintf(Fout, ' 
 Storage Temperature Range - Low: %9.3f %s', INCL_MATRIX5{1,14}, IncUnitLabelStr{14});
fprintf(Fout, ' 
                          High: %9.3f %s', INCL_MATRIX5{1,15}, IncUnitLabelStr{15});

% Power Requirements
fprintf(Fout, '

% Input Voltage Type: %s %s', INCL_MATRIX5{1,8}, IncUnitLabelStr{8});
for i = 1:length(INCL_MATRIX5{1,9})
  fprintf(Fout, ' 
            %d', INCL_MATRIX5{1,9}{i});
end
fprintf(Fout, ' 
 Input Voltage: %s', INCL_MATRIX5{1,9}{i});

fprintf(Fout, 'n

 Physical Properties');
fprintf(Fout, 'n Dimensions - Length: %10.3f %s', INCL_MATRIX5(1,16), IncUnitLabelStr(16));
fprintf(Fout, 'n Width: %10.3f %s', INCL_MATRIX5(1,17), IncUnitLabelStr(17));
fprintf(Fout, 'n Height: %10.3f %s', INCL_MATRIX5(1,18), IncUnitLabelStr(18));
fprintf(Fout, 'n Sensor Mass: %10.3f %s', INCL_MATRIX5(1,19), IncUnitLabelStr(19));

fprintf(Fout, 'n

 Sensor Comments:');
for r = 1:length(INCL_MATRIX5(1,20))
    fprintf(Fout, '
 %s', INCL_MATRIX5(1,20){r} );
end
fclose(Fout);

D13. Nephelometer Sensors
% function NEPHELOMETER_SENSORS(UNITS,Print)
% Nephelometer Sensor Database
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
% This file contains Nephelometer Sensor values for the ISSPO program.
% Data here is loaded into the main program to determine the proper sensor
% to use for Missions based on the sensor requirements.
% Values are assigned to a variable name and saved to a MatLAB '.mat' file
% and loaded when calling the ISSPO program.
% Some of the values are based on the Unit system chosen and are loaded
% in separate sections.
% References
% http://www.tsi.com/Product.aspx?Pid=64
% http://www.optecinc.com/cptec_041.htm

function NEPHELOMETER_SENSORS(UNITS,Print)
% Load Constants values into Local Program
load('Constants_DB.mat');
load('Planet_Data.mat');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%%%%%
%% 2 - D Matrix NEPHELOMETER_DB(N_Type, Property)

if strcmpi(UNITS,'SI') == 1
% Properties % Units
% Wavelength1 Wavelength2 Wavelength3 Bandwidth Weight Power Voltage Width Height
% nm nm nm kg W V mm mm
NEXPHLOMETER_DB = {'TSI-3563' 450 550 700 40 18 175 24 300 1100
  4096 2.000E-02 7 170
  271 600 0 5 175
  271 810 0 5 175};
% Properties % Units
% Response Time Rec Flow Rate Operating Temperature Ambien % RH
% sec min L/min Low deg C High deg C Low % High %
NEXPHLOMETER_DB2 = [ 10 20 200 10 40 0 95 ];
  4 2 2 -20 45 0 0 ];
  4 7 9 -10 45 0 100 ];;
NephUnitLabelStrl = ['sensor' 'nm' 'nm' 'nm' 'kg' 'W' 'V' 'mm' 'mm' 'sec' 'per m @ 60 sec' 'deg' 'deg']
NephUnitLabelStr2 = ['sec' 'L/min' 'L/min' 'deg C' 'deg C' ' RH' ' RH' ];

% Required Power Supply Specifications
% Properties % Units
% Weight Power Type Voltage Width Height Length
% kg W N/A V mm mm mm
NephPS_DB = { 18 175 'DC' 24 305 178 102 ];
  0 62.5 'DC' 13.8 203 152 51 ];
  0 62.5 'DC' 13.8 203 152 51 ];;
NephFSLabelStrl = ['kg' 'W' 'N/A' 'V' 'mm' 'mm' 'mm'];

end

if strcmpi(UNITS,'BRITISH') == 1
% Properties % Units
% Height Length Avg Time Drift Angle Integration
% inch inch inch inch inch inch lb BTU's/hr V inch
NEXPHLOMETER_DB = {'TSI-3563' 1.7716E-05 9.84 43.3 4096 6.096E-08 2.1653E-05 2.7559E-05 1.5748E-06 39.68 597.656 24 11.82
  1.2598E-05 1.8492 2.37755 14 4.5175 ];
  0 5 175 ];
  16.5 8.2 600 0 5 175 ];
  16.5 8.2 810 0 5 175 ];
% Properties % Units
% Response Time Rec Flow Rate Operating Temperature Ambien % RH
% sec min L/min Low deg F High deg F Low % High %
NEXPHLOMETER_DB2 = [ 10 5.28344 52.8344 50 104 0 95 ];
  4 0.528344 52.8344 -4 113 0 0 ];
  4 1.8492 2.37755 14 113 0 100 ];;

end
NephUnitLabelStr1 = {'N/A' 'inch' 'inch' 'inch' 'inch' 'lb' 'BTU/hr' 'V' 'inch' 'inch' 'inch' 'sec' 'per ft @ 60 sec' 'deg' 'deg'};
NephUnitLabelStr2 = {'sec' 'L/min' 'L/min' 'deg F' 'deg F' '% RH' '% RH'};

% Required Power Supply Specifications

% Properties
% Weight Power Type Voltage Width Height Length
NephPS_DB = {
    [11 597.656 'DC' 24 12 7 4 ;
    0 213.5 'DC' 13.8 8 6 2 ;
    0 213.5 'DC' 13.8 8 6 2 ];

NephPSLabelStr = {'lb' 'BTU/hr' 'N/A' 'V' 'in' 'in' 'in'};

end

NEPHELOMETER_DB = cat(2,NEPHELOMETER_DB,NEPHELOMETER_DB2);
NephUnitLabelStr = cat(2,NephUnitLabelStr1,NephUnitLabelStr2);

%comments on sensors
NEPHELOMETER_DB{1,23} = 'High sensitivity';
NEPHELOMETER_DB{2,23} = 'Unit may be outdated, floppy drive data recovery';
NEPHELOMETER_DB{3,23} = 'Unit may be outdated, floppy drive data recovery';

% Find the Max Operational Range
for n = 1:size(NEPHELOMETER_DB,1)
    N_DELTA(n) = NEPHELOMETER_DB{n,4} - NEPHELOMETER_DB{n,2};
end

% Find the Maximum Nephelometer Operating Range of the Sensor
MAX_N_DELTA = max(N_DELTA);

% For Multiple Sensors with the Same Range Reduce Sensor Matrix to Those
% with the Greatest Range
k = 1;
for c = 1:length(N_DELTA)
    if N_DELTA(c) == MAX_N_DELTA
        for j = 1:size(NEPHELOMETER_DB,2)
            N_MATRIX{k,j} = NEPHELOMETER_DB{c,j};
        end
        k = k + 1;
    end
end
if exist('N_MATRIX') == 0
    N_MATRIX = NEPHELOMETER_DB;
end

% Select Sensor Based on Integration Range if More than One Sensor Exists In Database
if size(N_MATRIX,1) > 1
    % Find the Max Operational Range
    for n = 1:size(N_MATRIX,1)
        N_DELTA_RNG(n) = N_MATRIX{n,15} - N_MATRIX{n,14};
    end
...
% Find the Maximum Nephelometer Operating Range of the Sensor
MAX_N_DELTA_RNG = max(N_DELTA_RNG);

% For Multiple Sensors with the Same Range Reduce Sensor Matrix to Those
% with the Greatest Range
k = 1;
for c = 1:length(N_DELTA_RNG)
    if N_DELTA_RNG(c) == MAX_N_DELTA_RNG
        for j = 1:size(NEPHELOMETER_DB,2)
            N_MATRIX2{k,j} = N_MATRIX{c,j};
        end
        k = k + 1;
    end
end

if exist('N_MATRIX2') == 0
    N_MATRIX2 = N_MATRIX;
end

% Select Sensor Based on Minimum Volume if More than One Sensor Exists In Database
if size(N_MATRIX,1) > 1
    % Create Volume Scaling Factor Optimization
    for v = 1:size(N_MATRIX2,1)
        N_SCALE_FAC(v) = N_MATRIX2{v,9} * N_MATRIX2{v,10} * N_MATRIX2{v,11} * N_MATRIX2{v,6};
        if (N_MATRIX2{v,9} == 0) || (N_MATRIX2{v,10} == 0) || (N_MATRIX2{v,11} == 0) || (N_MATRIX2{v,6} == 0)
            N_SCALE_FAC(v) = 9.99999;
        end
    end
    [ N_MIN_SCL, INDEX ] = min(N_SCALE_FAC);
    % Build Sensor Matrix Based on Scale Factors
    k = 1;
    for c = 1:length(N_SCALE_FAC)
        if N_SCALE_FAC(c) == N_MIN_SCL
            for j = 1:size(N_MATRIX2,2)
                N_MATRIX3{k,j} = N_MATRIX2{c,j};
            end
            k = k + 1;
        end
    end
end

% Copies N_MATRIX2 to N_MATRIX3 if all Sensors Eliminated
if exist('N_MATRIX3') == 0
    N_MATRIX3 = N_MATRIX2;
end

% Determine the Corresponding Power Supply Array
for n = 1:size(NEPHELOMETER_DB,1)
    if strcmpi(NEPHELOMETER_DB{n,1}, N_MATRIX3{1,1}) == 1
        break;
    end
end

% Save Nephelometer Sensor Properties to Data File
save('NEPHEL0METER_SENSOR_Data.mat');
save('NEPHEL0METER_SENSOR_Final.mat','N_MATRIX3','NephUnitLabelStr');

% *** PRINT SUMMARY Output Section for 'Print' = 'Y'
if strcmpi(Print,'Y') == 1
    fprintf(' *** NEPHEL0METER SENSOR SUMMARY RESULTS ***

    *** INPUTS ***
    Unit System: %s', UNITS);
    fprintf('
    Power Source: %12.3f %s', N_MATRIX3{1,7}, NephUnitLabelStr{7});
    fprintf('
    Input Voltage: %12.3f %s', N_MATRIX3{1,8}, NephUnitLabelStr{8});
    fprintf('
    Power Requirements');
    fprintf('
    Sensor Mass: %12.2f %s', N_MATRIX3{1,6}, NephUnitLabelStr{6});
    fprintf('
    Sensor Dimensions - Length: %12.2f %s', N_MATRIX3{1,11}, NephUnitLabelStr{11});
    fprintf('
    Width: %12.2f %s', N_MATRIX3{1,9}, NephUnitLabelStr{9});
    fprintf('
    Height: %12.2f %s', N_MATRIX3{1,10}, NephUnitLabelStr{10});

% Environmental Properties
fprintf('

 Environmental Properties');
 fprintf('
 Temperature Range - Low: %12.2f %s', N MATRIX3{1,19}, NephUnitLabelStr{19} );
 fprintf('
 High: %12.2f %s', N MATRIX3{1,20}, NephUnitLabelStr{20} );
 fprintf('
 Relative Humidity - Min: %12.2f %s', N MATRIX3{1,21}, NephUnitLabelStr{21} );
 fprintf('
 Max: %12.2f %s', N MATRIX3{1,22}, NephUnitLabelStr{22} );

fprintf('

 Sensor Comments:');
 fprintf(Fout, '

 % Power Supply Properties
 fprintf(Fout, '

 % Power Generated: %12.3f %s', N_matrix3{1,7}, NephUnitLabelStr{7} );

end

************************************************************************

% Writes a text file summary of the Planetary Properties
 Fout = fopen('NEPHELOMETER_SENSOR_Summary.txt','w+');
 fprintf(Fout, '

 *** NEPHELOMETER SENSOR SUMMARY RESULTS ***
 fprintf(Fout, '

 *** INPUTS ***
 fprintf(Fout, '

 Unit System: %s', UNITS);

fprintf(Fout, '

*** NEPHELOMETER SENSOR PROPERTIES ***
 fprintf(Fout, '

 Nephelometer Sensor Type: %s', N MATRIX3{1,1});
 fprintf(Fout, '

 Sensing Wavelengths - Min: %12.4f %s', N MATRIX3{1,2}, NephUnitLabelStr{2} );
 fprintf(Fout, '

 Middle: %12.4f %s', N MATRIX3{1,3}, NephUnitLabelStr{3} );
 fprintf(Fout, '

 Max: %12.4f %s', N MATRIX3{1,4}, NephUnitLabelStr{4} );
 fprintf(Fout, '

 Sensing Bandwidth: %12.4f %s', N MATRIX3{1,5}, NephUnitLabelStr{5} );
 fprintf(Fout, '

 Sampling Time: %12.2f %s', N MATRIX3{1,12}, NephUnitLabelStr{12} );
 fprintf(Fout, '

 Sensing Drift Rate: %12.3E %s', N MATRIX3{1,13}, NephUnitLabelStr{13} );
 fprintf(Fout, '

 Integration Angles - Min: %12.2f %s', N MATRIX3{1,14}, NephUnitLabelStr{14} );
 fprintf(Fout, '

 Max: %12.2f %s', N MATRIX3{1,15}, NephUnitLabelStr{15} );
 fprintf(Fout, '

 Response Time: %12.2f %s', N MATRIX3{1,16}, NephUnitLabelStr{16} );
 fprintf(Fout, '

 Sampling Flow Rate - Min: %12.2f %s', N MATRIX3{1,17}, NephUnitLabelStr{17} );
 fprintf(Fout, '

 Max: %12.2f %s', N MATRIX3{1,18}, NephUnitLabelStr{18} );
D14. Pressure Sensors

% function PRESSURE_SENSORS(PLPRESS,UNITS,Print)
% Pressure Sensor Database
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
% This file contains Pressure Sensor values for the ISSPO program.
% Data here is loaded into the main program to determine the proper sensor
% to use for Missions based on the planetary Atmosphere composition.
% Values are assigned to a variable name and saved to a MatLAB .mat file
% and loaded when calling the ISSPO program.
Some of the values are based on the Unit system chosen and are loaded in separate sections.

References

http://www.ascovalve.com/Applications/Products/SensorsPressureSensorsData.aspx
http://sensing.honeywell.com/index.cfm?ci_id=140264&defId=121030
http://www.gesensing.com/products/pdcr3500.htm?bc=bc_drucks
http://www.climatronics.com/Products/Sensors/atmospheric_pressure.php
TO BE ADDED
http://www.taberindustries.com/transducer/Aerospace.asp

TO BE ADDED

function PRESSURE_SENSORS(PLPRESS, UNITS, Print)

% Load Constants values into Local Program
load('Constants_DB.mat');
load('Planet_Data.mat');

% Assign Planetary Surface Properties to Local Variable
PLTEMP = Planet_Properties{7};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Pressure Sensor Data Base Created Based on un Unit System
%% 2 - D Matrix PRESSSENSOR_DB (P_Type, Property)
if strcmpi(UNITS,'SI')

% PROOF BURST: SENSOR SHOCK IMPULSE
% ACCURACY STABILITY \| OVER PRESSURE BURST PRESSURE LIFE CYCLE PRESSURE RANGE
% UNITS: TYPE g's sec % % % % % % % % % % % % % % %
% UNITS: Rated # Low kPa High

kPa PRESSSENSOR_DB1 = [ 'Ser 48-0025' 0.5 0.25 2.0 5.0 100E+06 0.0 172.369
344.738 861.845
'Ser 48-0050' 0.5 0.25 2.0 5.0 100E+06 0.0 344.738
689.476 1723.690
'Ser 48-0100' 0.5 0.25 2.0 5.0 100E+06 0.0 689.476
1378.950 3447.380
'Ser 43-0200' 0.5 0.25 2.0 5.0 100E+06 0.0 1378.950
2757.900 6894.760
'Ser 48-1000' 0.5 0.25 2.0 5.0 100E+06 0.0 6894.760
13789.500 34473.850
'19(C,U)-003P' 2.0 0.60 3.0 5.0 1E+06 0.0 20.684
62.053 103.421
'19(C,U)-005P' 2.0 0.60 3.0 5.0 1E+06 0.0 34.474
103.421 172.369
'19(C,U)-010P' 2.0 0.60 3.0 5.0 1E+06 0.0 68.948
206.843 344.738
];
| 85.47 | 19.05 | 19.05 | -40.0 | 125.00 | -40.0 | 125.00 | 5.0 | 2.0 | 0 | ‘Stainless Steel’ | ‘Stainless Steel’ |
| 85.47 | 19.05 | 19.05 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | % 7 |
| 85.47 | 19.05 | 19.05 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | % 8 |
| 85.47 | 19.05 | 19.05 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | % 9 |
| 85.47 | 19.05 | 19.05 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | % 10 |
| 85.47 | 19.05 | 19.05 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | % 11 |
| 85.47 | 19.05 | 19.05 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | % 12 |
| 85.47 | 19.05 | 19.05 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | % 13 |
| 85.47 | 19.05 | 19.05 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | % 14 |
| 85.47 | 19.05 | 19.05 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | % 15 |
| 31.00 | 30.00 | 30.00 | 79.38 | 0 | 0 | 0 | 0 | 0 | 0 | % 16 |
| 31.00 | 30.00 | 30.00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | % 17 |
| 81.03 | 19.05 | 19.05 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | % 18 |
| 81.03 | 19.05 | 19.05 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | % 19 |
| 81.03 | 19.05 | 19.05 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | % 20 |
| 81.03 | 19.05 | 19.05 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | % 21 |
| 81.03 | 19.05 | 19.05 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | % 22 |
| 92.71 | 25.40 | 25.40 | -53.9 | 121.11 | -53.9 | 125.00 | 5.0 | 0.0 | 0 | ‘Stainless Steel’ | ‘Stainless Steel’ |
| 102.87 | 25.40 | 25.40 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | % 23 |
| 88.90 | 31.75 | 31.75 | -54.0 | 121.00 | -54.0 | 121.00 | 36.0 | 0.0 | 0 | ‘Stainless Steel’ | ‘Stainless Steel’ |

```
<table>
<thead>
<tr>
<th>Units</th>
<th>TYPE</th>
<th>Low #1 yr</th>
<th>@ Mx Rated Low</th>
<th>@ Mx Rated High</th>
<th>Rated #</th>
<th>Low psig</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>50.0</td>
<td>'Ser 48-0025'</td>
<td>0.5</td>
<td>0.25</td>
<td>2.0</td>
<td>5.0</td>
<td>100E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>100.0</td>
<td>'Ser 48-0050'</td>
<td>0.5</td>
<td>0.25</td>
<td>2.0</td>
<td>5.0</td>
<td>100E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>200.0</td>
<td>'Ser 48-0100'</td>
<td>0.5</td>
<td>0.25</td>
<td>2.0</td>
<td>5.0</td>
<td>100E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>400.0</td>
<td>'Ser 48-0200'</td>
<td>0.5</td>
<td>0.25</td>
<td>2.0</td>
<td>5.0</td>
<td>100E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>2000.0</td>
<td>'Ser 48-1000'</td>
<td>0.5</td>
<td>0.25</td>
<td>2.0</td>
<td>5.0</td>
<td>100E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>9.0</td>
<td>'19(C,U)-003P'</td>
<td>2.0</td>
<td>0.60</td>
<td>3.0</td>
<td>5.0</td>
<td>1E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>15.0</td>
<td>'19(C,U)-005P'</td>
<td>2.0</td>
<td>0.60</td>
<td>3.0</td>
<td>5.0</td>
<td>1E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>30.0</td>
<td>'19(C,U)-010P'</td>
<td>2.0</td>
<td>0.60</td>
<td>3.0</td>
<td>5.0</td>
<td>1E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>45.0</td>
<td>'19(C,U)-015P'</td>
<td>2.0</td>
<td>0.60</td>
<td>3.0</td>
<td>5.0</td>
<td>1E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>90.0</td>
<td>'19(C,U)-030P'</td>
<td>2.0</td>
<td>0.60</td>
<td>3.0</td>
<td>5.0</td>
<td>1E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>150.0</td>
<td>'19(C,U)-050P'</td>
<td>2.0</td>
<td>0.60</td>
<td>3.0</td>
<td>5.0</td>
<td>1E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>300.0</td>
<td>'19(C,U)-100P'</td>
<td>2.0</td>
<td>0.60</td>
<td>3.0</td>
<td>5.0</td>
<td>1E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>600.0</td>
<td>'19(C,U)-200P'</td>
<td>2.0</td>
<td>0.60</td>
<td>3.0</td>
<td>5.0</td>
<td>1E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>900.0</td>
<td>'19(C,U)-300P'</td>
<td>2.0</td>
<td>0.60</td>
<td>3.0</td>
<td>5.0</td>
<td>1E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>1200.0</td>
<td>'19(C,U)-500P'</td>
<td>2.0</td>
<td>0.60</td>
<td>2.4</td>
<td>4.8</td>
<td>1E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>'OMRON EB'</td>
<td>3.0</td>
<td>0.0</td>
<td>25.14</td>
<td></td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>'OMRON EB'</td>
<td>3.0</td>
<td>0.0</td>
<td>9.93</td>
<td></td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>1500.0</td>
<td>'13(C,U)-050P'</td>
<td>2.0</td>
<td>0.60</td>
<td>3.0</td>
<td>5.0</td>
<td>1E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>3000.0</td>
<td>'13(C,U)-100P'</td>
<td>2.0</td>
<td>0.60</td>
<td>3.0</td>
<td>5.0</td>
<td>1E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>6000.0</td>
<td>'13(C,U)-200P'</td>
<td>2.0</td>
<td>0.60</td>
<td>3.0</td>
<td>5.0</td>
<td>1E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>9000.0</td>
<td>'13(C,U)-300P'</td>
<td>2.0</td>
<td>0.60</td>
<td>3.0</td>
<td>5.0</td>
<td>1E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>10000.0</td>
<td>'13(C,U)-500P'</td>
<td>2.0</td>
<td>0.60</td>
<td>2.0</td>
<td>4.0</td>
<td>1E+06</td>
<td>0.0</td>
</tr>
<tr>
<td>10000.0</td>
<td>'PDAC 3500'</td>
<td>0.1</td>
<td>0.30</td>
<td>2.0</td>
<td>6.0</td>
<td>0</td>
<td>5.0</td>
</tr>
<tr>
<td>2000.0</td>
<td>'PDAC 3500'</td>
<td>0.1</td>
<td>0.30</td>
<td>2.0</td>
<td>6.0</td>
<td>0</td>
<td>5.0</td>
</tr>
<tr>
<td>10000.0</td>
<td>'240VSAT'</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>3.0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>10000.0</td>
<td>'240VSAT'</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>3.0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>% Properties</td>
<td>OPERATING TEMP</td>
<td>STORAGE TEMP</td>
<td>VOLTAGE</td>
<td>CURRENT</td>
<td>BANDWIDTH</td>
<td>MATERIAL</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
<td>--------------</td>
<td>---------</td>
<td>---------</td>
<td>------------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>LENGTH</td>
<td>TEMPERATURE SENSOR PROPERTIES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRESSSENSOR_DB2 = {</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.912 0.875 0.875</td>
<td>-40.0 220.0 -60.0 250.0 5.0 5.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>1.912 0.875 0.875</td>
<td>0.0 -65.0 250.0 5.0 2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>1.912 0.875 0.875</td>
<td>-40.0 220.0 -60.0 250.0 5.0 5.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>1.912 0.875 0.875</td>
<td>0.0 -65.0 250.0 5.0 2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>1.912 0.875 0.875</td>
<td>-40.0 220.0 -60.0 250.0 5.0 5.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>1.912 0.875 0.875</td>
<td>0.0 -65.0 250.0 5.0 2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>1.912 0.875 0.875</td>
<td>-40.0 257.0 -40.0 257.0 5.0 2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.365 0.750 0.750</td>
<td>0.0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.365 0.750 0.750</td>
<td>-40.0 257.0 -40.0 257.0 5.0 2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.365 0.750 0.750</td>
<td>0.0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.365 0.750 0.750</td>
<td>-40.0 257.0 -40.0 257.0 5.0 2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.365 0.750 0.750</td>
<td>0.0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.365 0.750 0.750</td>
<td>-40.0 257.0 -40.0 257.0 5.0 2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.365 0.750 0.750</td>
<td>0.0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.365 0.750 0.750</td>
<td>-40.0 257.0 -40.0 257.0 5.0 2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.365 0.750 0.750</td>
<td>0.0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.365 0.750 0.750</td>
<td>-40.0 257.0 -40.0 257.0 5.0 2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.365 0.750 0.750</td>
<td>0.0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.365 0.750 0.750</td>
<td>-40.0 257.0 -40.0 257.0 5.0 2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.365 0.750 0.750</td>
<td>0.0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>1.220 1.180 1.180</td>
<td>2.0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Resin pipe/Zinc' 'PBT'</td>
<td></td>
</tr>
<tr>
<td>1.220 1.180 1.180</td>
<td>2.8 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Resin pipe/Zinc' 'PBT'</td>
<td></td>
</tr>
<tr>
<td>3.190 0.750 0.750</td>
<td>-40.0 257.0 -40.0 257.0 15.0 2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.190 0.750 0.750</td>
<td>0.0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.190 0.750 0.750</td>
<td>-40.0 257.0 -40.0 257.0 15.0 2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.190 0.750 0.750</td>
<td>0.0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.190 0.750 0.750</td>
<td>-40.0 257.0 -40.0 257.0 15.0 2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.190 0.750 0.750</td>
<td>0.0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>3.650 1.000 1.000</td>
<td>-65.0 250.0 -65.0 250.0 5.0 0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
<tr>
<td>4.050 1.000 1.000</td>
<td>-65.0 250.0 -65.0 250.0 15.0 2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'Stainless Steel' 'Stainless Steel'</td>
<td></td>
</tr>
</tbody>
</table>
% Combine Pressure Sensor Arrays
PRESSSENSOR_DB = cat(2, PRESSSENSOR_DB1, PRESSSENSOR_DB2);

% Comments On the Bare Wire Environment - Use of the Thermocouple Material
PRESSSENSOR_DB{1,34} = 'Combination Pressure/Temperature Sensor. No silicone oil, no internal O-rings, no welds. Low static / thermal errors. Rugged design for harsh environments. Vacuum Calibration available.';
PRESSSENSOR_DB{2,34} = 'Combination Pressure/Temperature Sensor. No silicone oil, no internal O-rings, no welds. Low static / thermal errors. Rugged design for harsh environments. Vacuum Calibration available.';
PRESSSENSOR_DB{3,34} = 'Combination Pressure/Temperature Sensor. No silicone oil, no internal O-rings, no welds. Low static / thermal errors. Rugged design for harsh environments. Vacuum Calibration available.';
PRESSSENSOR_DB{4,34} = 'Combination Pressure/Temperature Sensor. No silicone oil, no internal O-rings, no welds. Low static / thermal errors. Rugged design for harsh environments. Vacuum Calibration available.';
PRESSSENSOR_DB{5,34} = 'Combination Pressure/Temperature Sensor. No silicone oil, no internal O-rings, no welds. Low static / thermal errors. Rugged design for harsh environments. Vacuum Calibration available.';
PRESSSENSOR_DB{6,34} = 'Low cost. Rugged Isolated Stainless Steel Design. Small size. Calibrated and temperature compensated. Applications involving corrosive liquids or gases.';
PRESSSENSOR_DB{7,34} = 'Low cost. Rugged Isolated Stainless Steel Design. Small size. Calibrated and temperature compensated. Applications involving corrosive liquids or gases.';
PRESSSENSOR_DB{8,34} = 'Low cost. Rugged Isolated Stainless Steel Design. Small size. Calibrated and temperature compensated. Applications involving corrosive liquids or gases.';
PRESSSENSOR_DB{9,34} = 'Low cost. Rugged Isolated Stainless Steel Design. Small size. Calibrated and temperature compensated. Applications involving corrosive liquids or gases.';
PRESSSENSOR_DB{10,34} = 'Low cost. Rugged Isolated Stainless Steel Design. Small size. Calibrated and temperature compensated. Applications involving corrosive liquids or gases.';
PRESSSENSOR_DB{11,34} = 'Low cost. Rugged Isolated Stainless Steel Design. Small size. Calibrated and temperature compensated. Applications involving corrosive liquids or gases.';
PRESSSENSOR_DB{12,34} = 'Low cost. Rugged Isolated Stainless Steel Design. Small size. Calibrated and temperature compensated. Applications involving corrosive liquids or gases.';
PRESSSENSOR_DB{13,34} = 'Low cost. Rugged Isolated Stainless Steel Design. Small size. Calibrated and temperature compensated. Applications involving corrosive liquids or gases.';
PRESSSENSOR_DB{14,34} = 'Low cost. Rugged Isolated Stainless Steel Design. Small size. Calibrated and temperature compensated. Applications involving corrosive liquids or gases.';
PRESSSENSOR_DB{15,34} = 'Low cost. Rugged Isolated Stainless Steel Design. Small size. Calibrated and temperature compensated. Applications involving corrosive liquids or gases.';
PRESSSENSOR_DB{16,34} = 'Miniature Differential Pressure Sensor.';
PRESSSENSOR_DB{17,34} = 'Miniature Differential Pressure Sensor.';
PRESSSENSOR_DB{18,34} = 'Low cost. Rugged Isolated Stainless Steel Design. Calibrated and temperature compensated. Oil-isolated housing. Measurement in hostile environments.';
PRESSSENSOR_DB{19,34} = 'Low cost. Rugged Isolated Stainless Steel Design. Calibrated and temperature compensated. Oil-isolated housing. Measurement in hostile environments.';
PRESSSENSOR_DB{20,34} = 'Low cost. Rugged Isolated Stainless Steel Design. Calibrated and temperature compensated. Oil-isolated housing. Measurement in hostile environments.';
PRESSSENSOR_DB{21,34} = 'Low cost. Rugged Isolated Stainless Steel Design. Calibrated and temperature compensated. Oil-isolated housing. Measurement in hostile environments.';
PRESSSENSOR_DB{22,34} = 'Low cost. Rugged Isolated Stainless Steel Design. Calibrated and temperature compensated. Oil-isolated housing. Measurement in hostile environments.';
PRESSSENSOR_DB{23,34} = 'Digitally enhanced, high accuracy output, coupled with high analog bandwidth.';
PRESSSENSOR_DB{24,34} = 'Digitally enhanced, high accuracy output, coupled with high analog bandwidth.';

% Build Up Database of Sensors Based On Operational Range
C = 1;
for p = 1:size(PRESSSENSOR_DB,1)
if (PLPRESS <= (PRESSSENSOR_DB{p,8} * Press_Cnvt))
    for a = 1:size(PRESSSENSOR_DB,2)
        P_MATRIX{c,a} = PRESSSENSOR_DB{p,a};
    end
    c = c+1;
end
if exist('P_MATRIX') == 0
    fprintf('

 *** WARNING ***
NO VIABLE PRESSURE SENSOR EXISTS IN DATABASE!

Other methods to determine Planetary Pressure must be used.') ;
end

% Down Select a Single Sensor if Multiple Options Exist
if size(P_MATRIX,1) > 1
% Select Sensors From Operational Temperature Range
C = 1;
for t = 1:size(P_MATRIX,1)
    if (PLTEMP >= P_MATRIX{t,18}) && (PLTEMP <= P_MATRIX{t,19})
        for a = 1:size(P_MATRIX,2)
            P_MATRIX2{c,a} = P_MATRIX{t,a};
        end
        c = c+1;
    end
end
% Copies P_MATRIX to P_MATRIX2 if All Sensors Eliminated Based on Temperature Range
if exist('P_MATRIX2') == 0
    P_MATRIX2 = P_MATRIX;
end
if size(P_MATRIX2,1) > 1
% Add Seasonal tolerances to Planetary Atmosphere and select Lowest Successful Sensor
PLPRESS2 = PLPRESS + 0.35 * PLPRESS;
C = 1;
for t = 1:size(P_MATRIX2,1)
    if (PLPRESS2 <= P_MATRIX2{t,8})
        for a = 1:size(P_MATRIX2,2)
            _MATRIX3{c,a} = P_MATRIX2{t, a};
        end
        c = c+1;
    end
end

% Copies P_MATRIX2 to P_MATRIX3 if all Sensors Eliminated
if exist('PMATRIX3') == 0
    P_MATRIX3 = P_MATRIX2;
end

if size(P_MATRIX3,1) > 1
    for c = 1:size(P_MATRIX3,1)
        Min_Des(c) = P_MATRIX3{c,3};
    end
    Min_Sens = min(Min_Des);
    k = 1;
    for c = 1:size(P_MATRIX3,1)
        if P_MATRIX3{c,8} == Min_Sens
            for j = 1:size(P_MATRIX3,2)
                P_MATRIX4{k,j} = P_MATRIX3{c , j} ;
            end
            k = k + 1;
        end
    end
end

% Copies P_MATRIX3 to P_MATRIX4 if all Sensors Eliminated
if exist('P_MATRIX4') == 0
    P_MATRIX4 = P_MATRIX3;
end

if size(P_MATRIX4,1) > 1
    % Create Mass Volume Scaling Factor Optimization
    for v = 1:size(P_MATRIX4,1)
        SCALE_FAC(v) = P_MATRIX4{v,27} * P_MATRIX4{v,28} * P_MATRIX4{v,29} * P_MATRIX4{v,30};
        if (P_MATRIX4{v,27} == 0) || (P_MATRIX4{v,28} == 0) || (P_MATRIX4{v,29} == 0) || (P_MATRIX4{v,30} == 0)
            SCALE_FAC(v) = 9.99E99;
        end
    end
    [ MIN_SCL, INDEX ] = min(SCALE_FAC);
    % Build Sensor Matrix Based on Scale Factors
    k = 1;
for c = 1:length(SCALE_FAC)
    if SCALE_FAC(c) == MIN_SCL
        for j = 1:size(P_MATRIX4,2)
            P_MATRIX5(k, j) = P_MATRIX4(c, j);
        end
        k = k + 1;
    end
end

% Copies P_MATRIX4 to P_MATRIX5 if all Sensors Eliminated
if exist('P_MATRIX5') == 0
    P_MATRIX5 = P_MATRIX4;
end

% Down Select a Single Sensor if Multiple Options Exist
if size(P_MATRIX5,1) > 1
    % Create Variable Array of Sensor Accuracy
    for a = 1:size(P_MATRIX5,1)
        ACCURACY(a) = P_MATRIX5(a, 2);
    end

    % Find the Sensors with the Tightest Accuracy Tolerances
    MIN_ACC = min(ACCURACY);

    % Build Sensor Matrix Based on Min Accuracy
    k = 1;
    for c = 1:size(P_MATRIX5,1)
        if P_MATRIX5(c, 2) == MIN_ACC
            for j = 1:size(P_MATRIX5,2)
                P_MATRIX6(k, j) = P_MATRIX5(c, j);
            end
            k = k + 1;
        end
    end
end

% Copies P_MATRIX5 to P_MATRIX6 if all Sensors Eliminated
if exist('P_MATRIX6') == 0
    P_MATRIX6 = P_MATRIX5;
end

% Adds Seasonal tolerances to Planetary Atmosphere and select Lowest Successful Sensor
if size(P_MATRIX4,1) > 1
    % Add Seasonal tolerances to Planetary Atmosphere and select Lowest Successful Sensor
    PLPRESS2 = PLPRESS + 0.25*PLPRESS;
    c = 1;
    for t = 1:size(P_MATRIX4,1)
        if (PLPRESS2 <= P_MATRIX4(t, 8))
            for a = 1:size(P_MATRIX4,2)
                P_MATRIX5(c, a) = P_MATRIX4(t, a);
            end
        end
end
% Copies P_MATRIX4 to P_MATRIX5 if all Sensors Eliminated
if exist('P_MATRIX5') == 0
    P_MATRIX5 = P_MATRIX4;
end

if size(P_MATRIX5,1) > 1
    for c = 1:size(P_MATRIX5,1)
        Min_Des(c) = P_MATRIX5{c,8};
    end
    Min_Sens = min(Min_Des);
    k = 1;
    for c = 1:size(P_MATRIX5,1)
        if P_MATRIX5{c,8} == Min_Sens
            for j = 1:size(P_MATRIX5,2)
                P_MATRIX6{k,j} = P_MATRIX5{c,j};
            end
            k = k + 1;
        end
    end
end

% Copies P_MATRIX5 to P_MATRIX6 if all Sensors Eliminated
if exist('P_MATRIX6') == 0
    P_MATRIX6 = P_MATRIX5;
else
    P_MATRIX6 = P_MATRIX;
end

if exist('P_MATRIX') == 0
    for wv = 1:size(PRESSSENSOR_DB,2)
        P_MATRIX6{1,wv} = 0.0;
    end
    P_MATRIX6{1,1} = 'N/A';
    P_MATRIX6{1,25} = 'N/A';
    P_MATRIX6{1,26} = 'N/A';
    P_MATRIX6{1,34} = 'N/A';
end

% Save Pressure Sensor Properties to Data File
save('PRESSURE_SENSOR_Data.mat');

save('PRESSURE_SENSOR_Final.mat','P_MATRIX6','PressUnitLabelStr');
% *** PRINT SUMMARY Output Section for 'Print' = 'Y'

if strcmpi(Print, 'Y') == 1

fprintf('

*** PRESSURE SENSOR SUMMARY RESULTS ***

*** INPUTS ***

Planetary Surface Pressure: %7.2f %s', PLPRESS, PressureUnit);
fprintf('
Unit System: %s', UNITS);

************************************************************************

% Pressure Sensor Properties

fprintf('

% Pressure Sensor Type: %s', P_MATRIX6{1,1});
fprintf('
Pressure Sensor Accuracy: %10.3f %s', P_MATRIX6{1,2}, PressUnitLabelStr{2});
fprintf('
Pressure Sensor Stability: %10.3f %s', P_MATRIX6{1,3}, PressUnitLabelStr{3});
fprintf('
Over Pressure Rated Value: %10.3f %s', P_MATRIX6{1,4}, PressUnitLabelStr{4});
fprintf('
Burst Pressure Rating: %10.3f %s', P_MATRIX6{1,5}, PressUnitLabelStr{5});
fprintf('
Life Cycles: %14.3E %s', P_MATRIX6{1,6}, PressUnitLabelStr{6});
fprintf('
Sensing Pressure Range - Low: %10.3f %s', P_MATRIX6{1,7}, PressUnitLabelStr{7});
fprintf('
High : %10.3f %s', P_MATRIX6{1,8}, PressUnitLabelStr{8});
fprintf('
Proof Pressure: %10.3f %s', P_MATRIX6{1,9}, PressUnitLabelStr{9});
fprintf('
Burst Pressure: %10.3f %s', P_MATRIX6{1,10}, PressUnitLabelStr{10});

% Environmental Properties

fprintf('

% Maximum Shock Load: %10.3f %s', P_MATRIX6{1,11}, PressUnitLabelStr{11});
fprintf('
Shock Impulse Time: %10.3f %s', P_MATRIX6{1,12}, PressUnitLabelStr{12});
fprintf('
Vibration Shock Limit: %10.3f %s', P_MATRIX6{1,13}, PressUnitLabelStr{13});
fprintf('
Operating Frequency Range - Low: %10.3f %s', P_MATRIX6{1,14}, PressUnitLabelStr{14});
fprintf('
High: %10.3f %s', P_MATRIX6{1,15}, PressUnitLabelStr{15});
fprintf('
Operating Humidity Range - Low: %10.3f %s', P_MATRIX6{1,16}, PressUnitLabelStr{16});
fprintf('
High: %10.3f %s', P_MATRIX6{1,17}, PressUnitLabelStr{17});
fprintf('
Operating Temperature Range - Low: %10.3f %s', P_MATRIX6{1,18}, PressUnitLabelStr{18});
fprintf('
High: %10.3f %s', P_MATRIX6{1,19}, PressUnitLabelStr{19});
fprintf('
Storage Temperature Range - Low: %10.3f %s', P_MATRIX6{1,20}, PressUnitLabelStr{20});
fprintf('
High: %10.3f %s', P_MATRIX6{1,21}, PressUnitLabelStr{21});

% Power Requirements

fprintf('

% Input Voltage: %10.3f %s', P_MATRIX6{1,22}, PressUnitLabelStr{22});
fprintf('
Input Amperage: %10.3f %s', P_MATRIX6{1,23}, PressUnitLabelStr{23});
fprintf('
Sensing Bandwidth: %10.3f %s', P_MATRIX6{1,24}, PressUnitLabelStr{24});

% Physical Properties

fprintf('

% Sensor Material: %s', P_MATRIX6{1,25});
fprintf('
Case Material: %s', P_MATRIX6{1,26});
fprintf('
Dimensions - Length: %10.3f %s', P_MATRIX6{1,27}, PressUnitLabelStr{27});
fprintf('
Width: %10.3f %s', P_MATRIX6{1,28}, PressUnitLabelStr{28});
fprintf('
Height: %10.3f %s', P_MATRIX6{1,29}, PressUnitLabelStr{29});
fprintf('
Sensor Mass: %10.3f %s', P_MATRIX6{1,30}, PressUnitLabelStr{30});
% Temperature Sensing Range
fprintf('

 Temperature Sensing Range
 Lower Sensing Limit: %10.3f
 Upper Sensing Limit: %10.3f
 Accuracy: %10.3f

 Sensor Comments:
%s', P_MATRIX6{1,34});

end

%***********************************************************************
% Writes a text file summary of the Planetary Properties
Fout = fopen('PRESSURE_SENSOR_Summary.txt', 'w+');
fprintf(Fout, 'PRESSURE SENSOR SUMMARY RESULTS

*** INPUTS ***
Planetary Surface Pressure: %7.2f %s
Unit System: %s

************************************************************************

Environmental Properties

Maximum Shock Load: %10.3f %s, P_MATRIX6{1,11}, PressUnitLabelStr(11);
Shock Impulse Time: %10.3f %s, P_MATRIX6{1,12}, PressUnitLabelStr(12);
Vibration Shock Limit: %10.3f %s, P_MATRIX6{1,13}, PressUnitLabelStr(13);
Operating Frequency Range - Low: %10.3f %s, P_MATRIX6{1,14}, PressUnitLabelStr(14);
Operating Frequency Range - High: %10.3f %s, P_MATRIX6{1,15}, PressUnitLabelStr(15);
Operating Humidity Range - Low: %10.3f %s, P_MATRIX6{1,16}, PressUnitLabelStr(16);
Operating Humidity Range - High: %10.3f %s, P_MATRIX6{1,17}, PressUnitLabelStr(17);
Operating Temperature Range - Low: %10.3f %s, P_MATRIX6{1,18}, PressUnitLabelStr(18);
Operating Temperature Range - High: %10.3f %s, P_MATRIX6{1,19}, PressUnitLabelStr(19);
Storage Temperature Range - Low: %10.3f %s, P_MATRIX6{1,20}, PressUnitLabelStr(20);
Storage Temperature Range - High: %10.3f %s, P_MATRIX6{1,21}, PressUnitLabelStr(21);

Power Requirements

%***********************************************************************

Power Requirements

%***********************************************************************

end
fprintf(Fout, 'n Input Voltage: %.3f %s', P_MATRIX6{1,22}, PressUnitLabelStr{22});
fprintf(Fout, 'n Input Amperage: %.3f %s', P_MATRIX6{1,23}, PressUnitLabelStr{23});
fprintf(Fout, 'n Sensing Bandwidth: %.3f %s', P_MATRIX6{1,24}, PressUnitLabelStr{24});

fprintf(Fout, 'n\n Physical Properties');
fprintf(Fout, 'n Sensor Material: %s', P_MATRIX6{1,25});
fprintf(Fout, 'n Case Material: %s', P_MATRIX6{1,26});
fprintf(Fout, 'n Dimensions - Length: %.3f %s', P_MATRIX6{1,27}, PressUnitLabelStr{27});
fprintf(Fout, 'n Width: %.3f %s', P_MATRIX6{1,28}, PressUnitLabelStr{28});
fprintf(Fout, 'n Height: %.3f %s', P_MATRIX6{1,29}, PressUnitLabelStr{29});
fprintf(Fout, 'n Sensor Mass: %.3f %s', P_MATRIX6{1,30}, PressUnitLabelStr{30});

% Temperature Sensing Range
fprintf(Fout, 'n\n Temperature Sensing Range');
fprintf(Fout, 'n Lower Sensing Limit: %.3f %s', P_MATRIX6{1,31}, PressUnitLabelStr{31});
fprintf(Fout, 'n Upper Sensing Limit: %.3f %s', P_MATRIX6{1,32}, PressUnitLabelStr{32});
fprintf(Fout, 'n Accuracy: %.3f %s', P_MATRIX6{1,33}, PressUnitLabelStr{33});

fprintf(Fout, 'n\n Sensor Comments:');
fprintf(Fout, 'n %s', P_MATRIX6{1,34});
fclose(Fout);

D15. Radiation Sensors

function RADIATION_SENSORS(RAD_TYPE, UNITS, Print)
% Radiation Sensor Database
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
% This file contains Radiation Sensor values for the ISSPO program.
% Data here is loaded into the main program to determine the proper sensor
% to use for Missions based on the sensor requirements.
% Values are assigned to a variable name and saved to a MatLAB '.mat' file
% and loaded when calling the ISSPO program.
% Some of the values are based on the Unit system chosen and are loaded
% in separate sections.
% References
% http://www.psicorp.com/products/lpd-spectrometer.shtml
% http://www.blackcatsystems.com/GM/page5.html
% http://www.seintl.com/products/monitor_4.html
% http://www.pnwx.com/Equipment/Test/SurveyMeters/ASM-990/
% http://www.evproducts.com/pdf/capture_plus.pdf
function RADIATION_SENSORS(RAD_TYPE,UNITS,Print)

% Load Constants values into Local Program
load('Constants_DB.mat');
load('Planet Data.mat');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%%%%%%%%
%% RADIATION Sensor Data Base Created Based on un Unit System
%% 2 - D Matrix RADIATIONSENSOR_DB(DS_Type, Property)

if strcmpi(UNITS, 'SI') == 1

% Units
% # of Bins Low keV High # of Bins Low MeV High # of Bins Low keV High # of Bins Low MeV High
# of Bins Low keV High # of Bins Low MeV High # of Bins Low keV High # of Bins Low MeV High
Wi d   Hot   Hea t   W   1   RADIATIONSENSOR_DB = { 'LPD' [ 'Charged Particle' ] 1.00 250.00 15 0.50
20.00 7 6.00 250.00 6 1.50 0 100 [ 0.05 0.1 0.3 ] 7.00 150 150
0 0 0 0 0 0 150 0 0
300 15.0 ;

'LPD-HECPS' { 'Charged Particle' } 1.00 150.00 5 0.30
10.00 5 12.00 0 0 0 3.00 0 1 0 0
0 0 0 0 0 0 200 [ 0.2 ] 3.00 150 150
120 2.0 ;

'GM-180' { 'Alpha' 'Beta' 'Gamma' 'X-Ray' } 0 0 0 0 0 0
0 0 3.00 0 0 0 0 0 0 0
1 7.00 0 0 0 0 0 0 0 0
0 0 3.00 0 0 0 0 0 0 0
38 0 ;

'M4EC' { 'Alpha' 'Beta' 'Gamma' 'X-Ray' } 0 0 0 0 0 0
0 0 2.50 0 0 0 0 0 0 0
1 10.00 1300.0 1 0 0 0 0 0 0
0 0 1300.0 0 0 0 0 0 0 0
48 0 ;

'ASM-993' { 'Alpha' 'Beta' 'Gamma' } 0 0 0 0 0
0 0 3.50 0 0 0 0 0 0 0
1 6.00 0 0 0 0 0 0 0 0
104.7 63.5 0 ;

'CAPture Plus' { 'Gamma' } 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 30.00 1300.0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
15 0 ;

'X-123' { 'X-Ray' } 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
25 1.2 ;

'PM1401GNB' { 'Charged Particle' 'Gamma' } 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```plaintext
if strcmpi(UNITS,'British') == 1
% Properties
% DETECTOR TYPE PROTON ENERGY RANGE PROTON BINS ELECTRON
ENERGY RANGE ELECTRON BINS ALPHA PARTICLE RANGE ALPHA BINS NUCLEON ION RANGE HEAVY ION BINS BETA PARTICLE
RANGE BETA BINS GAMMA RAY RANGE GAMMA BINS X-RAY RANGES X-RAY BINS PARTICLE COUNT G-FACTOR WEIGHT
DIMENSIONS - in POWER

% Units
% High # of Bins Low MeV High # of Bins Low MeV High # of Bins Low keV High # of Bins Low keV High
RADIATIONSENSOR_DB = {'LPD', 'Charged Particle'}, 1.00 250.00 15 0.50
7 6.00 5.00 0 0 0 0 100 [7.75 15.5 46.5] 246.92 5.91 5.91
11.81 51.182 ;

'LPD-HECPS' {'Charged Particle'}, 1.00 150.00 5 0.30
12.00 0 0 0 0 0 3.00 0 0 0
0 0 0 0 0 0 200 [31.0] 105.82 5.91 5.91
20.00 7 6.00 250.00 6 1.50 0 1 0 0
4.73 6.824 ;

'GM-180' {'Alpha', 'Beta', 'Gamma', 'X-Ray'}, 0 0 0 0 0 50.00 0
1 7.00 0 0 7.00 0 1 0 0 0
1.50 0 ;

'M4EC' {'Alpha', 'Beta', 'Gamma', 'X-Ray'}, 0 0 0 0 0 50.00 0
1 10.00 1300.0 1 10.00 1300.0 1 0 0 [0] 8.70 8.25 2.75
1.875 0 ;

'ASM-993' {'Alpha', 'Beta', 'Gamma'}, 0 0 0 0 0 35.00 0
1 6.00 0 1 0 0 83.333 [0] 38.40 10.91
4.125 2.50 0 ;

'CAPture Plus' {'Gamma'}, 0 0 0 0 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
1.00 4.094 ;

'PM1401GNB' {'Charged Particle', 'Gamma'}, 0 0 0 0 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
0 0 0 0 0 0 0 [0] 0 0 0
1.30 0 ;
```

% Comments on Features / Use
RADIATIONSENSOR_DB{1,31} = {'16 Different Operational Modes'; 'Excellent Particle Specificity'; 'High Level Of Redundancy'; 'Radiation Hardened Components'; 'Fully Flight Qualified: URTS E, MDS-1, and ISS'};

RADIATIONSENSOR_DB{2,31} = {'Custom configurable energy bin structure. Baseline configuration given'; 'Energy resolution: dE/E < 0.3 (fwhm)'; 'Excellent particle discrimination'; 'Fully Flight Qualified: 1000 g shock, 20 g vibration, -55 to +85 storage temp (deg F), -55 to +60 operation temp (deg F)' ;

Two Versions: RS232 interface, USB port/power'; 'User selectable integration times'; 'Geiger Counter clicking may be enabled'; 'Built in graphing/statistics package'; 'Trend line may be superimposed over graphed data'};

RADIATIONSENSOR_DB{4,31} = {'Alpha - typical detection efficiency at 3.6 MeV > 80%'; 'Beta - 50 keV with 35% efficiency, 150 keV with 75% efficiency'; 'Gamma / X-Ray - 10 keV through window, 40 keV minimum through sidewall'; 'Power Requirement: 1 9-V battery 2000hrs lifetime, Wt without battery'; 'Accuracy -±15% of full scale (referenced to Cs-137)'; 'Available with radhard or COTS components'; 'Available as single-sided or fully redundant system'};

RADIATIONSENSOR_DB{5,31} = {'Two Versions: RS232 interface, USB port/power'; 'User selectable integration times'; 'Geiger Counter clicking may be enabled'; 'Built in graphing/statistics package'; 'Trend line may be superimposed over graphed data'};

RADIATIONSENSOR_DB{6,31} = {'CZT Gamma-Ray Spectrometer'; 'Energy Resolution: <5% FWHM @ 122keV <3.0% @ 662keV'; 'Power Requirement: AC or DC coupled, +/- HV supply, 1500V Nominal'; 'Temperature Range: -10° to +35° C Standard'};

RADIATIONSENSOR_DB{7,31} = {'Efficiency is >25% for X-rays from 1.5 to 25 keV. May be used outside this range with lower efficiency'; 'Energy Resolution: 145 to 260 eV FWHM @5.9 keV. Depends on detector, peaking time, and temperature'; 'Power Requirement: Nominal Input +/- VDC at 250 mA (1.2 W)'; 'High Voltage Supply Internal multiplier, adjustable to 400 V'; 'Communications - USB 1.1 (2.0 compatible) at full speed (12 mbps), Serial Standard RS232 at up to 57.6 kbaud'};

RADIATIONSENSOR_DB{8,31} = {'Detector Type - Gamma Ray and Neutron'; 'Detects and identifies Special Nuclear Materials, Naturally Occurring Radioactive Materials (NORM), and Industrial radionuclides'; 'Power Requirement: One AA Battery 800hr life'; 'Communications - PC Communication node (IR, Bluetooth)'};
Identifies: 233U, 235U, 237Np, 237Th, 238U, 239Pu, 67Ga, 51Cr, 75Se, 99mTc, 103Pd, 111In, 123I, 131I, 201Ti, 133Xe, 40K, 226Ra, 232Th, 238U, 57Co, 60Co, 133Ba, 137Cs, 192Ir, 204Ti, 226Ra, 241Am

RAD_MATRIX = RADIATIONSENSOR_DB;

% Down Select Sensors Based on Radiation Types
for loc = 1:length(RAD_TYPE)
    STR1 = RAD_TYPE{loc};
    if strcmpi(STR1, 'Charged Particle') == 1
        c = 1;
        FLAG = 0;
        for r = 1:size(RAD_MATRIX,1)
            for pos = 1:length(RAD_MATRIX{r,2})
                STR2 = RAD_MATRIX{r,2}{pos};
                if strcmpi(STR1,STR2) == 1
                    for s = 1:size(RAD_MATRIX,2)
                        RAD_MATRIX_TMP{c,s} = RAD_MATRIX{r,s};
                    end
                    FLAG = 1;
                end
            end
        end
        if FLAG == 1
            c = c + 1;
        end
        FLAG = 0;
    end
    if exist('RAD_MATRIX_TMP') == 0
        if strcmpi(Print,'Y') == 1
            fprintf('

*** ERROR: NULL DATABASE ***
No Sensor Exist in Database which Meet Radiation Requirements!
Radiation Type: %s', STR1);
            RAD_MATRIX_TMP = RAD_MATRIX;
        end
    end
    clear RAD_MATRIX;
end
if strcmpi(STR1, 'Alpha') == 1
    c = 1;
    FLAG = 0;
    for r = 1:size(RAD_MATRIX,1)
        for pos = 1:length(RAD_MATRIX{r,2})
            STR2 = RAD_MATRIX{r,2}{pos};
            if strcmpi(STR1,STR2) == 1
                for s = 1:size(RAD_MATRIX,2)
                    RAD_MATRIX_TMP{c,s} = RAD_MATRIX{r,s};
                end
            end
        end
    end
end

318
FLAG = 1;
end
if FLAG == 1
C = C + 1;
end
FLAG = 0;
end
if exist('RAD_MATRIX_TMP') == 0
if strcmpi(Print,'Y') == 1
fprintf('

 *** ERROR: NULL DATABASE ***
No Sensor Exist in Database which Meet Radiation Requirements!
Image Type: %s', STR1);
end
RAD_MATRIX_TMP = RAD_MATRIX;
clear RAD MATRIX_TMP;
end
if strcmpi(STR1, 'Beta') == 1
if strcmpi(STR1, STR2) == 1
for r = 1:size(RAD_MATRIX,1)
for pos = 1:length(RAD_MATRIX{r,2})
STR2 = RAD_MATRIX{r,2}{pos};
if strcmpi(STR1,STR2) == 1
for s = 1:size(v,2)
RAD_MATRIX_TMP{c,s} = RAD_MATRIX{r,s};
end
FLAG = 1;
end
if FLAG == 1
C = C + 1;
end
FLAG = 0;
end
if exist('RAD_MATRIX_TMP') == 0
if strcmpi(Print,'Y') == 1
fprintf('

 *** ERROR: NULL DATABASE ***
No Sensor Exist in Database which Meet Radiation Requirements!
Image Type: %s', STR1);
end
RAD_MATRIX_TMP = RAD_MATRIX;
clear RAD MATRIX_TMP;
end
if strcmpi(STR1, 'Beta') == 1
if strcmpi(STR1, STR2) == 1
for r = 1:size(RAD_MATRIX,1)
for pos = 1:length(RAD_MATRIX{r,2})
STR2 = RAD_MATRIX{r,2}{pos};
if strcmpi(STR1,STR2) == 1
for s = 1:size(v,2)
RAD_MATRIX_TMP{c,s} = RAD_MATRIX{r,s};
end
FLAG = 1;
end
if FLAG == 1
C = C + 1;
end
FLAG = 0;
end
if exist('RAD_MATRIX_TMP') == 0
if strcmpi(Print,'Y') == 1
fprintf('

 *** ERROR: NULL DATABASE ***
No Sensor Exist in Database which Meet Radiation Requirements!
Image Type: %s', STR1);
end
RAD_MATRIX_TMP = RAD_MATRIX;
clear RAD MATRIX_TMP;
end
if strcmpi(STR1, 'Beta') == 1
if strcmpi(STR1, STR2) == 1
for r = 1:size(RAD_MATRIX,1)
for pos = 1:length(RAD_MATRIX{r,2})
STR2 = RAD_MATRIX{r,2}{pos};
if strcmpi(STR1,STR2) == 1
for s = 1:size(v,2)
RAD_MATRIX_TMP{c,s} = RAD_MATRIX{r,s};
end
FLAG = 1;
end
if FLAG == 1
C = C + 1;
end
FLAG = 0;
end
if exist('RAD_MATRIX_TMP') == 0
if strcmpi(Print,'Y') == 1
fprintf('

 *** ERROR: NULL DATABASE ***
No Sensor Exist in Database which Meet Radiation Requirements!
Image Type: %s', STR1);
end
RAD_MATRIX_TMP = RAD_MATRIX;
clear RAD MATRIX_TMP;
end
if strcmpi(STR1, 'Beta') == 1
if strcmpi(STR1, STR2) == 1
for r = 1:size(RAD_MATRIX,1)
for pos = 1:length(RAD_MATRIX{r,2})
STR2 = RAD_MATRIX{r,2}{pos};
if strcmpi(STR1,STR2) == 1
for s = 1:size(v,2)
RAD_MATRIX_TMP{c,s} = RAD_MATRIX{r,s};
end
FLAG = 1;
end
if FLAG == 1
C = C + 1;
end
FLAG = 0;
end
if exist('RAD_MATRIX_TMP') == 0
if strcmpi(Print,'Y') == 1
fprintf('

 *** ERROR: NULL DATABASE ***
No Sensor Exist in Database which Meet Radiation Requirements!
Image Type: %s', STR1);
end
RAD_MATRIX_TMP = RAD_MATRIX;
clear RAD MATRIX_TMP;
end
if strcmpi(STR1, 'Beta') == 1
if strcmpi(STR1, STR2) == 1
for r = 1:size(RAD_MATRIX,1)
for pos = 1:length(RAD_MATRIX{r,2})
STR2 = RAD_MATRIX{r,2}{pos};
if strcmpi(STR1,STR2) == 1
for s = 1:size(v,2)
RAD_MATRIX_TMP{c,s} = RAD_MATRIX{r,s};
end
FLAG = 1;
end
if FLAG == 1
C = C + 1;
end
FLAG = 0;
end
if exist('RAD_MATRIX_TMP') == 0
if strcmpi(Print,'Y') == 1
fprintf('

 *** ERROR: NULL DATABASE ***
No Sensor Exist in Database which Meet Radiation Requirements!
Image Type: %s', STR1);
end
RAD_MATRIX_TMP = RAD_MATRIX;
clear RAD MATRIX_TMP;
end
if strcmpi(STR1, 'Gamma') == 1
    c = 1;
    FLAG = 0;
    for r = 1:size(RAD_MATRIX,1)
        for pos = 1:length(RAD_MATRIX{r,2})
            STR2 = RAD_MATRIX{r,2}{pos};
            if strcmpi(STR1,STR2) == 1
                for s = 1:size(RAD_MATRIX,2)
                    RAD_MATRIX_TMP{c,s} = RADMATRIX{r,s};
                end
                FLAG = 1;
            end
        end
        if FLAG == 1
            c = c + 1;
        end
        FLAG = 0;
    end
    if exist('RAD_MATRIX_TMP') == 0
        if strcmpi(Print,'Y') == 1
            fprintf('

 *** ERROR: NULL DATABASE ***

 No Sensor Exist in Database which Meet Radiation Requirements!

 Image Type: %s', STR1);
            RAD_MATRIX_TMP = RAD_MATRIX;
            clear RAD_MATRIX_TMP;
        end
    end
end
if strcmpi(STR1, 'X-Ray') == 1
    c = 1;
    FLAG = 0;
    for r = 1:size(RAD_MATRIX,1)
        for pos = 1:length(RAD_MATRIX{r,2})
            STR2 = RAD_MATRIX{r,2}{pos};
            if strcmpi(STR1,STR2) == 1
                for s = 1:size(RAD_MATRIX,2)
                    RAD_MATRIX_TMP{c,s} = RAD_MATRIX{r,s};
                end
                FLAG = 1;
            end
        end
        if FLAG == 1
            c = c + 1;
        end
        FLAG = 0;
    end
end
if exist('RAD_MATRIX_TMP') == 0
    if strcmpi(Print,'Y') == 1
        fprintf('

 *** ERROR: NULL DATABASE ***
 No Sensor Exist in Database which Meet Radiation Requirements!

 Image Type: %s ' , STR1);
    end
end
end
end
clear RAD_MATRIX;
RAD_MATRIX = RAD_MATRIX_TMP;
clear RAD_MATRIX_TMP;
end

% Sort Sensor by Operational Range Of Different Radiation Types
if size(RAD_MATRIX,1) > 1
    % Create Summary Operational Range Array
    SUM_RAD_ARRAY = zeros(size(RAD_MATRIX,1),1);
    for rad = 1:length(RAD_TYPE)
        if strcmpi(RAD_TYPE{rad},'Charged Particle') == 1
            % Add up Proton Energy Range
            for p = 1:size(RAD_MATRIX,1)
                SUM_RAD_ARRAY{p} = SUM_RAD_ARRAY{p} + RAD_MATRIX{p,4} - RAD_MATRIX{p,3};
            end
            % Add up Electron Energy Range
            for e = 1:size(RAD_MATRIX,1)
                SUM_RAD_ARRAY{e} = SUM_RAD_ARRAY{e} + RAD_MATRIX{e,7} - RAD_MATRIX{e,6};
            end
            % Add up Alpha Particle Energy Range
            for a = 1:size(RAD_MATRIX,1)
                SUM_RAD_ARRAY{a} = SUM_RAD_ARRAY{a} + RAD_MATRIX{a,10} - RAD_MATRIX{a,9};
            end
        end
        if strcmpi(RAD_TYPE{rad},'Alpha') == 1
            % Add up Alpha Particle Energy Range
            for a = 1:size(RAD_MATRIX,1)
                SUM_RAD_ARRAY{a} = SUM_RAD_ARRAY{a} + RAD_MATRIX{a,10} - RAD_MATRIX{a,9};
            end
end
end
if strcmpi(RAD_TYPE{rad},'Beta') == 1
% Add up Alpha Particle Energy Range
for b = 1:size(RAD_MATRIX,1)
    SUM_RAD_ARRAY(b) = SUM_RAD_ARRAY(b) + RAD_MATRIX{b, 16}
end
end
if strcmpi(RAD_TYPE{rad},'Gamma') == 1
% Add up Alpha Particle Energy Range
for g = 1:size(RAD_MATRIX,1)
    SUM_RAD_ARRAY(g) = SUM_RAD_ARRAY(g) + RAD_MATRIX{g, 19}
end
end
if strcmpi(RAD_TYPE{rad},'X-Ray') == 1
% Add up Alpha Particle Energy Range
for a = 1:size(RAD_MATRIX,1)
    SUM_RAD_ARRAY(a) = SUM_RAD_ARRAY(a)
end
end
% Find the Maximum Operational Range sensor
[RAD_MAX_SCL, INDEX] = max(SUM_RAD_ARRAY);
% For Multiple Sensors with the Same Range Reduce Sensor Matrix to Those
pos = 1;
for r = 1:length(SUM_RAD_ARRAY)
    if SUM_RAD_ARRAY(r) == RAD_MAX_SCL
        for j = 1:size(RAD_MATRIX,2)
            RAD_MATRIX2{pos, j} = RAD_MATRIX{r, j};
        end
        pos = pos + 1;
    end
end
% Create Volume Scaling Factor Optimization
for v = 1:size(RAD_MATRIX2,1)
    RAD_SCALE_FAC(v) = RAD_MATRIX2{v, 27} * RAD_MATRIX2{v, 28} * RAD_MATRIX2{v, 29};
end
RAD_SCALE_FAC(v) = 9.99E9;
end

[ RAD_MIN_SCL, INDEX ] = min(RAD_SCALE_FAC);

% Build Sensor Matrix Based on Scale Factors
k = 1;
for c = 1:length(RAD_SCALE_FAC)
    if RAD_SCALE_FAC(c) == RAD_MIN_SCL
        for j = 1:size(RAD_MATRIX2,2)
            RAD_MATRIX3{k,j} = RAD_MATRIX2{c,j};
        end
        k = k + 1;
    end
end

% Copies RAD_MATRIX2 to RAD_MATRIX3 if all Sensors Eliminated
if exist('RAD_MATRIX3') == 0
    RAD_MATRIX3 = RAD_MATRIX2;
end

% Save Density Sensor Properties to Data File
save('RADIATION_SENSOR_Data.mat');
save('RADIATION_SENSOR_Final.mat','RAD_MATRIX3','RadUnitLabelStr');

% *** PRINT SUMMARY Output Section for 'Print' = 'Y'
if strcmpi(Print,'Y') == 1
    fprintf('

 *** RADIATION SENSOR
 *** INPUTS ***
 Unit System: %s', UNITS);

* ******************************************************

% Radiation Sensor Properties
fprintf('

** Sensor Properties **
 Radiation Sensor Type: %s', RAD_MATRIX3{1,1});
fprintf('
 Radiation Detected: %s', RAD_MATRIX3{1,2});
for r = 1:length(RAD_MATRIX3{1,2})
    fprintf(' %s', RAD_MATRIX3{1,2}{r} );
end
fprintf('
 Proton Energy Range - Low: %lld %s', RAD_MATRIX3{1,3}, RadUnitLabelStr{3} );
fprintf('
 High: %lld %s', RAD_MATRIX3{1,4}, RadUnitLabelStr{4} );
fprintf('
 Number of Detection Bins: %lld %s', RAD_MATRIX3{1,5}, RadUnitLabelStr{5} );
fprintf('
 Electron Energy Range - Low: %lld %s', RAD_MATRIX3{1,6}, RadUnitLabelStr{6} );
fprintf('
 High: %lld %s', RAD_MATRIX3{1,7}, RadUnitLabelStr{7} );

* ******************************************************

for g = 1:length(RAD_MATRIX3{1,25})
    fprintf(' %s', RAD_MATRIX3{1,25}(g));
end
fprintf('

 Physical Properties');
fprintf('
 Sensor Mass: %11.3f %s', RAD_MATRIX3{1,1}, RadUnitLabelStr{1});
fprintf('
 Dimensions - Length: %11.3f %s', RAD_MATRIX3{1,27}, RadUnitLabelStr{27});
fprintf('
 Width: %11.3f %s', RAD_MATRIX3{1,28}, RadUnitLabelStr{28});
fprintf('
 Height: %11.3f %s', RAD_MATRIX3{1,29}, RadUnitLabelStr{29});
fprintf('
 Power Requirement: %11.3f %s', RAD_MATRIX3{1,30}, RadUnitLabelStr{30});

fprintf('

 Sensor Comments: ');
for r = 1:length(RAD_MATRIX3{1,31})
    fprintf(' %s', RAD_MATRIX3{1,31}(r));
end
end

%***********************************************************************
%***  Writes a text file summary of the Planetary Properties

Fout = fopen('RADIATION_SENSOR_Summary.txt','w+');
fprintf(Fout, '

 *** RADIATION SENSOR SUMMARY RESULTS ***

*** INPUTS ***
Unit System: %s', UNITS);
% Radiation Sensor Properties
fprintf(Fout, '

 %s', RAD_MATRIX3{1,2}, RadUnitLabelStr{2});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,3}, RadUnitLabelStr{3});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,4}, RadUnitLabelStr{4});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,5}, RadUnitLabelStr{5});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,6}, RadUnitLabelStr{6});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,7}, RadUnitLabelStr{7});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,8}, RadUnitLabelStr{8});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,9}, RadUnitLabelStr{9});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,10}, RadUnitLabelStr{10});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,11}, RadUnitLabelStr{11});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,12}, RadUnitLabelStr{12});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,13}, RadUnitLabelStr{13});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,14}, RadUnitLabelStr{14});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,15}, RadUnitLabelStr{15});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,16}, RadUnitLabelStr{16});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,17}, RadUnitLabelStr{17});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,18}, RadUnitLabelStr{18});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,19}, RadUnitLabelStr{19});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,20}, RadUnitLabelStr{20});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,21}, RadUnitLabelStr{21});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,22}, RadUnitLabelStr{22});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,23}, RadUnitLabelStr{23});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,24}, RadUnitLabelStr{24});
fprintf(Fout, '
 %s', RAD_MATRIX3{1,25}, RadUnitLabelStr{25});
fprintf(Fout, '

 Sensor Comments: ');
for r = 1:length(RAD_MATRIX3{1,31})
    fprintf(' %s', RAD_MATRIX3{1,31}(r));
end

end
fprintf(Fout, ' Radiation Detected: 
');
for r = 1:length(RAD_MATRIX3{1,2})
  fprintf(Fout, ' %s', RAD_MATRIX3{1,2}{r});
end
fprintf(Fout, ' Proton Energy Range - Low: %11.3f High: %11.3f 
Number of Detection Bins: %lld 
Electron Energy Range - Low: %11.3f High: %11.3f 
Number of Detection Bins: %lld 
Alpha Particle Range - Low: %11.3f High: %11.3f 
Number of Detection Bins: %lld 
Nucleon/Heavy Ion Range - Low: %11.3f High: %11.3f 
Number of Detection Bins: %lld 
Beta Particle Range - Low: %11.3f High: %11.3f 
Number of Detection Bins: %lld 
Gamma Particle Range - Low: %11.3f High: %11.3f 
Number of Detection Bins: %lld 
X-Ray Energy Range - Low: %11.3f High: %11.3f 
Number of Detection Bins: %lld 
Particle Detection Rate: %ll.3f 
G-Factor: 
');
for g = 1:length(RAD_MATRIX3{1,25})
  fprintf(Fout, ' %6.3f , RAD_MATRIX3{1,25}{g} 
end
fclose(Fout);

fprintf(Fout, ' Physical Properties 
Sensor Mass: %11.3f %s', RAD_MATRIX3{1,26}, RadUnitLabelStr{26});
fprintf(Fout, ' Dimensions - Length: %11.3f %s', RAD_MATRIX3{1,27}, RadUnitLabelStr{27});
fprintf(Fout, ' Width: %11.3f %s', RAD_MATRIX3{1,28}, RadUnitLabelStr{28});
fprintf(Fout, ' Height: %11.3f %s', RAD_MATRIX3{1,29}, RadUnitLabelStr{29});
fprintf(Fout, ' Power Requirement: %11.3f %s', RAD_MATRIX3{1,30}, RadUnitLabelStr{30});

fprintf(Fout, ' Sensor Comments: 
');
for r = 1:length(RAD_MATRIX3{1,31})
  fprintf(Fout, ' %s', RAD_MATRIX3{1,31}{r});
end
fclose(Fout);
D16. Temperature Sensors

% function TEMP_SENSORS(TEMP_SENS_TYPE, PL TEMP, UNITS, PLNAME, Print)
% Temperature Sensor Database
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
% This file contains Temperature Sensor values for the ISSPO program.
% Data here is loaded into the main program to determine the proper sensor
% to use for Missions based on the planetary Atmosphere composition.
% Values are assigned to a variable name and saved to a MatLAB '.mat' file
% and loaded when calling the ISSPO program.
% Some of the values are based on the Unit system chosen and are loaded
% in separate sections.
% References
% http://www.omega.com/guides/thermocouples.html
% http://www.lakeshore.com/temp/sen/prtdts.html
% http://www.sensors.goodrich.com/literature/lit_pdf_s/4025_Sensor_16MDD.pdf
% http://www.rdfcorp.com/products/aerospace/aerospace.shtml

function TEMPERATURE_SENSORS(TEMP_SENS_TYPE, PL TEMP, UNITS, PLNAME, Print)

% Define Internal Program Variables
Poor = 1;
Fair = 2;
Good = 3;
Excellent = 4;

% Load Constants values into Local Program
load('Constants_DB.mat');
load('Planet Data.mat');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Temperature Sensor Data Base Created Based on un Unit System
%% 2 - D Matrix TEMPSSENSOR_DB(T_Type, Property)

if strcmpi(UNITS,'SI') == 1

% Properties
% SENSOR COLD LIMIT HOT LIMIT POWER REQ MASS REQ DIMENSIONS (mm) PRESSURE
ERROR TOL ERROR TOL TRL Alloy Combination TEMP OVER MAX TRANGE OPERATING PLANETS USED ]

% Units TYPE deg C deg C g - TBD Length Width Height Pascals -

% TBL %

TBD deg C deg C deg C deg C NO. - TBD

% TEMPSSENSOR_DB = [ 'B' 0.000 1700.000 0.000 0.000 0.00 0.00 0.00 0.00 0.000
0.5 0.0 0 'Pt-30%Rh' 'Pt-6%Rh' 0.000 13.280 'Voltage' [ ] ;
1.0 4.5 0 'W-5%Re' 'W-26%Re' 0.000 37.066 'Voltage' [ ] ;
| TempUnitLabelStr = {'Type' 'deg C' 'deg C' 'W' 'g' 'mm' 'mm' 'mm' 'Pascals' '%' 'deg C' 'TRL' 'Lead' 'Lead' 'mV' 'mV' 'N/A'} |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| if strcmp('CNITS', 'British') == 1 |
| % Properties [ SENSOR COLD LIMIT HOT LIMIT POWER REQ MASS REQ % Operating PLANETS USED DIMENSIONS (in) PRESSURE |
| % Units TYPE deg F deg F BTU/sec - TBD oz - TBD Length Width Height psi - TBD |
| % deg F LEVEL + Lead - Lead deg F deg F Type N/A |
| 0.5 | 0.0 | 0 | 'Pt-30%Rh' 'Pt-30%Rh' | 32.000 | 3092.000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 1.0 | 40.1 | 0 | 'W-5%Re' 'W-25%Re' | 32.000 | 4208.000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 1.0 | 40.1 | 0 | 'W-5%Re' 'W-25%Re' | 32.000 | 4208.000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 0.000 | 0.5 | 35.06 | 0 | 'Ni-Cr' 'Cu-Ni' | 32.000 | 4208.000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 0.000 | 1.0 | 40.1 | 0 | 'W' 'W-25%Re' | 32.000 | 4208.000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 0.000 | 0.75 | 35.96 | 0 | 'Fe' 'Cu-Ni' | 32.000 | 4208.000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 0.000 | 0.75 | 35.96 | 0 | 'Ni-Cr' 'Ni-Al' | 32.000 | 4208.000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 0.000 | 0.75 | 35.96 | 0 | 'Ni-Cr-Si' 'Ni-Al-Si-Mg' | 32.000 | 4208.000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 0.000 | 0.25 | 34.7 | 0 | 'Pt-13%Rh' 'Pt' | 32.000 | 4208.000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 0.000 | 0.25 | 34.7 | 0 | 'Pt-10%Rh' 'Pt' | 32.000 | 4208.000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
% Comments On the Bare Wire Environment - Use of the Thermocouple Material
TEMPSENSOR_DB{1,19} = 'Oxidizing or Inert. Do Not Insert in Metal Tubes. Beware of Contamination. High Temp. Common Use in Glass Industry.';
TEMPSENSOR_DB{4,19} = 'Oxidizing or Inert. Limited Use in Vacuum or Reducing. Highest EMF Change Per Degree.';
TEMPSENSOR_DB{6,19} = 'Reducing, Vacuum, Inert. Limited Use in Oxidizing at High Temperatures. Not Recommended for Low Temperatures.';
TEMPSENSOR_DB{7,19} = 'Clean Oxidizing and Inert. Limited Use in Vacuum or Reducing. Wide Temperature Range, Most Popular Calibration.';
TEMPSENSOR_DB{8,19} = 'Alternative to Type K. More Stable at High Temps.';
TEMPSENSOR_DB{9,19} = 'Oxidizing or Inert. Do Not Insert in Metal Tubes. Beware of Contamination. High Temperature.';
TEMPSENSOR_DB{10,19} = 'Oxidizing or Inert. Do Not Insert in Metal Tubes. Beware of Contamination. High Temperature.';
TEMPSENSOR_DB{11,19} = 'Mild Oxidizing, Reducing Vacuum or Inert. Good Where Moisture Is Present. Low Temperature & Cryogenic Applications.';
TEMPSENSOR_DB{12,19} = 'Resistance type Thermal Sensor.';
TEMPSENSOR_DB{13,19} = 'Resistance type Thermal Sensor. High Reliability, Linear Output, Miniature Size, Wide Temperature Range, Recommended for Space Vehicles.';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Thermocouple Wire Insulator Data Base Created Based on un Unit System
%% 2 - D Matrix TCWIns_DB(Ins_Type, Property)
if strcmp(UNITS,'SI') == 1
  Properties [ INSULATION COLD LIMIT HOT LIMIT ]
  Units [ TYPE deg C deg C ]
  TCWIns_DB1 = {
    'PP' -40.000 105.000 ;
    'FF' -200.000 200.000 ;
    'TT' -267.000 260.000 ;
    'KK' -267.000 316.000 ;
    'TG' -73.000 260.000 ;
    'GG' -73.000 482.000 ;
    'HH' -73.000 871.000 ;
    'XR' -73.000 871.000 ;
    'XC' -73.000 1204.000 ;
    'XS' -73.000 1038.000 ;
    'TFE' -267.000 260.000 ;
};
WInsUnitLabelStr = { 'Type' 'deg C' 'deg C' }; end

if strcmpi(UNITS, 'British') == 1

% Properties [ INSULATION COLD LIMIT HOT LIMIT ]
% Units [ TYPE deg F deg F ]
TCWIns_DB1 = { 'PP' -40.000 221.000 ; 'FF' -338.000 392.000 ; 'TT' -450.000 500.000 ; 'KK' -450.000 600.000 ; 'TG' -100.000 500.000 ; 'GG' -100.000 900.000 ; 'HH' -100.000 1300.000 ; 'XR' -100.000 1600.000 ; 'XC' -100.000 2200.000 ; 'XS' -100.000 1900.000 ; 'TFE' -450.000 500.000 }

WInsUnitLabelStr = { 'Type' 'deg F' 'deg F' }; end

% Adds Additional Environmental Insulation Data to the Thermal Limit Properties
% [ Abrasion Flexibility Water Material Construction ]
% Resistance To: Acid Base Flame
TCWIns_DB2 = { 'Good' 'Excellent' 'Good' 'Fair' 'Good' 'Good' 'Good' 'PVC' 'PVC' ; 'PP' 'Type' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'PFP Teflon' 'PFP Teflon' ; 'PP' 'Type' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'PFA Teflon' 'PFA Teflon' ; 'PP' 'Type' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Kapton' 'Kapton' ; 'PP' 'Type' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Glass Braid' 'PFA Teflon' ; 'PP' 'Type' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Glass Braid' 'PFA Teflon' ; 'PP' 'Type' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Excellent' 'Poor' ; 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' 'Poor' ' Poor
if strcmpi(TEMPSENSOR_DB{t,17}, TEMP_SENS_TYPE) == 1
    for s = 1:size(TEMPSENSOR_DB,2)
        MATRIX{tmp,s} = TEMPSENSOR_DB{t,s};
    end
    tmp = tmp + 1;
end

if strcmpi(TEMP_SENS_TYPE, 'VOLTAGE') == 1
    c = 1;
    % Build Up Database of Possible Thermocouple Sensors Based on Temperature Range
    for t = 1:size(MATRIX,1)
        if (PLTEMP >= MATRIX{t,2}) && (PLTEMP <= MATRIX{t,3})
            for a = 1:size(MATRIX,2)
                MATRIX2{c,a} = MATRIX{t,a};
            end
            c = c+1;
        end
    end
    if exist('MATRIX2') == 0
        fprintf('

    *** WARNING ***
    NO VIABLE TEMPERATURE SENSOR EXISTS IN DATABASE!');
        fprintf('
    Other methods to determine Planetary Temperature must be used.
    
end
    for c = 1:size(MATRIX2,1)
        RESULT = findstr(MATRIX2{c,19}, 'Most Popular');
        if size(RESULT) == 0
            SENS_NUM = c;
            break;
        end
    end
end
if strcmpi(PLNAME, 'PLUTO') == 1
    for c = 1:size(MATRIX2,1)
        RESULT = findstr(MATRIX2{c,19}, 'Cryogenic');
        if size(RESULT) == 0
            SENS_NUM = c;
            break;
        end
    end
end
\[ k = 1; \]
\% Build Up Database of Possible Thermocouple Insulation Material Based on Temperature Range
for insul = 1:size(TCWIns_DB,1)
    if (PLTEMP >= TCWIns_DB{insul,2}) && (PLTEMP <= TCWIns_DB{insul,3})
        for a = 1:size(TCWIns_DB,2)
            INS_MATRIX{k,a} = TCWIns_DB{insul,a};
        end
        k = k+1;
    end
end

SUM_ARRAY = [];
total = 0;
% Determine Optimal Insulation Material
for z = 1:size(INS_MATRIX,1)
    for mat = 4:11
        total = 0;
        switch (INS_MATRIX{z,mat})
            case 'Poor'
                total = total + Poor;
            case 'Fair'
                total = total + Fair;
            case 'Good'
                total = total + Good;
            case 'Excellent'
                total = total + Excellent;
        end
    end
    SUM_ARRAY(z) = total;
end

[ SUM, Max_INS ] = max (SUM_ARRAY);

% Save Temperature Sensor Properties to Data File
save ('TEMP_SENSOR_Data.mat');
save ('TEMP_SENSOR_Final.mat', 'MATRIX2', 'TempUnitLabelStr');

% *** PRINT SUMMARY Output Section for 'Print' = 'Y'
if strcmpi(Print,'Y') == 1
    fprintf('

*** TEMPERATURE SENSOR SUMMARY RESULTS ***

*** INPUTS ***
Planetary Surface Temperature: %7.2f %s', PLTEMP, TempUnit);
fprintf('
 Unit System: %s', UNITS);

fprintf('

% Temperature Sensor Properties
fprintf('
** Sensor Properties **
fprintf('
 Thermal Sensor Type: %s', MATRIX2{SENS_NUM,1});
fprintf('
 Thermal Sensor Low Limit: %9.3f %s', MATRIX2{SENS_NUM,2}, TempUnitLabelStr{2});
fprintf('
 Thermal Sensor High Limit: %9.3f %s', MATRIX2{SENS_NUM,3}, TempUnitLabelStr{3});
fprintf('
 Mass Requirements: %9.3f %s', MATRIX2{SENS_NUM,4}, TempUnitLabelStr{4});
fprintf('
 Power Requirements: %9.3f %s', MATRIX2{SENS_NUM,5}, TempUnitLabelStr{5});
fprintf('
 Dimensions - Length: %9.3f %s', MATRIX2{SENS_NUM,6}, TempUnitLabelStr{6});
fprintf('
 Width: %9.3f %s', MATRIX2{SENS_NUM,7}, TempUnitLabelStr{7});
fprintf('
 Height: %9.3f %s', MATRIX2{SENS_NUM,8}, TempUnitLabelStr{8});
fprintf('
 Operating Pressure: %9.3f %s', MATRIX2{SENS_NUM,9}, TempUnitLabelStr{9});
fprintf('
 Error Tolerance: %9.3f %s', MATRIX2{SENS_NUM,10}, TempUnitLabelStr{10});
fprintf('
 TRL Level: %9d %s', MATRIX2{SENS_NUM,11}, TempUnitLabelStr{11});

fprintf('

** Thermocouple Construction Materials **
fprintf('
 Positive Terminal: %10s %6s', MATRIX2{SENS_NUM,13}, TempUnitLabelStr{13});
fprintf('
 Negative Terminal: %10s %6s', MATRIX2{SENS_NUM,14}, TempUnitLabelStr{14});

fprintf('

EMF Voltage spread over full Temperature Range: %7.3f %s', MATRIX2{SENS_NUM,15}, TempUnitLabelStr{15});

fprintf('

** Sensor Comments: **
fprintf('
 %s', MATRIX2{SENS_NUM,19});

fprintf('

** INSULATION MATERIAL PROPERTIES **
fprintf('
 Insulation Type: %s', INS_MATRIX{Max_INS,1});
fprintf('
 Insulation Lower Limit: %9.3f %s', INS_MATRIX{Max_INS,2}, WInsUnitLabelStr{2});
fprintf('
 Insulation Upper Limit: %9.3f %s', INS_MATRIX{Max_INS,3}, WInsUnitLabelStr{3});

fprintf('

** Environmental Properties **
fprintf('
 Abrasion Resistance: %s', INS_MATRIX{Max_INS,4});
fprintf('
 Flexibility: %s', INS_MATRIX{Max_INS,5});
fprintf('
 Water Submersion: %s', INS_MATRIX{Max_INS,6});
fprintf('
 Resistance to Solvents: %s', INS_MATRIX{Max_INS,7});
fprintf('
 Resistance to Acids: %s', INS_MATRIX{Max_INS,8});
fprintf('
 Resistance to Bases: %s', INS_MATRIX{Max_INS,9});
fprintf('
 Resistance to Flame: %s', INS_MATRIX{Max_INS,10});
fprintf('
 Resistance to Humidity: %s', INS_MATRIX{Max_INS,11});

fprintf('

 Overall Material Construction: %s', INS_MATRIX{Max_INS,12});
fprintf('
 Conductor Material Construction: %s', INS_MATRIX{Max_INS,13});

end


%%% Writes a text file summary of the Planetary Properties
Fout = fopen('TEMPERATURE_SENSOR_Summary.txt','w+');
fprintf(Fout, '\n \n *** TEMPERATURE SENSOR SUMMARY RESULTS ***');
fprintf(Fout, '\n Planetary Surface Temperature: %7.2f \%s', PLTEMP, TempUnit);
fclose(Fout);
if strcmpi(TEMP_SENS_TYPE,'RESISTANCE') == 1
if size(MATRIX,1) > 1
  c = 1;
  % Build Up Database of Possible Thermocouple Sensors Based on Temperature Range
  for t = 1:size(MATRIX,1)
    if (PLTEMP >= MATRIX(t,2)) && (PLTEMP <= MATRIX(t,3))
      for a = 1:size(MATRIX,2)
        MATRIX2{c,a} = MATRIX{t,a};
      end
      c = c+1;
    end
  end
  if exist('MATRIX2') == 0
    fprintf('

 *** WARNING ***

 NO VIABLE TEMPERATURE SENSOR EXISTS IN DATABASE!

 Other methods to determine Planetary Temperature must be used.
');
    MATRIX2 = MATRIX;
  end
if size(MATRIX2,1) > 1
  % Sort Sensors By Smallest Error Tolerance Percent
  for err = 1:size(MATRIX2,1)
    TEMP_ERR(err) = MATRIX2{err,10};
  end;
  [ MIN_ERR_SCL, ER_INDEX ] = min(TEMP_ERR);
  % Build Sensor Matrix Based on Min ERROR Tolerances
  k = 1;
  for c = 1:length(TEMP_ERR)
    if TEMP_ERR(c) == MIN_ERR_SCL
      for j = 1:size(MATRIX2,2)
        MATRIX3{k,j} = MATRIX2{c,j};
      end
      k = k + 1;
    end
  end
  if exist('MATRIX3') == 0
    MATRIX3 = MATRIX2;
  end
  clear MATRIX2;
  MATRIX2 = MATRIX3;
end
if strcmpi(PLNAME,'PLUTO') == 1
for c = 1:size(MATRIX2,1)
    RESULT = findstr(MATRIX2{c,19},'Cryogenic');
    if size(RESULT) == 0
        else
            SENS_NUM = c;
            break;
        end
    end
end

k = 1;
% Build Up Database of Possible Thermocouple Insulation Material Based on Temperature Range
for insul = 1:size(TCWIns_DB,1)
    if (PLTEMP >= TCWIns_DB{insul,2}) && (PLTEMP <= TCWIns_DB{insul,3})
        for a = 1:size(TCWIns_DB,2)
            INS_MATRIx{k,a} = TCWIns_DB{insul,a};
        end
        k = k+1;
    end
end

SUM_ARRAY = [];
total = 0;
% Determine Optimal Insulation Material
for z = 1:size(INS_MATRIX,1)
    for mat = 4:11
        total = 0;
        switch (INS_MATRIX{z,mat})
            case 'Poor'
                total = total + Poor;
            case 'Fair'
                total = total + Fair;
            case 'Good'
                total = total + Good;
            case 'Excellent'
                total = total + Excellent;
        end
    end
    SUM_ARRAY(z) = total;
end

[ SUM, Max_INS ] = max(SUM_ARRAY);
if exist('MATRIX2') == 0
    MATRIX2 = MATRIX;
end
% Save Temperature Sensor Properties to Data File
save('TEMP_SENSOR_Data.mat');
save('TEMP_SENSOR_Final.mat','MATRIX2','TempUnitLabelStr');

% *** PRINT SUMMARY Output Section for 'Print' = 'Y'
if strcmpi(Print,'Y') == 1

fprintf('

 *** TEMPERATURE SENSOR SUMMARY

*** INPUTS ***
Planetary Surface Temperature: %7.2f %s' PLTEMP, TempUnit);
Unit System: %s', UNITS);

% Temperature Sensor Properties
Sensor Properties
Thermal Sensor Type: %s', MATRIX2{1,1});
Thermal Sensor Low Limit: %9.3f %s', MATRIX2{1,2}, TempUnitLabelStr{2});
Thermal Sensor High Limit: %9.3f %s', MATRIX2{1,3}, TempUnitLabelStr{4});
Power Requirements: %9.3f %s', MATRIX2{1,4}, TempUnitLabelStr{6});
Mass Requirements: %9.3f %s', MATRIX2{1,5}, TempUnitLabelStr{8});
Dimensions - Length: %9.3f %s', MATRIX2{1,6}, TempUnitLabelStr{10});
Width: %9.3f %s', MATRIX2{1,7}, TempUnitLabelStr{12});
Height: %9.3f %s', MATRIX2{1,8}, TempUnitLabelStr{14});
Operating Pressure: %9.3f %s', MATRIX2{1,9}, TempUnitLabelStr{16});
Error Tolerance: %9.3f %s', MATRIX2{1,10}, TempUnitLabelStr{18});

% TRL Level: %d %s', MATRIX2{1,12}, TempUnitLabelStr{10});
Thermocouple Construction Materials:
Positive Terminal: %10s %6s', MATRIX2{1,13}, TempUnitLabelStr{13});
Negative Terminal: %10s %6s', MATRIX2{1,14}, TempUnitLabelStr{15});

EMF Voltage spread over full Temperature Range: %7.3f %s', MATRIX2{1,15},TempUnitLabelStr{17});

Sensor Comments:

end

% Writes a text file summary of the Planetary Properties
Fout = fopen('TEMPERATURE_SENSOR_Summary.txt','w+');
fprintf(Fout, '

 *** TEMPERATURE SENSOR SUMMARY RESULTS ***

*** INPUTS ***

');
fprintf(Fout, ' Planetary Surface Temperature: %7.2f %s', PLTEMP, TempUnit);
fprintf(Fout, ' Unit System: %s', UNITS);
fprintf(Fout, '**************************************************************
% Temperature Sensor Properties
fprintf(Fout, ' Sensor Properties **');
fprintf(Fout, ' Thermal Sensor Type: %s', MATRIX2{1,1});
fprintf(Fout, ' Thermal Sensor Limit: %9.3f %s', MATRIX2{1,2}, TempUnitLabelStr{2} );
fprintf(Fout, ' Thermal Sensor High Limit: %9.3f %s', MATRIX2{1,3}, TempUnitLabelStr{3} );
fprintf(Fout, ' Power Requirements: %9.3f %s', MATRIX2{1,4}, TempUnitLabelStr{4} );
fprintf(Fout, ' Mass Requirements: %9.3f %s', MATRIX2{1,5}, TempUnitLabelStr{5} );
fprintf(Fout, ' Dimensions - Length: %9.3f %s', MATRIX2{1,6}, TempUnitLabelStr{6} );
fprintf(Fout, ' Dimensions - Width: %9.3f %s', MATRIX2{1,7}, TempUnitLabelStr{7} );
fprintf(Fout, ' Dimensions - Height: %9.3f %s', MATRIX2{1,8}, TempUnitLabelStr{8} );
fprintf(Fout, ' Operating Pressure: %9.3f %s', MATRIX2{1,9}, TempUnitLabelStr{9} );
fprintf(Fout, ' Error Tolerance: %9.3f %s', MATRIX2{1,10}, TempUnitLabelStr{10} );
fprintf(Fout, ' TRL Level: %9d %s', MATRIX2{1,11}, TempUnitLabelStr{11} );
fprintf(Fout, ' BMF Voltage spread over full Temperature Range: %7.3f %s', MATRIX2{1,15}, MATRIX2{1,16}, TempUnitLabelStr{15} );
fprintf(Fout, ' Sensor Comments:');
fprintf(Fout, ' %s', MATRIX2{1,19});
fclose(Fout);

D17. Wind Velocity Sensors
% function WIND_VELOCITY_SENSORS(WV_TYPE,UNITS,Print)
% Wind Velocity Sensor Database
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
% This file contains Wind Velocity Sensor values for the ISSPO program.
% Data here is loaded into the main program to determine the proper sensor
% to use for Missions based on the Planetary Atmosphere composition.
% Values are assigned to a variable name and saved to a MatLAB *.mat' file
% and loaded when calling the ISSPO program.
% Some of the values are based on the Unit system chosen and are loaded
% in separate sections.
% References
% http://www.omega.com/guides/PortableVelocityMeters.html
% http://www.calright.com/_coreModules/common/categoryDetail.aspx?entityType=6&categoryID=39
% http://www.symmttm.com/products_precision_frequency_references.asp
% http://www.symmsda.com/applications/space.asp

function WIND_VELOCITY_SENSORS(WV_TYPE, UNITS, Print)

% Load Constants values into Local Program
load('Constants_DB.mat');
load('Planet_Data.mat');

if strcmpi(WV_TYPE, 'ANEMOMETER') == 1
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% Wind Velocity Sensor Data Base Created Based on un Unit System
    %% 2 - D Matrix WIND_VELOCITY_SENSOR_DB(WV_Type, Property)
    if strcmpi(UNITS, 'SI') == 1
        % Properties
        INST TEMP RANGE [ SENSOR WIND VELOCITY ACCURACY DISP VOL FLOW RATE PROBE TEMP RANGE
        % Units
        Low deg C High Number TYPE [ Low m/s High % FS N/A Low deg C High
        0.0 50.0 2 'AA' 'HH-30A' 0.2032 39.624 0.50 'NO' -20.0 100.0
        0.0 50.0 2 'AA' 'HHF300A' 0.3048 34.544 0.50 'NO' -20.0 100.0
        0.0 50.0 6 'AAA' 'HHF42' 0.0000 20.015 1.00 'NO' 0.0 50.0
        0.0 100.0 0 'NiCad w/AC' 'HHF750' 0.2032 39.624 0.50 'YES' -20.0 100.0
        0.0 50.0 2 'AA' 'HHF753' 0.2032 39.624 0.50 'NO' -20.0 100.0
        0.0 50.0 2 'AA' 'HHF91' 0.6150 24.892 2.00 'YES' -10.0 50.0
        -10.0 50.0 1 '9 Volt' 'HHF92A' 0.4600 35.000 3.00 'YES' -10.0 50.0
        -10.0 50.0 1 '9 Volt' 'REED YK-80A' 0.8000 12.000 2.00 'NO' 0.0 80.0
        0.0 80.0 1 '9 Volt' 'REED 8903' 0.4000 30.000 0 'NO' -10.0 50.0
        -10.0 50.0 1 '9 Volt' 'FT 702' 0.0000 70.000 4.00 'NO' -40.0 85.0
        -40.0 85.0 0 'N/A' ];

        % Properties
        TYPE CURRENT POWER ] [ WEIGHT DIMENSIONS HANDHELD - mm DIMENSIONS PROBE - mm VOLTAGE
        % Units
        N/A mA mw | kg Length Width Height Length Width Height V
        N/A
<table>
<thead>
<tr>
<th>WIND VELOCITY SENSOR</th>
<th>DB1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
% Properties | WEIGHT | DIMENSIONS HANDHELD - in | DIMENSIONS PROBE - in | VOLTAGE | TYPE
mA | BTUs/hr | [ ] | [ ] | [ ] | [ ] | [ ]
0 | 0.500 | 7.10 | 3.00 | 0.80 | 4.875 | 2.750 | 1.635 | 0 | 'N/A'
14.0 | 0.342 |

WIND_VELOCITY_SENSOR_DB2 = { 'FT 702', 0.00, 13779.5, 4.00, 'NO', -40.0, 185.0
}

V/185.0 0 'N/A'

END
WIND_VELOCITY_SENSOR_DB{10,1} = 'Combination Velocity/Temperature Probe. Min/Max/Ave Velocity Measurements. 2 hour Continuous Ave Velocity. Rs232 Signal Output Format. Integrated Probe/Electronics configuration.';
WIND_VELOCITY_SENSOR_DB{11,1} = 'High accuracy wind speed and direction sensing. Wind Direction Measurement. Compact, unobtrusive solid-state design with no moving parts';

WIND_VELOCITY_SENSOR_DB = cat(2, WIND_VELOCITY_SENSOR_DB1, WIND_VELOCITY_SENSOR_DB2, WIND_VELOCITY_SENSOR_DB3);
WVelocityUnitLabelStr = cat(2, WVelocityUnitLabelStrl, WVelocityUnitLabelStr2);

\% Determine Optimal Sensor if Planetary Atmosphere Exists
ATM_FLG = 0;
\%
Search Planetary Atmosphere for Components
if strcmpi(PlanAtm_Prcp{l.1},'N/A') == 0
    ATM_FLG = 1;
end
if ATM_FLG == 0
    if strcmpi(PlanAtm_MEProp{1,1},'N/A') == 0
        ATM_FLG = 1;
    end
end
\%
Determine the WIND VELOCITY Sensor required if Atmosphere is Defined
if ATM_FLG == 1
    \% Find the Max Operational Range
    for w = 1:size(WIND_VELOCITY_SENSOR_DB,1)
        W_DELTA(w) = WIND_VELOCITY_SENSOR_DB{w,3} - WIND_VELOCITY_SENSOR_DB{w, 2};
    end
    \% Find the Maximum Wind Velocity Range of the Sensor
    MAX_DELTA = max(W_DELTA);
    \% For Multiple Sensors with the Same Range Reduce Sensor Matrix to Those
    \% with the Greatest Range
    pos = 1;
    for r = 1:length(W_DELTA)
        if W_DELTA(r) == MAX_DELTA
            for j = 1:size(WIND_VELOCITY_SENSOR_DB,2)
                W_MATRIX{pos,j}' = WIND_VELOCITY_SENSOR_DB{r, j} ;
            end
            pos = pos + 1;
        end
    end
    \% Continue Selection Process if More than one Sensor have Same Max Range
    if size(W_MATRIX,1) > 1
        \% Down Select Remaining Sensors Based on Mass and Dimensions
        % Create Mass / Volume Scaling Factor Optimization
        for mv = 1:size(W_MATRIX,1)
            % Probe Dimensions...
WV_SCALE_FAC(mv) = W_MATRIX{mv, 12} * (W_MATRIX{mv, 13} + W_MATRIX{mv, 14} * W_MATRIX{mv, 15}) * W_MATRIX{mv, 17} * W_MATRIX{mv, 18});

if (W_MATRIX{mv, 12} == 0) || (W_MATRIX{mv, 13} == 0) || (W_MATRIX{mv, 14} == 0) || (W_MATRIX{mv, 15} == 0)
WV_SCALE_FAC(mv) = 9.99E99;
end
end

[ WV_MIN_SCL, WV_INDEX ] = min (WV_SCALE_FAC);

% Build Sensor Matrix Based on Scale Factors
k = 1;
for c = 1:length(WV_SCALE_FAC)
if WV_SCALE_FAC(c) == WV_MIN_SCL
for j = 1:size(W_MATRIX, 2)
W_MATRIX2[k, j] = W_MATRIX{c, j};
end
k = k + 1;
end
end

% Copies W_MATRIX to W_MATRIX2 if all Sensors Eliminated
if exist('W_MATRIX2') == 0
W_MATRIX2 = W_MATRIX;
end

if size(W_MATRIX2, 1) > 1
% Sort Sensors by Accuracy
% Find the Min Wind Velocity Range of the Sensor
[ MIN_ACC, ACC_INDEX ] = min(W_MATRIX2);
W_MATRIX3 = W_MATRIX2{ACC_INDEX, :};
end

% Copies W_MATRIX2 to W_MATRIX if all Sensors Eliminated
if exist('W_MATRIX3') == 0
W_MATRIX3 = W_MATRIX2;
end
end
if exist('W_MATRIX3') == 0
W_MATRIX3 = W_MATRIX;
end

% Closes IF Loop If Atmosphere is Defined
% Determines if a sensor is determined from database
if exist('W_MATRIX3') == 0
for wv = 1:size(WIND_VELOCITY_SENSOR_DB, 2)
W_MATRIX3{1, w} = 0.0;
end
% Save Wind Velocity Sensor Properties to Data File
save('WIND_VELOCITY_SENSOR_Data.mat');
save('WIND_VELOCITY_SENSOR_Final.mat','W_MATRIX3','WVelocityUnitLabelStr', 'WV_TYPE');

% ******************************************
% *** PRINT SUMMARY Output Section for 'Print' = 'Y'
% ******************************************

if strcmpi(Print, 'Y') == 1
    fprintf('

 *** WIND VELOCITY SENSOR SUMMARY

 *** INPUTS ***
 Wind Velocity Sensor Type: %s', WV_TYPE);
 Unit System: %s', UNITS);

 *** RESULTS

 Wind Velocity Sensor Type: %s', WV_TYPE);
 Velocity Sensing Range - Low: %9.4f %s', W_MATRIX3{1,2}, WVelocityUnitLabelStr{2} );
 Velocity Sensing Range - High: %9.4f %s', W_MATRIX3{1,3}, WVelocityUnitLabelStr{3} );
 Velocity Accuracy: %9.4f %s', W_MATRIX3{1,4}, WVelocityUnitLabelStr{4} );
 Volumetric Flow Rate: %9.4f %s', W_MATRIX3{1,5} );

 *** Environmental Properties

 Probe Temperature Range - Low: %9.3f %s', W_MATRIX3{1,6}, WVelocityUnitLabelStr{6} );
 Probe Temperature Range - High: %9.3f %s', W_MATRIX3{1,7}, WVelocityUnitLabelStr{7} );
 Instrument Temperature Range - Low: %9.3f %s', W_MATRIX3{1,8}, WVelocityUnitLabelStr{8} );
 Instrument Temperature Range - High: %9.3f %s', W_MATRIX3{1,9}, WVelocityUnitLabelStr{9} );

 *** Power Requirements

 Number of Batteries: %9d %s', W_MATRIX3{1,10}, WVelocityUnitLabelStr{10} );
 Battery Type: %s %s', W_MATRIX3{1,11}, WVelocityUnitLabelStr{11} );
 Voltage Input: %9.3f %s', W_MATRIX3{1,19}, WVelocityUnitLabelStr{19} );
 Voltage Type: %s %s', W_MATRIX3{1,20}, WVelocityUnitLabelStr{20} );
 Current Input: %9.3f %s', W_MATRIX3{1,21}, WVelocityUnitLabelStr{21} );
 Input Power: %9.3f %s', W_MATRIX3{1,22}, WVelocityUnitLabelStr{22} );

 *** Physical Properties

 Sensor Mass: %9.3f %s', W_MATRIX3{1,12}, WVelocityUnitLabelStr{12} );
 Handheld Dimensions - Length: %9.3f %s', W_MATRIX3{1,13}, WVelocityUnitLabelStr{13} );
 Handheld Dimensions - Width: %9.3f %s', W_MATRIX3{1,14}, WVelocityUnitLabelStr{14} );
 Handheld Dimensions - Height: %9.3f %s', W_MATRIX3{1,15}, WVelocityUnitLabelStr{15} );
 Probe Dimensions - Length: %9.3f %s', W_MATRIX3{1,16}, WVelocityUnitLabelStr{16} );

% ******************************************
% *** PRINT SUMMARY Output Section for 'Print' = 'Y'
% ******************************************

if strcmpi(Print, 'Y') == 1
    fprintf('

 *** WIND VELOCITY SENSOR SUMMARY

 *** INPUTS ***
 Wind Velocity Sensor Type: %s', WV_TYPE);
 Unit System: %s', UNITS);

 *** RESULTS

 Wind Velocity Sensor Type: %s', WV_TYPE);
 Velocity Sensing Range - Low: %9.4f %s', W_MATRIX3{1,2}, WVelocityUnitLabelStr{2} );
 Velocity Sensing Range - High: %9.4f %s', W_MATRIX3{1,3}, WVelocityUnitLabelStr{3} );
 Velocity Accuracy: %9.4f %s', W_MATRIX3{1,4}, WVelocityUnitLabelStr{4} );
 Volumetric Flow Rate: %9.4f %s', W_MATRIX3{1,5} );

 *** Environmental Properties

 Probe Temperature Range - Low: %9.3f %s', W_MATRIX3{1,6}, WVelocityUnitLabelStr{6} );
 Probe Temperature Range - High: %9.3f %s', W_MATRIX3{1,7}, WVelocityUnitLabelStr{7} );
 Instrument Temperature Range - Low: %9.3f %s', W_MATRIX3{1,8}, WVelocityUnitLabelStr{8} );
 Instrument Temperature Range - High: %9.3f %s', W_MATRIX3{1,9}, WVelocityUnitLabelStr{9} );

 *** Power Requirements

 Number of Batteries: %9d %s', W_MATRIX3{1,10}, WVelocityUnitLabelStr{10} );
 Battery Type: %s %s', W_MATRIX3{1,11}, WVelocityUnitLabelStr{11} );
 Voltage Input: %9.3f %s', W_MATRIX3{1,19}, WVelocityUnitLabelStr{19} );
 Voltage Type: %s %s', W_MATRIX3{1,20}, WVelocityUnitLabelStr{20} );
 Current Input: %9.3f %s', W_MATRIX3{1,21}, WVelocityUnitLabelStr{21} );
 Input Power: %9.3f %s', W_MATRIX3{1,22}, WVelocityUnitLabelStr{22} );

 *** Physical Properties

 Sensor Mass: %9.3f %s', W_MATRIX3{1,12}, WVelocityUnitLabelStr{12} );
 Handheld Dimensions - Length: %9.3f %s', W_MATRIX3{1,13}, WVelocityUnitLabelStr{13} );
 Handheld Dimensions - Width: %9.3f %s', W_MATRIX3{1,14}, WVelocityUnitLabelStr{14} );
 Handheld Dimensions - Height: %9.3f %s', W_MATRIX3{1,15}, WVelocityUnitLabelStr{15} );
 Probe Dimensions - Length: %9.3f %s', W_MATRIX3{1,16}, WVelocityUnitLabelStr{16} );

% ******************************************
% *** PRINT SUMMARY Output Section for 'Print' = 'Y'
% ******************************************

if strcmpi(Print, 'Y') == 1
    fprintf('

 *** WIND VELOCITY SENSOR SUMMARY

 *** INPUTS ***
 Wind Velocity Sensor Type: %s', WV_TYPE);
 Unit System: %s', UNITS);

 *** RESULTS

 Wind Velocity Sensor Type: %s', WV_TYPE);
 Velocity Sensing Range - Low: %9.4f %s', W_MATRIX3{1,2}, WVelocityUnitLabelStr{2} );
 Velocity Sensing Range - High: %9.4f %s', W_MATRIX3{1,3}, WVelocityUnitLabelStr{3} );
 Velocity Accuracy: %9.4f %s', W_MATRIX3{1,4}, WVelocityUnitLabelStr{4} );
 Volumetric Flow Rate: %9.4f %s', W_MATRIX3{1,5} );

 *** Environmental Properties

 Probe Temperature Range - Low: %9.3f %s', W_MATRIX3{1,6}, WVelocityUnitLabelStr{6} );
 Probe Temperature Range - High: %9.3f %s', W_MATRIX3{1,7}, WVelocityUnitLabelStr{7} );
 Instrument Temperature Range - Low: %9.3f %s', W_MATRIX3{1,8}, WVelocityUnitLabelStr{8} );
 Instrument Temperature Range - High: %9.3f %s', W_MATRIX3{1,9}, WVelocityUnitLabelStr{9} );

 *** Power Requirements

 Number of Batteries: %9d %s', W_MATRIX3{1,10}, WVelocityUnitLabelStr{10} );
 Battery Type: %s %s', W_MATRIX3{1,11}, WVelocityUnitLabelStr{11} );
 Voltage Input: %9.3f %s', W_MATRIX3{1,19}, WVelocityUnitLabelStr{19} );
 Voltage Type: %s %s', W_MATRIX3{1,20}, WVelocityUnitLabelStr{20} );
 Current Input: %9.3f %s', W_MATRIX3{1,21}, WVelocityUnitLabelStr{21} );
 Input Power: %9.3f %s', W_MATRIX3{1,22}, WVelocityUnitLabelStr{22} );

 *** Physical Properties

 Sensor Mass: %9.3f %s', W_MATRIX3{1,12}, WVelocityUnitLabelStr{12} );
 Handheld Dimensions - Length: %9.3f %s', W_MATRIX3{1,13}, WVelocityUnitLabelStr{13} );
 Handheld Dimensions - Width: %9.3f %s', W_MATRIX3{1,14}, WVelocityUnitLabelStr{14} );
 Handheld Dimensions - Height: %9.3f %s', W_MATRIX3{1,15}, WVelocityUnitLabelStr{15} );
 Probe Dimensions - Length: %9.3f %s', W_MATRIX3{1,16}, WVelocityUnitLabelStr{16} );
fprintf('\n
Sensor Comments:');
fprintf('\n %s', W_MATRIX3(1, 1) );
end

%**************************************************************************%****  Write a text file summary of the Planetary Properties
Fout = fopen('WIND VELOCITY SENSOR_Summary.txt','w+');
fprintf(Fout, '\n *** WIND VELOCITY SENSOR SUMMARY RESULTS ***\n');
fprintf(Fout, '\n *** INPUTS ***\n Wind Velocity Sensor Type: %s', W_MATRIX3(1, 2) );
fprintf(Fout, ' Unit System: %s', W_MATRIX3(1, 3) );
fprintf(Fout, '\n Wind Velocity Sensor Type: %s', W_MATRIX3(1, 1) );
fprintf(Fout, ' Unit System: %s', W_MATRIX3(1, 18) );
fprintf(Fout, '\n Velocity Sensing Range - Low: $9.4f $s', W_MATRIX3(1, 2) );
fprintf(Fout, ' Velocity Sensing Range - High: $9.4f $s', W_MATRIX3(1, 3) );
fprintf(Fout, ' Velocity Accuracy: $9.4f $s', W_MATRIX3(1, 4) );
fprintf(Fout, ' Volumetric Flow Rate: $9.4f $s', W_MATRIX3(1, 5) );
% Wind Velocity Sensor Properties
fprintf(Fout, '\n *** SENSOR PROPERTIES ***\n Wind Velocity Sensor Type: %s', W_MATRIX3(1, 1) );
fprintf(Fout, ' Unit System: %s', W_MATRIX3(1, 18) );
fprintf(Fout, '\n Velocity Sensing Range - Low: $9.4f $s', W_MATRIX3(1, 2) );
fprintf(Fout, ' Velocity Sensing Range - High: $9.4f $s', W_MATRIX3(1, 3) );
fprintf(Fout, ' Velocity Accuracy: $9.4f $s', W_MATRIX3(1, 4) );
fprintf(Fout, ' Volumetric Flow Rate: $9.4f $s', W_MATRIX3(1, 5) );
% Environmental Properties
fprintf(Fout, '\n *** ENVIRONMENTAL PROPERTIES ***\n Probe Temperature Range - Low: $9.3f $f', W_MATRIX3(1, 6) );
fprintf(Fout, ' Probe Temperature Range - High: $9.3f $f', W_MATRIX3(1, 7) );
fprintf(Fout, ' Instrument Temperature Range - Low: $9.3f $f', W_MATRIX3(1, 8) );
fprintf(Fout, ' Instrument Temperature Range - High: $9.3f $f', W_MATRIX3(1, 9) );
% Power Requirements
fprintf(Fout, '\n *** POWER REQUIREMENTS ***\n Number of Batteries: $9d $s', W_MATRIX3(1, 10) );
fprintf(Fout, ' Battery Type: %s', W_MATRIX3(1, 11) );
fprintf(Fout, ' Voltage Input: $9.3f $v', W_MATRIX3(1, 19) );
fprintf(Fout, ' Voltage Type: %s', W_MATRIX3(1, 20) );
fprintf(Fout, ' Current Input: $9.3f $a', W_MATRIX3(1, 21) );
fprintf(Fout, ' Input Power: $9.3f $w', W_MATRIX3(1, 22) );
% Physical Properties
fprintf(Fout, '\n *** PHYSICAL PROPERTIES ***\n Sensor Mass: $9.3f $s', W_MATRIX3(1, 12) );
fprintf(Fout, ' Handheld Dimensions - Length: $9.3f $s', W_MATRIX3(1, 13) );
fprintf(Fout, ' Width: $9.3f $s', W_MATRIX3(1, 14) );
fprintf(Fout, ' Height: $9.3f $s', W_MATRIX3(1, 15) );
fprintf(Fout, ' Probe Dimensions - Length: $9.3f $s', W_MATRIX3(1, 16) );
fprintf(Fout, ' Width: $9.3f $s', W_MATRIX3(1, 17) );
fprintf(Fout, ' Height: $9.3f $s', W_MATRIX3(1, 18) );
fprintf(Fout, '\n *** SENSOR COMMENTS ***
');
fprintf(Fout, ' *** WIND VELOCITY SENSOR SUMMARY RESULTS ***

*** INPUTS ***
 Wind Velocity Sensor Type: %s', W_MATRIX3(1, 2) );
unit System: %s', W_MATRIX3(1, 3) );
fprintf(Fout, ' Velocity Sensing Range - Low: $9.4f $s', W_MATRIX3(1, 2) );
fprintf(Fout, ' Velocity Sensing Range - High: $9.4f $s', W_MATRIX3(1, 3) );
fprintf(Fout, ' Velocity Accuracy: $9.4f $s', W_MATRIX3(1, 4) );
fprintf(Fout, ' Volumetric Flow Rate: $9.4f $s', W_MATRIX3(1, 5) );
% Wind Velocity Sensor Properties
fprintf(Fout, ' Velocity Sensing Range - Low: $9.4f $s', W_MATRIX3(1, 2) );
fprintf(Fout, ' Velocity Sensing Range - High: $9.4f $s', W_MATRIX3(1, 3) );
fprintf(Fout, ' Velocity Accuracy: $9.4f $s', W_MATRIX3(1, 4) );
fprintf(Fout, ' Volumetric Flow Rate: $9.4f $s', W_MATRIX3(1, 5) );
% Environmental Properties
fprintf(Fout, ' Probe Temperature Range - Low: $9.3f $f', W_MATRIX3(1, 6) );
fprintf(Fout, ' Probe Temperature Range - High: $9.3f $f', W_MATRIX3(1, 7) );
fprintf(Fout, ' Instrument Temperature Range - Low: $9.3f $f', W_MATRIX3(1, 8) );
fprintf(Fout, ' Instrument Temperature Range - High: $9.3f $f', W_MATRIX3(1, 9) );
% Power Requirements
fprintf(Fout, ' Number of Batteries: $9d $s', W_MATRIX3(1, 10) );
fprintf(Fout, ' Battery Type: %s', W_MATRIX3(1, 11) );
fprintf(Fout, ' Voltage Input: $9.3f $v', W_MATRIX3(1, 19) );
fprintf(Fout, ' Voltage Type: %s', W_MATRIX3(1, 20) );
fprintf(Fout, ' Current Input: $9.3f $a', W_MATRIX3(1, 21) );
fprintf(Fout, ' Input Power: $9.3f $w', W_MATRIX3(1, 22) );
% Physical Properties
fprintf(Fout, ' Sensor Mass: $9.3f $s', W_MATRIX3(1, 12) );
fprintf(Fout, ' Handheld Dimensions - Length: $9.3f $s', W_MATRIX3(1, 13) );
fprintf(Fout, ' Width: $9.3f $s', W_MATRIX3(1, 14) );
fprintf(Fout, ' Height: $9.3f $s', W_MATRIX3(1, 15) );
fprintf(Fout, ' Probe Dimensions - Length: $9.3f $s', W_MATRIX3(1, 16) );
fprintf(Fout, ' Width: $9.3f $s', W_MATRIX3(1, 17) );
fprintf(Fout, ' Height: $9.3f $s', W_MATRIX3(1, 18) );
fprintf(Fout, ' *** SENSOR COMMENTS ***
');
```matlab
fprintf(Fout, ' 
 %s', W_MATRIX3{l,23 });
fclose(Fout);
end

if strcmpi(WV_TYPE, 'DOPPLER') == 1

% Begins Selection Section For Doppler Type Wind Velocity Sensors

%%% Wind Velocity Sensor Data Base Created Based on a Unit System

%%% 2-D Matrix WIND_VELOCITY_SENSOR_DB (WV_Type, Property)

% Electrical Specifications

<table>
<thead>
<tr>
<th>AGING DATA</th>
<th>ACCURACY</th>
<th>STABILITY</th>
<th>WARM UP TIME</th>
<th>VOLTAGE TYPE</th>
<th>VOLTAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Month</td>
<td>Year</td>
<td>@ 25 C</td>
<td>@ 100 sec</td>
<td>Min</td>
<td>N/A</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>---------</td>
<td>-----------</td>
<td>----</td>
<td>-----</td>
</tr>
<tr>
<td>WIND_VELOCITY_SENSOR_DB1 = { '8130A'</td>
<td>10.0</td>
<td>7.0</td>
<td>1.50</td>
<td>-30.0</td>
<td>-80.0</td>
</tr>
<tr>
<td>5.0E-11</td>
<td>1.0E-09</td>
<td>1.0E-11</td>
<td>3.0E-12</td>
<td>14.0</td>
<td>'DC'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3E-09</td>
<td>1.0E-06</td>
<td>5.0E-12</td>
<td>0.0</td>
<td>7.0</td>
<td>2.00</td>
</tr>
</tbody>
</table>

% Properties

<table>
<thead>
<tr>
<th>PRESSURE SENS</th>
<th>WARM-UP MAX POWER</th>
<th>STEADY STATE POWER</th>
<th>INPUT POWER</th>
<th>QUIESCENT</th>
<th>WEIGHT</th>
<th>DIMENSIONS - cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hz/mbar</td>
<td>W @ 28 V</td>
<td>W @ 28 V</td>
<td>W @ 28 V</td>
<td>25 deg C</td>
<td>Kg</td>
<td>Length</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>------------</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td>1.0E-13</td>
<td>35.0</td>
<td>22.0</td>
<td>12.0</td>
<td>0.900</td>
<td>7.26</td>
<td>4.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>12.0</td>
<td>4.0</td>
<td>3.0</td>
<td>0.159</td>
<td>7.62</td>
<td>4.24</td>
</tr>
</tbody>
</table>

% Properties

<table>
<thead>
<tr>
<th>PRESSURE SENS</th>
<th>WARM-UP MAX POWER</th>
<th>STEADY STATE POWER</th>
<th>INPUT POWER</th>
<th>QUIESCENT</th>
<th>WEIGHT</th>
<th>DIMENSIONS - in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hz/psi</td>
<td>BTUs/hr @ 28 V</td>
<td>BTUs/hr @ 28 V</td>
<td>BTUs/hr @ 28 V</td>
<td>77 deg F</td>
<td>lbm</td>
<td>Length</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>------------</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

end

if strcmpi(UNITS, 'SI') == 1

% Properties

<table>
<thead>
<tr>
<th>PRESSURE SENS</th>
<th>WARM-UP MAX POWER</th>
<th>STEADY STATE POWER</th>
<th>INPUT POWER</th>
<th>QUIESCENT</th>
<th>WEIGHT</th>
<th>DIMENSIONS - cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hz/mbar</td>
<td>W @ 28 V</td>
<td>W @ 28 V</td>
<td>W @ 28 V</td>
<td>25 deg C</td>
<td>Kg</td>
<td>Length</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>------------</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td>1.0E-13</td>
<td>35.0</td>
<td>22.0</td>
<td>12.0</td>
<td>0.900</td>
<td>7.26</td>
<td>4.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>12.0</td>
<td>4.0</td>
<td>3.0</td>
<td>0.159</td>
<td>7.62</td>
<td>4.24</td>
</tr>
</tbody>
</table>

end

if strcmpi(UNITS, 'British') == 1

% Properties

<table>
<thead>
<tr>
<th>PRESSURE SENS</th>
<th>WARM-UP MAX POWER</th>
<th>STEADY STATE POWER</th>
<th>INPUT POWER</th>
<th>QUIESCENT</th>
<th>WEIGHT</th>
<th>DIMENSIONS - in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hz/psi</td>
<td>BTUs/hr @ 28 V</td>
<td>BTUs/hr @ 28 V</td>
<td>BTUs/hr @ 28 V</td>
<td>77 deg F</td>
<td>lbm</td>
<td>Length</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>------------</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

end
```
WIND VELOCITY SENSOR DB2 = { -40.0 154.4 -79.6 185.0 3.0E-10 5.0E-11 6.89E-12 19.351 75.134 40.982 1.984 4.04 2.92 2.87 20 0.00 40.982 13.661 0.245 0.351 3.00 1.67 1.60};

WVelocityUnitLabelStr2 = { 'deg F' 'deg F' 'deg F' 'deg F' 'Hz' 'Hz' 'Hz/psi' 'BTUs/hr' 'BTUs/hr' 'BTUs/hr'};

WIND VELOCITY SENSOR DB3{1,1} = { 'Modern Militarized Design' ;
'5 and 10 MHz Sinewave Outputs' ;
'Ruggedized High Performance Rb Physics Package' ;
'Meets many Mil-Spec Standards' ;
'Data for Single Unit' ;
'Two Units required for Doppler Tracking' ;
'Space Qualified Hardware Production capability' ;};

WIND VELOCITY SENSOR DB3{2,1} = { 'Modern Militarized Ovenized SC Design' ;
'10 MHz Sinewave Output' ;
'RS-232 Digital Control and Monitoring' ;
'Low g Sensitivity Military OCXO' ;
'Meets many Mil-Spec Standards' ;
'Data for Single Unit' ;
'Two Units required for Doppler Tracking' ;
'Space Qualified Hardware Production capability' ;};

% Combine Data Arrays into a Single Data Matrix
WIND_VELOCITY_SENSOR_DB = cat (2 , WIND_VELOCITY_SENSOR_DB1, WIND_VELOCITY_SENSOR_DB2, WIND_VELOCITY_SENSOR_DB3) ;
WVelocityUnitLabelStr = cat(2,WVelocityUnitLabelStr1, WVelocityUnitLabelStr2);

% Determine Optimal Sensor if Planetary Atmosphere Exists
ATM_FLG = 0;
% Search Planetary Atmosphere for Components
if strcmpi(PlanAtm_Prop{1,1},'N/A') == 0
    ATM_FLG = 1;
end
if ATM_FLG == 0
    if strcmpi(PlanAtm_MEPrcp{1,1},'N/A') == 0
        ATM_FLG = 1;
    end
end

% Determines the WIND VELOCITY Sensor required if Atmosphere is Defined
if ATM_FLG == 1
    % Find the Max Operational Output Range
    for w = 1:size(WIND_VELOCITYSENSOR_DB,1)
        W_DELTA(w) = WIND_VELOCITYSENSOR_DB(w,3);
    end
    % Find the Maximum Output Range of the Sensor
MAX_DELTA = max(W_DELTA);

% For Multiple Sensors with the Same Range Reduce Sensor Matrix to Those
% with the Greatest Range
pos = 1;
for r = 1:length(W_DELTA)
    if W_DELTA(r) == MAX_DELTA
        for j = 1:size(WIND_VELOCITY_SENSOR_DB,2)
            W_MATRIX{pos,j} = WIND_VELOCITY_SENSOR_DB{r,j};
        end
        pos = pos + 1;
    end
end

% Continue Selection Process if More than one Sensor have Same Max Range
if size(W_MATRIX,1) > 1

% Find the Min Spectral Purity Value
for w = 1:size(W_MATRIX,1)
    W_DELTA2(w) = abs(W_MATRIX{w,5});
end

% Find the Min Spectral Purity Value Range of the Sensor
MIN_DELTA2 = min(W_DELTA2);

% For Multiple Sensors with the Same Range Reduce Sensor Matrix to Those
% with the Greatest Range
pos = 1;
for r = 1:length(W_DELTA2)
    if W_DELTA2(r) == MIN_DELTA2
        for j = 1:size(W_MATRIX,2)
            W_MATRIX2{pos,j} = W_MATRIX{r,j};
        end
        pos = pos + 1;
    end
end

% Copies W_MATRIX to W_MATRIX2 if all Sensors Eliminated
if exist('W_MATRIX2') == 0
    W_MATRIX2 = W_MATRIX;
end
if size(W_MATRIX2,1) > 1
    % Down Select Remaining Sensors Based on Mass and Dimensions
    % Create Sensitivity Scaling Factor Optimization
    for sv = 1:size(W_MATRIX2,1)
        W.scale_fac(sv) = W_MATRIX2{sv,18} * W_MATRIX2{sv,18} * W_MATRIX2{sv,20};
        if (W_MATRIX2{sv,18} == 0) || (W_MATRIX2{sv,19} == 0) || (W_MATRIX2{sv,20} == 0)
            W.scale_fac(sv) = 9.99E99;
        end
    end
end
end

[ WV_MIN_SCL, WV_INDEX ] = min(WV_SCALE_FAC);

% Build Sensor Matrix Based on Scale Factors
k = 1;
for c = 1:length(WV_SCALE_FAC)
    if WV_SCALE_FAC(c) == WV_MIN_SCL
        for j = 1:size(W_MATRIX2,2)
            W_MATRIX3{k, j} = W_MATRIX2{c, j};
        end
        k = k + 1;
    end
end

% Copies W_MATRIX2 to W_MATRIX3 if all Sensors Eliminated
if exist('W_MATRIX3') == 0
    W_MATRIX3 = W_MATRIX2;
end

if exist('W_MATRIX3') == 0
    W_MATRIX3 = W_MATRIX;
end

% Closes IF Loop If Atmosphere is Defined

% Determines if a sensor is determined from database
if exist('W_MATRIX3') == 0
    for w = 1:size(WIND_VELOCITY_SENSOR_DB,2)
        W_MATRIX3{1, w} = 0.0;
    end
    W_MATRIX3{1,1} = 'N/A';
    W_MATRIX3{1,12} = 'N/A';
    W_MATRIX3{1,29} = 'N/A';
end

% Save Wind Velocity Sensor Properties to Data File
save('WIND_VELOCITY_SENSOR_Data.mat');
save('WIND_VELOCITY_SENSOR_Final.mat','W_MATRIX3','WVelocityUnitLabelStr','WV_TYPE');

% *** PRINT SUMMARY Output Section for 'Print' = 'Y'
if strcmpi(Print, 'Y') == 1
    fprintf(' *** WIND VELOCITY SENSOR SUMMARY RESULTS ***

*** INPUTS ***

% Print Table From Database

% Print Summary Output

Wind Velocity Sensor Type: %s', WV_TYPE);
fprintf('Unit System: %s', UNITS);

% Wind Velocity Sensor Properties
fprintf('Wind Velocity Sensor Type: %s', W_MATRIX3{1,1});
fprintf('Sensor Output Frequency: %9.3f %s', W_MATRIX3{1,2}, WVelocityUnitLabelStr{2} );
fprintf('Sensor Output Level: %9.3f %s', W_MATRIX3{1,3}, WVelocityUnitLabelStr{3} );
fprintf('Output Tolerance: %9.3f %s', W_MATRIX3{1,4}, WVelocityUnitLabelStr{4} );
fprintf('Spectral Purity - Harmonics: %9.3f %s', W_MATRIX3{1,5}, WVelocityUnitLabelStr{5} );
fprintf('Spectral Purity - Non-Harmonics: %9.3f %s', W_MATRIX3{1,6}, WVelocityUnitLabelStr{6} );
fprintf('Aging Data - Month: %9.2E %s', W_MATRIX3{1,7}, WVelocityUnitLabelStr{7} );
fprintf('Aging Data - 10 years: %9.2E %s', W_MATRIX3{1,8}, WVelocityUnitLabelStr{8} );
fprintf('Sensor Accuracy: %9.2E %s', W_MATRIX3{1,9}, WVelocityUnitLabelStr{9} );
fprintf('Sensor Stability: %9.2E %s', W_MATRIX3{1,10}, WVelocityUnitLabelStr{10} );
fprintf('Sensor Warm Up Time: %9.3f %s', W_MATRIX3{1,11}, WVelocityUnitLabelStr{11} );
fprintf('Sensor Lifetime: %9.1f %s', W_MATRIX3{1,12}, WVelocityUnitLabelStr{12} );

% Environmental Properties
fprintf('Operating Temperature Range - Low: %9.3f %s', W_MATRIX3{1,14}, WVelocityUnitLabelStr{14} );
fprintf('Operating Temperature Range - High: %9.3f %s', W_MATRIX3{1,15}, WVelocityUnitLabelStr{15} );
fprintf('Storage Temperature Range - Low: %9.3f %s', W_MATRIX3{1,16}, WVelocityUnitLabelStr{16} );
fprintf('Storage Temperature Range - High: %9.3f %s', W_MATRIX3{1,17}, WVelocityUnitLabelStr{17} );
fprintf('Temperature Sensitivity - Op Range: %9.2E %s', W_MATRIX3{1,18}, WVelocityUnitLabelStr{18} );
fprintf('Orientation Sensitivity: %9.2E %s', W_MATRIX3{1,19}, WVelocityUnitLabelStr{19} );
fprintf('Pressure Sensitivity: %9.2E %s', W_MATRIX3{1,20}, WVelocityUnitLabelStr{20} );

% Power Requirements
fprintf('Voltage Type: %s %s', W_MATRIX3{1,12}, WVelocityUnitLabelStr{12} );
for g = 1:length(W_MATRIX3{1,13})
    fprintf(' %6.3f', W_MATRIX3{1,13}(g) );
end
fprintf(' %s', WVelocityUnitLabelStr{13} );

% Physical Properties
fprintf('Sensor Mass: %9.3f %s', W_MATRIX3{1,24}, WVelocityUnitLabelStr{24} );
fprintf('Sensor Dimensions - Length: %9.3f %s', W_MATRIX3{1,25}, WVelocityUnitLabelStr{25} );
fprintf('Sensor Dimensions - Width: %9.3f %s', W_MATRIX3{1,26}, WVelocityUnitLabelStr{26} );
fprintf('Sensor Dimensions - Height: %9.3f %s', W_MATRIX3{1,27}, WVelocityUnitLabelStr{27} );

% Sensor Comments:
for r = 1:length(W_MATRIX3{1,28})
    fprintf('
 %s', W_MATRIX3{1,28}(r) );
end
end
%******************************************************************************
%***     Writes a text file summary of the Planetary Properties
%
Fout = fopen('WIND_VELOCITY_SENSOR_Summary.txt','w+');

fprintf(Fout, '

*** WIND VELOCITY SENSOR SUMMARY RESULTS ***
');
fprintf(Fout, ' 
 *** INPUTS
');
fprintf(Fout, ' 
 Wind Velocity Sensor Type: %s', WV_TYPE);
fprintf(Fout, ' 
 Unit System: %s', UNITS);

fprintf(Fout, '

** Sensor Properties **
');
fprintf(Fout, ' 
 Wind Velocity Sensor Type: %s', WV_TYPE);
fprintf(Fout, ' 
 Unit System: %s', UNITS);

fprintf(Fout, '

% Wind Velocity Sensor Properties
');
fprintf(Fout, ' 
 Sensor Output Frequency: %9.3f %s', W_MATRIX3{1,12}, W_velocityUnitLabelStr{12} );
fprintf(Fout, ' 
 Sensor Output Level: %9.3f %s', W_MATRIX3{1,13}, W_velocityUnitLabelStr{13} );
fprintf(Fout, ' 
 Spectral Purity - Harmonics: %9.3f %s', W_MATRIX3{1,15}, W_velocityUnitLabelStr{15} );
fprintf(Fout, ' 
 Spectral Purity - Non-Harmonics: %9.3f %s', W_MATRIX3{1,16}, W_velocityUnitLabelStr{16} );
fprintf(Fout, ' 
 Aging Data - Month: %9.2E %s', W_MATRIX3{1,17}, W_velocityUnitLabelStr{17} );
fprintf(Fout, ' 
 - 10 years: %9.2E %s', W_MATRIX3{1,18}, W_velocityUnitLabelStr{18} );
fprintf(Fout, ' 
 Sensor Accuracy: %9.2E %s', W_MATRIX3{1,19}, W_velocityUnitLabelStr{19} );
fprintf(Fout, ' 
 Sensor Stability: %9.2E %s', W_MATRIX3{1,20}, W_velocityUnitLabelStr{20} );
fprintf(Fout, ' 
 Sensor Warm Up Time: %9.2E %s', W_MATRIX3{1,21}, W_velocityUnitLabelStr{21} );
fprintf(Fout, ' 
 Sensor Lifetime: %9.2E %s', W_MATRIX3{1,22}, W_velocityUnitLabelStr{22} );

% Environmental Properties
');
fprintf(Fout, ' 
 Operating Temperature Range - Low: %9.3f %s', W_MATRIX3{1,14}, W_velocityUnitLabelStr{14} );
fprintf(Fout, ' 
 - High: %9.3f %s', W_MATRIX3{1,15}, W_velocityUnitLabelStr{15} );
fprintf(Fout, ' 
 Storage Temperature Range - Low: %9.3f %s', W_MATRIX3{1,16}, W_velocityUnitLabelStr{16} );
fprintf(Fout, ' 
 - High: %9.3f %s', W_MATRIX3{1,17}, W_velocityUnitLabelStr{17} );
fprintf(Fout, ' 
 Temperature Sensitivity - Op Range: %9.2E %s', W_MATRIX3{1,18}, W_velocityUnitLabelStr{18} );
fprintf(Fout, ' 
 Orientation Sensitivity: %9.2E %s', W_MATRIX3{1,19}, W_velocityUnitLabelStr{19} );
fprintf(Fout, ' 
 Pressure Sensitivity: %9.2E %s', W_MATRIX3{1,20}, W_velocityUnitLabelStr{20} );

fprintf(Fout, ' 
 Power Requirements
');
fprintf(Fout, ' 
 Voltage Type: %s %s', W_MATRIX3{1,11}, W_velocityUnitLabelStr{11} );
fprintf(Fout, ' 
 System Voltage Levels: %s %s', W_MATRIX3{1,12}, W_velocityUnitLabelStr{12} );

for g = 1:length(W_MATRIX3{1,13})
    fprintf(Fout, ' %6.3f', W_MATRIX3{1,13}(g) );
end

fprintf(Fout, ' %s', W_velocityUnitLabelStr{13} );
fprintf(Fout, ' %s', W_MATRIX3{1,21}, W_velocityUnitLabelStr{21} );
fprintf(Fout, ' %s', W_MATRIX3{1,22}, W_velocityUnitLabelStr{22} );
fprintf(Fout, ' %s', W_MATRIX3{1,23}, W_velocityUnitLabelStr{23} );

fprintf(Fout, ' 
 Physical Properties
');
fprintf(Fout, ' %s', W_MATRIX3{1,24}, W_velocityUnitLabelStr{24} );
fprintf(Fout, ' %s', W_MATRIX3{1,25}, W_velocityUnitLabelStr{25} );
fprintf(Fout, ' %s', W_MATRIX3{1,26}, W_velocityUnitLabelStr{26} );
fprintf(Fout, ' %s', W_MATRIX3{1,27}, W_velocityUnitLabelStr{27} );
function REFRACTION_SENSORS(UNITS,Print)

% Refraction Sensor Database

% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
%
% This file contains Refraction Sensor values for the ISSPO program.
% Data here is loaded into the main program to determine the proper sensor
% to use for Missions based on the sensor requirements.
% Values are assigned to a variable name and saved to a MatLAB '.mat' file
% and loaded when calling the ISSPO program.
%
% Some of the values are based on the Unit system chosen and are loaded
% in separate sections.
%
% References
% http://www.sensata.com/products/sensors/spreeta-r.htm

function REFRACTION_SENSORS(UNITS,Print)

% Load Constants values into Local Program
load('Constants_DB.mat');
load('Planet_Data.mat');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Refraction Sensor Data Base Created Based on un Unit System
%% 2 - D Matrix REFRACTION_DB(N_Type, Property)
if strcmpi(UNITS,'SI') == 1

% Properties
[ Sensor REFRACION_INDEX RANGE RESOLUTION DRIFT ACCURACY WEIGHT VOLTAGE TYPE
CURRENT POWER DIMENSIONS - cm ]
<table>
<thead>
<tr>
<th>Units</th>
<th>mA</th>
<th>mW</th>
<th>Length</th>
<th>Width</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>REFRACTION_DB1 =</td>
<td>{'TSPR2XXY-R'}</td>
<td>1.333</td>
<td>1.500</td>
<td>5.0E-06</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>0</td>
<td>3.00</td>
<td>0.70</td>
<td>1.50</td>
</tr>
<tr>
<td></td>
<td>'RFI-NP'</td>
<td>1.000</td>
<td>1.700</td>
<td>1.0E-04</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>22.80</td>
<td>0.97</td>
<td>0.97</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>'REF-Huy'</td>
<td>1.250</td>
<td>1.450</td>
<td>1.0E-03</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>% Properties</th>
<th>OPERATING TEMPERATURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Units</td>
<td>Low deg C</td>
</tr>
<tr>
<td>REFRACTION_DB2 =</td>
<td>-10.0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>-40.0</td>
</tr>
</tbody>
</table>

RefUnitLabelStr1 = {'sensor' | 'N/A' | 'N/A' | 'RIU' | 'RIU' | 'g' | 'V' | 'Type' | 'mA' | 'mW' | 'cm' | 'cm' | 'cm' | 'cm' | ;

RefUnitLabelStr2 = {'deg C' | 'deg C' | ;

end

if strcmpi(UNITS,'BRITISH') == 1
% Properties | Sensor | REFRAC TION INDEX RANGE | RESOLUTION | DRIFT | ACCURACY | WEIGHT | VOLTAGE | TYPE |
|--------------|--------|-------------------------|------------|-------|----------|--------|---------|------|
% Units | mA | BTU's/hr | Length | Width | Height |
| REFRACTION_DB1 = | {'TSPR2XXY-R'} | 1.333 | 1.500 | 5.0E-06 | 0 | 1.00 | 0 | 5 | 'DC' |
| 100 | 0  | 1.708 | 1.170 | 0.265 | 0.599 |
| | 'RFI-NP' | 1.000 | 1.700 | 1.0E-04 | 0 | 0.071 | 0 | 0 | 'N/A' |
| 0 | 8.980 | 0.382 | 0.382 | ; |
| | 'REF-Huy' | 1.250 | 1.450 | 1.0E-03 | 0 | 1.00 | 0 | 15 | 'DC' |

6.66 | 0.034 | 3.940 | 3.940 | 4.570 | ;

% Properties | OPERATING TEMPERATURE |
% Units | Low deg C | High deg C |
| REFRACTION_DB2 = | -14.0 | 158.0 | ; |
| | 32.0 | 212.0 | ; |
| | -40.0 | 149.0 | ; |

RefUnitLabelStr1 = {'sensor' | 'N/A' | 'N/A' | 'RIU' | 'RIU' | 'g' | 'V' | 'Type' | 'mA' | 'BTU/hr' | 'in' | 'in' | 'in' | ;

RefUnitLabelStr2 = {'deg f' | 'deg f' | ;

end

% comments on sensors
REFRAC TION_DB3{ 1,1 } = {'Optical based sensing technology for real-time, liquid quality and/or concentration analysis' ;
'High performance and low cost measurement systems' ;
'High performance, Highly quantitative, Customized software available' ;
'Robust, Lightweight, Small size, Low power' ;

REFRAC TION_DB3{ 2,1 } = {'Ideal for refractive index measurement of fluids in industrial, chemical and food processing industry' ;
'Rugged stainless steel package' ;
'Intrinsically safe, immune to EMI/RFI' ;
In-situ measurement

REFRACTION_DB3{ 3,1} = {
'Unique Design Solution - Combination NMOS Linear Image Sensor and Sapphire Refraction Prism';
'Self-Scanning Photodiode Array';
'Linear critical-angle refractometer.';
'Flight Proven Hardware configuration - Huygens Lander SSP.'
};

% Combine Refraction Sensor Arrays
REFRACTION_DB = cat(2,REFRACTION_DB1,REFRACTION_DB2,REFRACTION_DB3);
RefUnitLabelStr = cat(2,RefUnitLabelStr1,RefUnitLabelStr2);

% Find the Max Operational Temp Range
for r = 1:size(REFRACTION_DB,1)
    REF_DELTA(r) = REFRACTION_DB{r,16} - REFRACTION_DB{r,15};
end
% Find the Maximum Refraction Operating Range of the Sensor
MAX_REF_DELTA = max(REF_DELTA);
% For Multiple Sensors with the Same Temp Range Reduce Sensor Matrix to Those
% with the Greatest Range
k = 1;
for c = 1:length(REF_DELTA)
    if REF_DELTA(c) == MAX_REF_DELTA
        for j = 1:size(REFRACTION_DB,2)
            REF_MATRIX{k,j} = REFRACTION_DB{c,j};
        end
        k = k + 1;
    end
end
if exist('REF_MATRIX') == 0
    REF_MATRIX = REFRACTICN_DB;
end
% Select Sensor Based on Refraction Range if More than One Sensor Exists In Database
if size(REF_MATRIX,1) > 1
    % Find The Max Operational Range
    for n = 1:size(REF_MATRIX,1)
        REF_DELTA_RNG(n) = REF_MATRIX{n,3} - REF_MATRIX{n,2};
    end
    % Find the Maximum Refraction Operating Range of the Sensor
    MAX_REF_DELTA_RNG = max(REF_DELTA_RNG);
    % For Multiple Sensors with the Same Range Reduce Sensor Matrix to Those
    % with the Greatest Range
    k = 1;
    for c = 1:length(REF_DELTA_RNG)
        if REF_DELTA_RNG(c) == MAX_REF_DELTA_RNG
            for j = 1:size(REF_MATRIX,2)
                REF_MATRIX2{k,j} = REF_MATRIX{c,j};
            end
        end
    end
k = k + 1;
end
end

if exist('REF_MATRIX2') == 0
    REF_MATRIX2 = REF_MATRIX;
end

% Select Sensor Based on Minimum Volume if More than One Sensor Exists In Database
if size(REF_MATRIX2,1) > 1
  % Create Volume Scaling Factor Optimization
  for v = 1:size(REF_MATRIX2,1)
    REF_SCALE_FAC(v) = REF_MATRIX2(v,9) * REF_MATRIX2(v,10) * REF_MATRIX2(v,11) * REF_MATRIX2(v,6);
    if (REF_MATRIX2(v,9) == 0) || (REF_MATRIX2(v,10) == 0) || (REF_MATRIX2(v,11) == 0) || (REF_MATRIX2(v,6) == 0)
        REF_SCALE_FAC(v) = 9.99E99;
    end
  end
  [ REF_MIN_SCL, INDEX ] = min(REF_SCALE_FAC);
end

% Build Sensor Matrix Based on Scale Factors
k = 1;
for c = 1:length(REF_SCALE_FAC)
    if REF_SCALE_FAC(c) == REF_MIN_SCL
        for j = 1:size(REF_MATRIX2,2)
            REF_MATRIX3{k,j} = REF_MATRIX2{c, j};
        end
        k = k + 1;
    end
end

% Copies REF_MATRIX2 to REF_MATRIX3 if all Sensors Eliminated
if exist('REF_MATRIX3') == 0
    REF_MATRIX3 = REF_MATRIX2;
end

% Save Refraction Sensor Properties to Data File
save('REFRACTION_SENSOR_Data.mat');
save('REFRACTION_SENSOR_Final.mat', 'REF_MATRIX3', 'RefUnitLabelStr');

% *** PRINT SUMMARY Output Section for 'Print' = 'Y'
if strcmpi(Print,'Y') == 1
    for c = 1:length(REF_MATRIX2)
        if REF_MATRIX2(c,6) == 0
            for j = 1:size(REF_MATRIX2,2)
                REF_MATRIX2{c,j} = 9.99E99;
            end
        end
    end
end
fprintf('

 *** REFRACTION SENSOR SUMMARY RESULTS ***');
fprintf('

 *** INPUTS ***');
fprintf('
 Unit System: %s', UNITS);

fprintf('

SENSOR SUMMARY RESULTS

% Refraction Sensor Properties
fprintf('

** Sensor Properties **
');
fprintf('
 Refraction Sensor Type:
');
fprintf('
 Sensing Refraction Range - Min:
');
fprintf('
 Middle:
');
fprintf('
 Sensing Resolution:
');
fprintf('
 Sensing Drift Rate:
');
fprintf('
 Sensor Accuracy:

% Power Requirements
fprintf('

 Power Requirements
');
fprintf('
 Input Voltage:
');
fprintf('
 Voltage Type:
');
fprintf('
 Input Current:
');
fprintf('
 Input Power:

% Physical Properties
fprintf('

 Physical Properties
');
fprintf('
 Sensor Mass:
');
fprintf('
 Sensor Dimensions - Length:
');
fprintf('
 Width:
');
fprintf('
 Height:

% Environmental Properties
fprintf('

 Environmental Properties
');
fprintf('
 Temperature Range - Low:
');
fprintf('
 High:

% Sensor Comments:
for r = 1:length(REF_MATRIX3{1,17})
 fprintf('
 %s', REF_MATRIX3{1,17}{r});
end

% Writes a text file summary of the Planetary Properties
Fout = fopen('REFRACTION_SENSOR_Summary.txt','w+');
fprintf(Fout, '

 *** REFRACTION SENSOR SUMMARY RESULTS ***');
fprintf(Fout, '

 *** INPUTS ***');
fprintf(Fout, '
 Unit System: %s', UNITS);

fprintf(Fout, '

SENSOR SUMMARY RESULTS

% Refraction Sensor Properties
fprintf(Fout, '

** Sensor Properties **
');
fprintf(Fout, '
 Refraction Sensor Type:
');
fprintf(Fout, '
 Sensing Refraction Range - Min:
');
fprintf(Fout, '
 Middle:
');
fprintf(Fout, '
 Sensing Resolution:
');
fprintf(Fout, '
 Sensing Drift Rate:
');
fprintf(Fout, '
 Sensor Accuracy:

% Power Requirements
fprintf(Fout, '

 Power Requirements
');
fprintf(Fout, '
 Input Voltage:
');
fprintf(Fout, '
 Voltage Type:
');
fprintf(Fout, '
 Input Current:
');
fprintf(Fout, '
 Input Power:

% Physical Properties
fprintf(Fout, '

 Physical Properties
');
fprintf(Fout, '
 Sensor Mass:
');
fprintf(Fout, '
 Sensor Dimensions - Length:
');
fprintf(Fout, '
 Width:
');
fprintf(Fout, '
 Height:

% Environmental Properties
fprintf(Fout, '

 Environmental Properties
');
fprintf(Fout, '
 Temperature Range - Low:
');
fprintf(Fout, '
 High:

% Sensor Comments:
for r = 1:length(REF_MATRIX3{1,17})
 fprintf(Fout, '%s', REF_MATRIX3{1,17}{r});
end

end

%****************************

fprintf('****************************

-----------------------------------

')
% Refraction Sensor Properties
fprintf(Fout, '\n\n** Sensor Properties **');
fprintf(Fout, '\n Refraction Sensor Type: %s', REF_MATRIX3{1,1});
fprintf(Fout, '\n Sensing Refraction Range - Min: %12.5f %s', REF_MATRIX3{1,2}, RefUnitLabelStr{2});
fprintf(Fout, '\n Sensing Range - Middle: %12.5f %s', REF_MATRIX3{1,3}, RefUnitLabelStr{3});
fprintf(Fout, '\n Sensing Range - Max: %12.5f %s', REF_MATRIX3{1,4}, RefUnitLabelStr{4});
fprintf(Fout, '\n Sensing Resolution: %12.5f %s', REF_MATRIX3{1,5}, RefUnitLabelStr{5});
fprintf(Fout, '\n Sensing Drift Rate: %12.5f %s', REF_MATRIX3{1,6}, RefUnitLabelStr{6});

fprintf(Fout, '\n\n Power Requirements');
fprintf(Fout, '\n Input Voltage: %12.3f %s', REF_MATRIX3{1,8}, RefUnitLabelStr{8});
fprintf(Fout, '\n Input Current: %12.3f %s', REF_MATRIX3{1,10}, RefUnitLabelStr{10});

fprintf(Fout, '\n\n Physical Properties');
fprintf(Fout, '\n Sensor Mass: %12.2f %s', REF_MATRIX3{1,7}, RefUnitLabelStr{7});
fprintf(Fout, '\n Sensor Dimensions - Length: %12.2f %s', REF_MATRIX3{1,12}, RefUnitLabelStr{12});
fprintf(Fout, '\n Sensor Dimensions - Width: %12.2f %s', REF_MATRIX3{1,13}, RefUnitLabelStr{13});
fprintf(Fout, '\n Sensor Dimensions - Height: %12.2f %s', REF_MATRIX3{1,14}, RefUnitLabelStr{14});

% Environmental Properties
fprintf(Fout, '\n\n Environmental Properties');
fprintf(Fout, '\n Temperature Range - Low: %12.2f %s', REF_MATRIX3{1,15}, RefUnitLabelStr{15});
fprintf(Fout, '\n Temperature Range - High: %12.2f %s', REF_MATRIX3{1,16}, RefUnitLabelStr{16});

fprintf(Fout, '\n\n Sensor Comments:');
for r = 1:length(REF_MATRIX3{1,17})
fprintf(Fout, ' %s', REF_MATRIX3{1,17}{r});
end
fclose(Fout);

D19. Digital Signal Processing
% function DIGITAL_SIGNAL_PROCESSING(DIG_SIG_TYPE,UNITS,Print)
% Digital Signal Processing Database
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
This file contains Digital Signal Processing values for the ISSPO program. Data here is loaded into the main program to determine the proper sensor to use for Missions based on the sensor requirements. Values are assigned to a variable name and saved to a MatLAB .mat file and loaded when calling the ISSPO program. Some of the values are based on the Unit system chosen and are loaded in separate sections.

References

function DIGITAL_SIGNAL_PROCESSING(DIG_SIG_TYPE,UNITS,Print)

% Load Constants values into Local Program
load('Constants_DB.mat')
load('Planet_Data.mat');

% References
% http://www.ortodoxism.ro/datasheets/analogdevices/ADSF-2100KG.pdf

if strcmpi(UNITS, 'SI') == 1

% Properties
% COMPUTATIONAL UNITS CYCLE TIME EXTERNAL INTERRUPTS
% Units
% Locs Number of Units ns Category Number
DIGSIGPROC_DB1 = {'ADSP-2100' 'LOW' 8.192 1 16 16 32 24

3 80.0 4
3 40.0 0
3 13.3 6

% Properties
% TYPE CURRENT
% Units

DSPUnitLabelStr1 = {'sensor' 'N/A' 'mA' 'K-Words' 'K-Words' 'ns';

DSPUnitLabelStr2 = {'deg' 'deg' 'deg' 'deg' 'mA' 'V' 'V' 'N/A' 'mA' 'mm' 'mm'};

end
if strcmpi(UNITS,'BRITISH') == 1
% Properties [ Sensor

<table>
<thead>
<tr>
<th>COMPUTATIONAL UNITS</th>
<th>CYCLE TIME</th>
<th>EXTERNAL INTERRUPTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locs</td>
<td>Number</td>
<td>Mhz</td>
</tr>
<tr>
<td>DIGSIGPROC_DB1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'ADSP-2100'</td>
<td>80.0</td>
<td>'LOW'</td>
</tr>
<tr>
<td>'ADSP-2115'</td>
<td>0</td>
<td>'MEDIUM'</td>
</tr>
<tr>
<td>'ADSP-2185'</td>
<td>13.3</td>
<td>'HIGH'</td>
</tr>
</tbody>
</table>

% Properties

<table>
<thead>
<tr>
<th>TYPE</th>
<th>CURRENT</th>
<th>CPU DIMENSIONS - in</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>mA</td>
<td>Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Width</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Height</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIGSIGPROC_DB2 =</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'DC'</td>
<td>100</td>
<td>1.332</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.359</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-67.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.110</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.344</td>
</tr>
<tr>
<td></td>
<td>38</td>
<td>-0.400</td>
</tr>
<tr>
<td></td>
<td>38</td>
<td>0.638</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.638</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.063</td>
</tr>
</tbody>
</table>

DSPUnitLabelStr1 = {'sensor' 'N/A' 'MHz' '#' 'K-Words' '#' 'K-Words' '#' '#' 'ns' '#' 'K-Words' '#' 'sensor' 'deg F' 'deg F' 'deg F' 'deg F' 'BTUs/hr' 'lbm' 'V' 'V' 'N/A' 'mA' 'in' 'in' 'in'};

DSPUnitLabelStr2 = {'deg F' 'deg F' 'deg F' 'deg F' 'BTUs/hr' 'lbm' 'V' 'V' 'N/A' 'mA' 'in' 'in' 'in'};

end

%comments on sensors

DSISGPROC_DB3{ 1,1} = {'Dual Purpose Program Memory for Both Instruction and Data Storage';
'Three Independent Computational Units: ALU, Multiplier/Accumulator and Barrel Shifter';
'Two Independent Data Address Shifters';
'Powerful Program Sequencer';
'Internal Instruction Cache';
'Provisions for Multiprecision Computation and Saturation Logic';
'Single-Cycle Instruction Execution';
'Multifunction Instructions';
'APPLICATIONS'
' - Optimized for DSP Algorithms including: Digital Filtering, Fast Fourier Transforms';
' - Image Processing';
' - Radar, Sonar';
' - Speech Processing';
' - Telecommunications'};

DSISGPROC_DB3{ 2,1} = {'16-Bit Fixed-Point DSP Microprocessors with On-Chip Memory';
'Enhanced Harvard Architecture for Three-Bus Performance: Instruction Bus & Dual Data Buses';
'Independent Computation Units: ALU, Multiplier/Accumulator, and Shifter';
'Single-Cycle Instruction Execution & Multifunction Instructions';
'On-Chip Program Memory RAM or ROM & Data Memory RAM';
'Separate On-Chip Buses for Program and Data Memory';
'Program Memory Stores Both Instructions and Data (Three-Bus Performance)';
'Dual Data Address Generators with Modulo and Bit-Reverse Addressing';

358
'Efficient Program Sequencing with Zero-Overhead Looping: Single-Cycle Loop Setup' ;
'Automatic Booting of On-Chip Program Memory from Byte-Wide External Memory (e.g., EPROM )' ;
'Double-Buffereed Serial Ports with Commanding Hardware, Automatic Data Buffering, and Multichannel
Operation' ;
'Three Edge- or Level-Sensitive Interrupts' ;
'Low Power IDLE Instruction' ;
'MIL-STD-883B Versions Available' ;
'Single-Cycle Instruction Execution' ;
'Single-Cycle Context Switch' ;
'3-Bus Architecture Allows Dual Operand Fetches in Every Instruction Cycle' ;
'Multipurpose Instructions' ;
'Power-Down Mode Featuring Low CMOS Standby Power Dissipation with 200 CLKIN Cycle Recovery from
Power-Down Condition' ;
'Low Power Dissipation in Idle Mode' ;
'80K Bytes of On-Chip RAM, Configured as 16K Words Program Memory RAM 16K Words Data Memory RAM' ;
'Dual-Purpose Program Memory for Both Instruction and Data Storage' ;
'Independent ALU, Multiplier/Accumulator, and Barrel Shifter Computational Units' ;
'Two Independent Data Address Generators' ;
'Powerful Program Sequencer Provides Zero Overhead Looping Conditional Instruction Execution' ;
'Programmable 16-Bit Interval Timer with Prescaler' ;

% Combine Refraction Sensor Arrays
DIGSIGPROC_DB = cat(2, DIGSIGPROC_DB1, DIGSIGPROC_DB2, DIGSIGPROC_DB3);
DSPUnitLabelStr = cat(2, DSPUnitLabelStr1, DSPUnitLabelStr2);

tmp = 1;
for DSP = 1:size(DIGSIGPROC_DB, 1)
  if strcmpi(DIGSIGPROC_DB{DSP, 2}, DIG_SIG_TYPE) == 1
    for s = 1:size(DIGSIGPROC_DB, 2)
      DSP_MATRIX{tmp, s} = DIGSIGPROC_DB{DSP, s};
    end
    tmp = tmp + 1;
  end
end

% Select Processor Based on Temp Range if More than One Sensor Exists In Database
if size(DSP_MATRIX, 1) > 1
  % Find The Max Operational Range
  for n = 1:size(DSP_MATRIX, 1)
    DSP_DELTA_RNG(n) = DSP_MATRIX(n, 13) - DSP_MATRIX(n, 12);
  end
  % Find the Maximum Refraction Operating Range of the Sensor
  MAX_DSP_DELTA_RNG = max(DSP_DELTA_RNG);
  % For Multiple Sensors with the Same Range Reduce Sensor Matrix to Those
  % with the Greatest Range
  k = 1;
  for c = 1:length(DSP_DELTA_RNG)
    if DSP_DELTA_RNG(c) == MAX_DSP_DELTA_RNG
      DSP_MATRIX{tmp, s} = DIGSIGPROC_DB{DSP, s};
      tmp = tmp + 1;
    end
  end
end
for j = 1:size(DSP_MATRIX,2)
    DSP_MATRIX2{k,j} = DSP_MATRIX{c,j};
end
k = k + 1;
end
end
if exist('DSP_MATRIX2') == 0
    DSP_MATRIX2 = DSP_MATRIX;
end
% Save Digital Signal Processing Sensor Properties to Data File
save('DIGITAL_SIGNAL_PROCESSING_Data.mat');
save('DIGITAL SIGNAL PROCESSING Final.mat','DSP MATRIX2' , 'DSPUnitLabelStr');

% Digital Signal Processing Properties
fprintf('\n** Sensor Properties **\n\n\nCPU Operating Speed: %s', DIG_SIG_TYPE);
fprintf('\nCPU Clock Speed: %12.3f %s', DSP_MATRIX2{1,2}, DSPUnitLabelStr{3});
fprintf('\nNumber of CPU Processing Cores: %12d %s', DSP_MATRIX2{1,3}, DSPUnitLabelStr{4});
fprintf('\nAmount of On-Board Data Memory: %12.1f %s', DSP_MATRIX2{1,4}, DSPUnitLabelStr{5});
fprintf('\nNumber of Data Bins: %12d %s', DSP_MATRIX2{1,5}, DSPUnitLabelStr{6});
fprintf('\nAmount of On-Board Program Memory: %12.1f %s', DSP_MATRIX2{1,6}, DSPUnitLabelStr{7});
fprintf('\nNumber of Memory Bins: %12d %s', DSP_MATRIX2{1,7}, DSPUnitLabelStr{8});
fprintf('\nNumber of Computational Units: %12d %s', DSP_MATRIX2{1,8}, DSPUnitLabelStr{9});
fprintf('\nCPU Cycle Time: %12.3f %s', DSP_MATRIX2{1,9}, DSPUnitLabelStr{10});
fprintf('\nNumber of External Interrupts: %12d %s', DSP_MATRIX2{1,10}, DSPUnitLabelStr{11});
fprintf('\nPower Requirements':
\nCPU Output Power: %12.3f %s', DSP_MATRIX2{1,11}, DSPUnitLabelStr{12});
fprintf('\nInput Voltage - Min: %12.3f %s', DSP_MATRIX2{1,12}, DSPUnitLabelStr{13});
fprintf('\nMax: %12.3f %s', DSP_MATRIX2{1,13}, DSPUnitLabelStr{14});
fprintf('\nVoltage Type: %s %s', DSP_MATRIX2{1,14}, DSPUnitLabelStr{15});
fprintf('\nInput Current: %12.3f %s', DSP_MATRIX2{1,15}, DSPUnitLabelStr{16});
Physical Properties

Processor Core Mass: %12.2f %s', DSP MATRIX2{l,17},DSPUnitLabelStr{l,17} );
Sensor Dimensions - Length: %12.3f %s', DSP MATRIX2{l,22},DSPUnitLabelStr{l,22} );
Width: %12.3f %s', DSP MATRIX2{l,23},DSPUnitLabelStr{l,23} );
Height: %12.3f %s', DSP MATRIX2{l,24},DSPUnitLabelStr{l,24} );

Environmental Properties

Operating Temperature Range - Low: %12.2f %s', DSP MATRIX2{l,12},DSPUnitLabelStr{l,12} );
High: %12.2f %s', DSP MATRIX2{l,13},DSPUnitLabelStr{l,13} );
Storage Temperature Range - Low: %12.2f %s', DSP MATRIX2{l,14},DSPUnitLabelStr{l,14} );
High: %12.2f %s', DSP MATRIX2{l,15},DSPUnitLabelStr{l,15} );

Sensor Comments:
for r = 1:length(DSP_MATRIX2{l,25})
    fprintf(Fout, '%s', DSP_MATRIX2{l,25}{r} );
end

% Environmental Properties

Environmental Properties:

Operating Temperature Range - Low: %12.2f %s', DSP MATRIX2{l,12},DSPUnitLabelStr{l,12} );
High: %12.2f %s', DSP MATRIX2{l,13},DSPUnitLabelStr{l,13} );
Storage Temperature Range - Low: %12.2f %s', DSP MATRIX2{l,14},DSPUnitLabelStr{l,14} );
High: %12.2f %s', DSP MATRIX2{l,15},DSPUnitLabelStr{l,15} );

Sensor Comments:
for r = 1:length(DSP_MATRIX2{l,25})
    fprintf(Fout, '%s', DSP_MATRIX2{l,25}{r} );
end

% Digital Signal Processing Properties

Digital Signal Processing Properties:

CPU Operating Speed: %s', DIG_SIG_TYPE); DSP_UNITSYSTEMStr{1},DSP_UNITSTR{1} );
CPU Clock Speed: %12.3f %s', DSP MATRIX2{l,3},DSPUnitLabelStr{l,3} );
Number of CPU Processing Cores: %12d %s', DSP MATRIX2{l,4},DSPUnitLabelStr{l,4} );
Amount of On-Board Data Memory: %12.1f %s', DSP MATRIX2{l,5},DSPUnitLabelStr{l,5} );
Number of Data Bins: %12d %s', DSP MATRIX2{l,6},DSPUnitLabelStr{l,6} );
Amount of On-Board Program Memory: %12.1f %s', DSP MATRIX2{l,7},DSPUnitLabelStr{l,7} );
Number of Memory Bins: %12d %s', DSP MATRIX2{l,8},DSPUnitLabelStr{l,8} );
Number of Computational Units: %12d %s', DSP MATRIX2{l,9},DSPUnitLabelStr{l,9} );
CPU Cycle Time: %12.3f %s', DSP MATRIX2{l,10},DSPUnitLabelStr{l,10} );
Number of External Interrupts: %12d %s', DSP MATRIX2{l,11},DSPUnitLabelStr{l,11} );
CPU Output Power: %12.3f %s', DSP MATRIX2{l,16},DSPUnitLabelStr{l,16} );
Input Voltage - Min: %12.3f %s', DSP MATRIX2{l,18},DSPUnitLabelStr{l,18} );

% Power Requirements

Power Requirements:

CPU Output Power: %12.3f %s', DSP MATRIX2{l,16},DSPUnitLabelStr{l,16} );
Input Voltage - Min: %12.3f %s', DSP MATRIX2{l,18},DSPUnitLabelStr{l,18} );
Max: $12.3f$ %s', DSP_MATRIX2{1,19},DSPUnitLabelStr{19} );
fprintf(Fout, 'Voltage Type: %s %s', DSP MATRIX2{1,20},DSPUnitLabelStr{20} );
fprintf(Fout, 'Input Current: $12.3f$ %s', DSP MATRIX2{1,21},DSPUnitLabelStr{21} );

fprintf(Fout, '

Physical Properties');
fprintf(Fout, '
Processor Core Mass: $12.2f$ %s', DSP_MATRIX2{1,17},DSPUnitLabelStr{17} );
fprintf(Fout, '
Sensor Dimensions - Length: $12.3f$ %s', DSP MATRIX2{1,22},DSPUnitLabelStr{22} );
fprintf(Fout, '
Width: $12.3f$ %s', DSP MATRIX2{1,23},DSPUnitLabelStr{23} );
fprintf(Fout, '
Height: $12.3f$ %s', DSP MATRIX2{1,24},DSPUnitLabelStr{24} );

% Environmental Properties
fprintf(Fout, '

Operating Temperature Range - Low: $12.2f$ %s', DSP MATRIX2{1,12},DSPUnitLabelStr{12} );
fprintf(Fout, '
High: $12.2f$ %s', DSP_MATRIX2{1,13},DSPUnitLabelStr{13} );
fprintf(Fout, '
Storage Temperature Range - Low: $12.2f$ %s', DSP MATRIX2{1,14},DSPUnitLabelStr{14} );
fprintf(Fout, '
High: $12.2f$ %s', DSP_MATRIX2{1,15},DSPUnitLabelStr{15} );

% Sensor Comments:
for r = 1:length(DSP_MATRIX2{1,25})
    fprintf(Fout, '
%s', DSP MATRIX2{1,25}{r} });
end
fclose(Fout);
D20. Error Program

% function ERROR_PRG(ERR_Code)
% ERROR Code Handling Program
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
% This file contains ERROR code handles for the ISSPO program.
% When an ERROR case flag is detected within the main ISSPO program
% this program is called with the appropriate ERR_Code. A message as
% to the error that occurred and the point in the program at which it
% occurred is printed to the screen and the program either waits for
% corrected input or simply exits. Along with the error message is a
% possible cause of the problem or a method to fix it. In most cases
% the error is the result of an unexpected input not matching a pre-
% defined match in the program. This is usually the result of an
% erroneous input to the program and must be remedied by the user.

function ERROR_PRG(ERR_Code)

% DISPLAY ERROR CODE NUMBER
fprintf('

 *** ERROR CODE: %3d * * *',ERR_Code) ;
% Display Error Code Statement
switch(ERR_Code)
% CASE X ERRORS - PROGRAMATIC ERRORS DATA FILES DO NOT EXIST
  case 1 % ERROR 1 - INPUT FILE DOES NOT EXIST
    disp('ERROR: Declared INPUT_FILE does not exist!');
    disp(' Verify INPUT_FILE exists before executing ISSPO!');
    disp('*** TERMINATING ISSPO PROGRAM ***');

  case 2 % ERROR 2 - INPUTDECK FILE DID NOT LOAD SUCCESSFULLY
    disp('ERROR: Loaded InputDeck File does not exist!');
    disp(' Program erred off in loading InputDeck file. ');
    disp(' Verify INPUT_FILE is successfully loaded into InputDeck. ');
    disp('*** TERMINATING ISSPO PROGRAM ***');
    cd ..\..\..\..

  case 3 % ERROR 3 - CONSTANTS DATABASE FILE FAILED TO BE SUCCESSFULLY CREATED
    disp('ERROR: Constants Database File does not exist!');
    disp(' Program erred off in creating Constants database file. ');
    disp(' Verify Constants program is correctly creating the database file. ');
    disp('*** TERMINATING ISSPO PROGRAM ***');
    cd ..\..\..\..

  case 4 % ERROR 4 - PLANETARY DATABASE FILE FAILED TO BE SUCCESSFULLY CREATED
    disp('ERROR: Planetary Database File does not exist!');
    disp(' Program erred off in creating Planetary database file. ');
    disp(' Verify Planetary Database program is correctly creating the database file. ');
    disp('*** TERMINATING ISSPO PROGRAM ***');
    cd ..\..\..\..

% CASE 5X ERRORS UNKNOWN VARIABLES LOADED INTO PROGRAM
  case 50 % CASE 50 ERROR UNIDENTIFIED UNIT TYPE IN INPUT FILE
    disp('ERROR: Unknown UNIT TYPE in INPUT FILE!');
    disp(' Verify UNIT is "SI" or "British" in Input File. ');
    disp('*** TERMINATING ISSPO PROGRAM ***');
    cd ..\..\..\..

  case 51 % CASE 51 ERROR UNIDENTIFIED PRINTFLG IN INPUT FILE
    disp('ERROR: Unknown PRINTFLG in INPUT FILE!');
    disp(' Verify PRINTFLG is "Y" or "N" in Input File. ');
    disp('*** TERMINATING ISSPO PROGRAM ***');
    cd ..\..\..\..

  case 52 % CASE 52 ERROR UNIDENTIFIED DATA TYPE IN SENSOR_DATA
    disp('ERROR: Unknown SENSOR_DATA ARRAY DATA TYPE!');
disp(' Program erred off in DataTypeVerifier Program.');
disp(' Verify only allowed Data Types are entered in Input File.');
disp(' Data Input Types are case sensitive. Values must be entered in ALL CAPS.');
disp('*** TERMINATING ISSPO PROGRAM ***');

% Return to Main Directory

cd ..\..\..

case 53 % CASE 53 ERROR MASS_LIMIT VARIABLE UNDEFINED IN SENSOR_DATA
disp('ERROR: UNDEFINED MASS_LIMIT VARIABLE IN INPUT FILE!');
disp(' Verify MASS_LIMIT Variable is defined in Input File.');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..

case 54 % CASE 54 ERROR POWER_LIMIT VARIABLE UNDEFINED IN SENSOR_DATA
disp('ERROR: UNDEFINED POWER_LIMIT VARIABLE IN INPUT FILE!');
disp(' Program erred off in DataTypeVerifier Program. ');
disp(' Verify POWER_LIMIT Variable is defined in Input File.');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..

case 55 % CASE 55 ERROR VOLUME_LIMIT VARIABLE UNDEFINED IN SENSOR_DATA
disp('ERROR: VOLUME_LIMIT VARIABLE UNDEFINED IN INPUT FILE!');
disp(' Verify VOLUME_LIMIT Variable is defined in Input File.');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..

% Return to Main Directory
cd ..\..\..

case 56 % CASE 56 ERROR SENSOR OPTION REQUIRES ADDITIONAL DATA IN SENSOR_DATA
disp('ERROR: SENSOR OPTION REQUIRES ADDITIONAL DATA IN SENSOR_DATA ARRAY!');
disp(' Program erred off in DataTypeVerifier Program. ');
disp(' Selected Sensor Option ' 'OPTICS' ' Requires Additional Design Information.');
disp(' Verify Proper Data Input Format for each Sensor Type. ');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..

% Return to Main Directory
cd ..\..\..

case 57 % CASE 57 ERROR SENSOR OPTION OPTICS REQUIRES ADDITIONAL DATA IN SENSOR_DATA
disp('ERROR: SENSOR OPTION OPTICS REQUIRES ADDITIONAL DATA IN SENSOR_DATA ARRAY!');
disp(' Program erred off in DataTypeVerifier Program. ');
disp(' Selected Sensor Option ' 'OPTICS' ' Requires Additional Design Information.');
disp(' Verify Proper Data Input Format for each Sensor Type. ');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..

% Return to Main Directory
cd ..\..\..

case 58 % CASE 58 ERROR SENSOR OPTION OPTICS REQUIRES ADDITIONAL DATA IN SENSOR_DATA
disp('ERROR: SENSOR OPTION OPTICS REQUIRES ADDITIONAL DATA IN SENSOR_DATA ARRAY!');
disp(' Program erred off in DataTypeVerifier Program. ');
disp(' Selected Sensor Option ' 'OPTICS' ' Requires Additional Design Information.');
disp(' Verify Proper Data Input Format for each Sensor Type. ');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..

% Return to Main Directory
cd ..\..\..

case 59 % CASE 59 ERROR SENSOR OPTION OPTICS REQUIRES SYSTEM TYPE IN SENSOR_DATA
disp('ERROR: SENSOR OPTION OPTICS REQUIRES SYSTEM TYPE IN SENSOR_DATA ARRAY!');
disp(' Program erred off in DataTypeVerifier Program. ');
disp(' Selected Sensor Option ' 'OPTICS' ' Requires Additional Design Information.');
disp(' Verify Optical Sensor System Type is defined in 2nd Column position. ');
disp(' Specify Resolution as: ' 'CAMERA' ' 'ARRAY' ' 'LINEAR' ' ');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..

% Return to Main Directory
cd ..\..\..

case 60 % CASE 60 ERROR SENSOR OPTION OPTICS REQUIRES RESOLUTION DATA IN SENSOR_DATA
disp('ERROR: SENSOR OPTION OPTICS REQUIRES RESOLUTION DATA IN SENSOR_DATA ARRAY!');
disp(' Program erred off in DataTypeVerifier Program. ');
disp(' Selected Sensor Option ' 'OPTICS' ' Requires Additional Design Information.');
disp(' Verify Optical Sensor Resolution is defined in 3rd Column position. ');
% Case 61: Error Sensor Option Optics Undefined Image Type in Sensor Data

disp('ERROR: SENSOR OPTION OPTICS UNDEFINED IMAGE TYPE IN SENSOR_DATA ARRAY!');
disp('Selected Sensor Option 'OPTICS'' Requires Additional Design Information.);
disp('Undefined Image Option Type in Optical Sensor Data.');
disp('Specify Image Type as: 'X-RAY' 'VISUAL' 'UV' 'NIR' 'MICRO');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory

cd .. \ .. \\

% Case 62: Error Sensor Option Radiation Requires Additional Data in Sensor Data

disp('ERROR: SENSOR OPTION RADIATION REQUIRES ADDITIONAL DATA IN SENSOR_DATA ARRAY!');
disp('Selected Sensor Option 'RADIATION'' Requires Additional Design Information.);
disp('Verify Proper Data Input Format for each Sensor Type.');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory

cd .. \ .. \\

% Case 63: Error Sensor Option Optics Undefined Radiation Type in Sensor Data

disp('ERROR: SENSOR OPTION OPTICS UNDEFINED RADIATION TYPE IN SENSOR_DATA ARRAY!');
disp('Selected Sensor Option 'RADIATION'' Requires Additional Design Information.);
disp('Undefined RADIATION Option Type in Radiation Sensor Data.');
disp('Specify Image Type as: 'Charged Particle' 'Alpha' 'Beta' 'Gamma' 'X-Ray');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory

cd .. \ .. \\

% Case 64: Error Sensor Option Inclination Requires Additional Data in Sensor Data

disp('ERROR: SENSOR OPTION INCLINATION REQUIRES ADDITIONAL DATA IN SENSOR_DATA ARRAY!');
disp('Selected Sensor Option 'INCLINATION'' Requires Additional Design Information.);
disp('Verify Proper Data Input Format for each Sensor Type.');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory

cd .. \ .. \\

% Case 65: Error Sensor Option Inclination Requires Resolution Data in Sensor Data

disp('ERROR: SENSOR OPTION INCLINATION REQUIRES RESOLUTION DATA IN SENSOR_DATA ARRAY!');
disp('Selected Sensor Option 'INCLINATION'' Requires Additional Design Information.);
disp('Verify Inclination Sensor Range is defined in 2nd Column position.');
disp('Specify Range as: 'LOW' 'MEDIUM' 'HIGH');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory

cd .. \ .. \\

% Case 66: Error Sensor Option Acceleration Requires Additional Data in Sensor Data

disp('ERROR: SENSOR OPTION ACCELERATION REQUIRES ADDITIONAL DATA IN SENSOR_DATA ARRAY!');
disp('Selected Sensor Option 'ACCELERATION'' Requires Additional Design Information.);
disp('Verify Proper Data Input Format for each Sensor Type.');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory

cd .. \ .. \\

% Case 67: Error Sensor Option Acceleration Requires Range Data in Sensor Data

disp('ERROR: SENSOR OPTION ACCELERATION REQUIRES RANGE DATA IN SENSOR_DATA ARRAY!');
disp('Selected Sensor Option 'ACCELERATION'' Requires Additional Design Information.);
disp('ERROR: SENSOR OPTION ACCELERATION REQUIRES ACCELERATION DATA IN SENSOR_DATA ARRAY!');
disp(' Program erred off in DataTypeVerifier Program. ');
disp(' Selected Sensor Option 'ACCELERATION' Requires Additional Design Information. ');
disp(' Verify Acceleration Sensor Range is defined in 2nd Column position. ');
disp(' Specify Range as: 'SOFT' 'MEDIUM' 'HIGH' 'IMPACT' 'BALLISTIC' ' ');
disp('*** TERMINATING ISSPO PROGRAM ***');

% Return to Main Directory
return
cd . . . . . . . . ;
case 68 % CASE 68 ERROR SENSOR OPTION WIND VELOCITY REQUIRES ADDITIONAL DATA IN SENSOR_DATA ARRAY!!;
disp('ERROR: SENSOR OPTION ATMOSPHERE REQUIRES ADDITIONAL DATA IN SENSOR_DATA ARRAY!');
disp(' Program erred off in DataTypeVerifier Program. ');
disp(' Selected Sensor Option 'WIND VELOCITY' Requires Additional Design Information. ');
disp(' Verify Proper Data Input Format for each Sensor Type. ');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
return
cd . . . . . . . . ;
case 69 % CASE 69 ERROR SENSOR OPTION WIND VELOCITY REQUIRES ADDITIONAL DATA IN SENSOR_DATA ARRAY!!;
disp('ERROR: SENSOR OPTION ATMOSPHERE REQUIRES ADDITIONAL DATA IN SENSOR_DATA ARRAY!');
disp(' Program erred off in DataTypeVerifier Program. ');
disp(' Selected Sensor Option 'WIND VELOCITY' Requires Additional Design Information. ');
disp(' Verify Wind Velocity Sensor type is defined in the next Column position. ');
disp(' Specify Type as: 'ANEMOMETER' or 'DOPPLER' ' ');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
return
cd . . . . . . . . ;
case 70 % CASE 70 ERROR SENSOR OPTION GAS ANALYSIS REQUIRES ADDITIONAL DATA IN SENSOR_DATA ARRAY!!;
disp('ERROR: SENSOR OPTION GAS ANALYSIS REQUIRES ADDITIONAL DATA IN SENSOR_DATA ARRAY!');
disp(' Program erred off in DataTypeVerifier Program. ');
disp(' Selected Sensor Option 'GAS ANALYSIS' Requires Additional Design Information. ');
disp(' Verify GCMS Sensor type is defined in the 2nd Column position. ');
disp(' Specify Type as: 'WAVELENGTH' or 'MASS-CHARGE' ' ');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
return
cd . . . . . . . . ;
case 71 % CASE 71 ERROR SENSOR OPTION GAS ANALYSIS REQUIRES ADDITIONAL DATA IN SENSOR_DATA ARRAY!!;
disp('ERROR: SENSOR OPTION GAS ANALYSIS REQUIRES ADDITIONAL DATA IN SENSOR_DATA ARRAY!');
disp(' Program erred off in DataTypeVerifier Program. ');
disp(' Selected Sensor Option 'GAS ANALYSIS' Requires Additional Design Information. ');
disp(' Selected Sensing Type 'MASS-CHARGE' Option Requires input in the 3rd Column. ');
disp(' Verify Proper Data Input Format for each Sensor Type. ');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
return
cd . . . . . . . . ;
case 72 % CASE 72 ERROR SENSOR OPTION GAS ANALYSIS REQUIRES ADDITIONAL DATA IN SENSOR_DATA ARRAY!!;
disp('ERROR: SENSOR OPTION GAS ANALYSIS REQUIRES ADDITIONAL DATA IN SENSOR_DATA ARRAY!');
disp(' Program erred off in DataTypeVerifier Program. ');
disp(' Selected Sensor Option 'GAS ANALYSIS' Requires Additional Design Information. ');
disp(' Selected Sensing Type 'MASS-CHARGE' Option Requires input in the 3rd Column. ');
disp(' Verify Gas Analysis Sensor Range is defined in 3rd Column position. ');
disp(' Specify Range as: 'LOW' 'MEDIUM' 'HIGH' ' ');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
return
cd . . . . . . . . ;
case 73 % CASE 73 ERROR SENSOR OPTION ACCELERATION REQUIRES ADDITIONAL DATA IN SENSOR_DATA
disp( 'ERROR: SENSOR OPTION ACCELERATION REQUIRES ADDITIONAL DATA IN SENSOR_DATA
ARRAY( )' );
disp( 'Selected Sensor Option ' ACCELERATION ' Requires Additional Design
Information.' );
disp( 'Verify Acceleration Range in 2nd Column Position, Power Type in 3rd
Column.' );
disp( 'Verify Proper Data Input Format for each Sensor Type.' );
disp( '*** TERMINATING ISSPO PROGRAM ***' );
% Return to Main Directory
cd .. \ .. \ .. ;
case 74
% CASE 74 ERROR SENSOR OPTION ACCELERATION REQUIRES POWER TYPE DATA IN
SENSOR_DATA
disp( 'ERROR: SENSOR OPTION ACCELERATION REQUIRES ACCELERATION DATA IN
SENSOR_DATA ARRAY( )' );
disp( 'Program erred off in DataTypeVerifier Program.' );
disp( 'Selected Sensor Option ' ACCELERATION ' Requires Additional Design
Information.' );
disp( 'Verify Acceleration Sensor Power Type is defined in 3rd Column
position.' );
disp( 'Specify Type as: ' VOLTAGE ' or ' SENSOR ' or ' ARRAY ' or 'SELF GENERATING' 
' );
disp( '*** TERMINATING ISSPO PROGRAM ***' );
% Return to Main Directory
cd .. \ .. \ .. ;
case 75
% CASE 75 ERROR SENSOR OPTION TEMPERATURE REQUIRES ADDITIONAL DATA IN
SENSOR_DATA
disp( 'ERROR: SENSOR OPTION ATMOSPHERE REQUIRES ADDITIONAL DATA IN SENSOR_DATA
ARRAY( )' );
disp( 'Program erred off in DataTypeVerifier Program.' );
disp( 'Selected Sensor Option ' TEMPERATURE ' Requires Additional Design
Information.' );
disp( 'Verify Temperature Sensor type is defined in the next Column
position.' );
disp( 'Specify Type as: ' VOLTAGE ' or ' RESISTANCE ' or ' ' );
disp( '*** TERMINATING ISSPO PROGRAM ***' );
% Return to Main Directory
cd .. \ .. \ .. ;
case 76
% CASE 76 ERROR SENSOR OPTION TEMPERATURE REQUIRES ADDITIONAL DATA IN
SENSOR_DATA
disp( 'ERROR: SENSOR OPTION ATMOSPHERE REQUIRES ADDITIONAL DATA IN SENSOR_DATA
ARRAY( )' );
disp( 'Program erred off in DataTypeVerifier Program.' );
disp( 'Selected Sensor Option ' TEMPERATURE ' Requires Additional Design
Information.' );
disp( 'Verify Temperature Sensor type is defined in the next Column
position.' );
disp( 'Specify Type as: ' VOLTAGE ' or ' RESISTANCE ' or ' ' );
disp( '*** TERMINATING ISSPO PROGRAM ***' );
% Return to Main Directory
cd .. \ .. \ .. ;
case 77
% CASE 77 ERROR SENSOR OPTION ACOUSTICS REQUIRES ADDITIONAL DATA IN
SENSOR_DATA
disp( 'ERROR: SENSOR OPTION ACOUSTICS REQUIRES ADDITIONAL DATA IN SENSOR_DATA
ARRAY( )' );
disp( 'Program erred off in DataTypeVerifier Program.' );
disp( 'Selected Sensor Option ' ACOUSTICS ' Requires Additional Design
Information.' );
disp( 'Verify Acoustic Sensor type is defined in the next 2nd position.' );
disp( 'Specify Type as: ' SENSOR ' or ' ARRAY ' or ' ' );
disp( '*** TERMINATING ISSPO PROGRAM ***' );
% Return to Main Directory
cd .. \ .. \ .. ;
case 78
% CASE 78 ERROR SENSOR OPTION ACOUSTICS REQUIRES ADDITIONAL DATA IN
SENSOR_DATA
disp( 'ERROR: SENSOR OPTION ACOUSTICS REQUIRES ADDITIONAL DATA IN SENSOR_DATA
ARRAY( )' );
disp( 'Program erred off in DataTypeVerifier Program.' );
disp( 'Selected Sensor Option ' ACOUSTICS ' Requires Additional Design
Information.' );
disp( 'Verify Acoustic Sensor type is defined in the next 2nd position.' );
disp( 'Specify Type as: ' SENSOR ' or ' ARRAY ' or ' ' );
disp( '*** TERMINATING ISSPO PROGRAM ***' );
% Return to Main Directory
cd .. \ .. \ .. ;
case 79
% CASE 79 ERROR SENSOR OPTION DENSITY REQUIRES ADDITIONAL DATA IN
SENSOR_DATA
disp( 'ERROR: SENSOR OPTION ATMOSPHERE REQUIRES ADDITIONAL DATA IN SENSOR_DATA
ARRAY( )' );

disp('Program erred off in DataTypeVerifier Program.');
disp('Selected Sensor Option 'DENSITY' Requires Additional Design Information.);
disp('Verify Proper Data Input Format for each Sensor Type.');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd \ \ \ \;

% CASE 80 % CASE 80 ERROR SENSOR OPTION SENSITY REQUIRES ADDITIONAL DATA IN SENSOR_DATA

% CASE 81 % CASE 81 ERROR OPTION DIGITAL SIGNAL PROCESSING REQUIRES ADDITIONAL DATA IN SENSOR_DATA

% CASE 82 % CASE 82 ERROR OPTION DIGITAL SIGNAL PROCESSING REQUIRES ADDITIONAL DATA IN SENSOR_DATA

% CASE 83 % CASE 83 ERROR OPTION DIGITAL SIGNAL PROCESSING REQUIRES ADDITIONAL DATA IN SENSOR_DATA

% CASE 84 % CASE 84 ERROR OPTION DIGITAL SIGNAL PROCESSING REQUIRES ADDITIONAL DATA IN SENSOR_DATA

% CASE 1XX SENSOR RESULT DATA FILES NOT CREATED PROPERLY

% ERROR 100 - TEMPERATURE SENSOR SUMMARY DATA FILE FAILED TO BE CREATED

368
case 101 % ERROR 101 - PRESSURE SENSOR SUMMARY DATA FILE FAILED TO BE CREATED
disp('ERROR: PRESSURE SENSOR Database File Failed to be Created!');
disp(' Program erred off in PRESSURE_SENSORS subroutine. ');
disp(' Verify PRESSURE_SENSORS program is correctly creating the database file.');
disp('*** TERMINATING ISSPO PROGRAM ****');
% Return to Main Directory
cd .. \\

case 102 % ERROR 102 - NEPHELOMETER SENSOR SUMMARY DATA FILE FAILED TO BE CREATED
disp('ERROR: NEPHELOMETER SENSOR Database File Failed to be Created!');
disp(' Program erred off in NEPHELOMETER_SENSORS subroutine. ');
disp(' Verify NEPHELOMETER_SENSORS program is correctly creating the database file.');
disp('*** TERMINATING ISSPO PROGRAM ****');
% Return to Main Directory
cd .. \\

case 103 % ERROR 103 - HUMIDITY SENSOR SUMMARY DATA FILE FAILED TO BE CREATED
disp('ERROR: HUMIDITY SENSOR Database File Failed to be Created!');
disp(' Program erred off in HUMIDITY_SENSORS subroutine. ');
disp(' Verify HUMIDITY_SENSORS program is correctly creating the database file.');
disp('*** TERMINATING ISSPO PROGRAM ****');
% Return to Main Directory
cd .. \\

case 104 % ERROR 104 - DENSITY SENSOR SUMMARY DATA FILE FAILED TO BE CREATED
disp('ERROR: DENSITY SENSOR Database File Failed to be Created!');
disp(' Program erred off in DENSITY_SENSORS subroutine. ');
disp(' Verify DENSITY_SENSORS program is correctly creating the database file.');
disp('*** TERMINATING ISSPO PROGRAM ****');
% Return to Main Directory
cd .. \\

case 105 % ERROR 105 - ACOUSTIC SENSOR SUMMARY DATA FILE FAILED TO BE CREATED
disp('ERROR: ACOUSTIC SENSOR Database File Failed to be Created!');
disp(' Program erred off in ACOUSTIC_SENSORS subroutine. ');
disp(' Verify ACOUSTIC_SENSORS program is correctly creating the database file.');
disp('*** TERMINATING ISSPO PROGRAM ****');
% Return to Main Directory
cd .. \\

case 106 % ERROR 106 - AC FIELD SENSOR SUMMARY DATA FILE FAILED TO BE CREATED
disp('ERROR: AC FIELD SENSOR Database File Failed to be Created!');
disp(' Program erred off in AC_FIELD_SENSORS subroutine. ');
disp(' Verify AC_FIELD_SENSORS program is correctly creating the database file.');
disp('*** TERMINATING ISSPO PROGRAM ****');
% Return to Main Directory
cd .. \\

case 107 % ERROR 107 - WIND VELOCITY SENSOR SUMMARY DATA FILE FAILED TO BE CREATED
disp('ERROR: WIND VELOCITY SENSOR Database File Failed to be Created!');
disp(' Program erred off in WIND_VELOCITY_SENSORS subroutine. ');
disp(' Verify WIND_VELOCITY_SENSORS program is correctly creating the database file.');
disp('*** TERMINATING ISSPO PROGRAM ****');
% Return to Main Directory
cd .. \\

case 108 % ERROR 108 - ACCELEROMETER SENSOR SUMMARY DATA FILE FAILED TO BE CREATED
disp('ERROR: ACCELEROMETER SENSOR Database File Failed to be Created!');
disp(' Program erred off in ACCELEROMETER_SENSORS subroutine. ');
disp(' Verify ACCELEROMETER_SENSORS program is correctly creating the database file.');
disp('*** TERMINATING ISSPO PROGRAM ****');
% Return to Main Directory
cd .. \\

case 109 % ERROR 109 - GCMS SENSOR SUMMARY DATA FILE FAILED TO BE CREATED
disp('ERROR: GCMS SENSOR Database File Failed to be Created!');
disp(' Program erred off in GCMS_SENSORS subroutine. ');
disp(' Verify GCMS_SENSORS program is correctly creating the database file.');
disp('*** TERMINATING ISSPO PROGRAM ****');
% Return to Main Directory
cd .. \\

case 110 % ERROR 110 - RADIATION SENSOR SUMMARY DATA FILE FAILED TO BE CREATED
disp('ERROR: RADIATION SENSOR Database File Failed to be Created!');

disp('Program erred off in RADIATION_SENSORS subroutine.');

disp('Verify RADIATION_SENSORS program is correctly creating the database file.');

disp('*** TERMINATING ISSPO PROGRAM ***
% Return to Main Directory

cd ..\..\..;

case 11
% ERROR 111 - INCLINOMETER SENSOR SUMMARY DATA FILE FAILED TO BE CREATED

disp('ERROR: INCLINOMETER SENSOR Database File Failed to be Created!');

disp('Program erred off in INCLINOMETER_SENSORS subroutine.');

disp('Verify INCLINOMETER_SENSORS program is correctly creating the database file.');

disp('*** TERMINATING ISSPO PROGRAM ***
% Return to Main Directory

cd ..\..\..;

case 112
% ERROR 112 - IMAGING SENSOR SUMMARY DATA FILE FAILED TO BE CREATED

disp('ERROR: IMAGING SENSOR Database File Failed to be Created!');

disp('Program erred off in IMAGING_SENSORS subroutine.');

disp('Verify IMAGING_SENSORS program is correctly creating the database file.');

disp('*** TERMINATING ISSPO PROGRAM ***
% Return to Main Directory

cd ..\..\..;

case 113
% ERROR 113 - REFRACTION SENSOR SUMMARY DATA FILE FAILED TO BE CREATED

disp('ERROR: REFRACTION SENSOR Database File Failed to be Created!');

disp('Program erred off in REFRACTION_SENSORS subroutine.');

disp('Verify REFRACTION_SENSORS program is correctly creating the database file.');

disp('*** TERMINATING ISSPO PROGRAM ***
% Return to Main Directory

cd ..\..\..;

case 114
% ERROR 114 - DATA PROCESSING SUMMARY DATA FILE FAILED TO BE CREATED

disp('ERROR: DATA PROCESSING Database File Failed to be Created!');

disp('Program erred off in DIGITAL_SIGNAL_PROCESSING subroutine.');

disp('Verify DIGITAL_SIGNAL_PROCESSING program is correctly creating the database file.');

disp('*** TERMINATING ISSPO PROGRAM ***
% Return to Main Directory

cd ..\..\..;

case 200
% ERROR 200 - TEMPERATURE SENSOR RESULT SUMMARY FILE FAILED TO BE CREATED

disp('ERROR: TEMP SENSOR SUMMARY File Failed to be Created!');

disp('Program erred off in writing summary file in TEMP_SENSORS subroutine.
');

disp('Verify TEMP_SENSORS program is correctly creating the summary file.');

disp('*** TERMINATING ISSPO PROGRAM ***
% Return to Main Directory

cd ..\..\..;

case 201
% ERROR 201 - PRESSURE SENSOR RESULT SUMMARY FILE FAILED TO BE CREATED

disp('ERROR: PRESSURE SENSOR SUMMARY File Failed to be Created!');

disp('Program erred off in writing summary file in PRESSURE_SENSORS subroutine.
');

disp('Verify PRESSURE_SENSORS program is correctly creating the summary file.');

disp('*** TERMINATING ISSPO PROGRAM ***
% Return to Main Directory

cd ..\..\..;

case 202
% ERROR 202 - NEPHELOMETER SENSOR RESULT SUMMARY FILE FAILED TO BE CREATED

disp('ERROR: NEPHELOMETER SENSOR SUMMARY File Failed to be Created!');

disp('Program erred off in writing summary file in NEPHELOMETER_SENSORS subroutine.
');

disp('Verify NEPHELOMETER_SENSORS program is correctly creating the summary file.');

disp('*** TERMINATING ISSPO PROGRAM ***
% Return to Main Directory

cd ..\..\..;

case 203
% ERROR 203 - HUMIDITY SENSOR RESULT SUMMARY FILE FAILED TO BE CREATED

disp('ERROR: HUMIDITY SENSOR SUMMARY File Failed to be Created!');

disp('Program erred off in writing summary file in HUMIDITY_SENSORS subroutine.
');

disp('Verify HUMIDITY_SENSORS program is correctly creating the summary file.');

disp('*** TERMINATING ISSPO PROGRAM ***
% Return to Main Directory

cd ..\..\..;

370
case 204 % ERROR 204 - DENSITY SENSOR RESULT SUMMARY FILE FAILED TO BE CREATED
disp('ERROR: DENSITY SENSOR SUMMARY File Failed to be Created!');
disp(' Program erred off in writing summary file in DENSITY_SENSORS subroutine.
');
   disp(' Verify DENSITY_SENSORS program is correctly creating the summary file.
');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..
;
case 205 % ERROR 205 - ACOUSTIC SENSOR RESULT SUMMARY FILE FAILED TO BE CREATED
disp('ERROR: ACOUSTIC SENSOR SUMMARY File Failed to be Created!');
disp(' Program erred off in writing summary file in ACOUSTIC_SENSORS subroutine.
');
   disp(' Verify ACOUSTIC_SENSORS program is correctly creating the summary file.
');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..
;
case 206 % ERROR 206 - AC FIELD SENSOR RESULT SUMMARY FILE FAILED TO BE CREATED
disp('ERROR: AC FIELD SENSOR SUMMARY File Failed to be Created!');
disp(' Program erred off in writing summary file in AC_FIELD_SENSORS subroutine.
');
   disp(' Verify AC_FIELD_SENSORS program is correctly creating the summary file.
');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..
;
case 207 % ERROR 207 - WIND VELOCITY SENSOR RESULT SUMMARY FILE FAILED TO BE CREATED
disp('ERROR: WIND VELOCITY SENSOR SUMMARY File Failed to be Created!');
disp(' Program erred off in writing summary file in WIND VELOCITY_SENSORS subroutine.
');
   disp(' Verify WIND_VELOCITY_SENSORS program is correctly creating the summary file.
');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..
;
case 208 % ERROR 208 - ACCELEROMETER SENSOR RESULT SUMMARY FILE FAILED TO BE CREATED
disp('ERROR: ACCELEROMETER SENSOR SUMMARY File Failed to be Created!');
disp(' Program erred off in writing summary file in ACCELEROMETER_SENSORS subroutine.
');
   disp(' Verify ACCELEROMETER_SENSORS program is correctly creating the summary file.
');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..
;
case 209 % ERROR 209 - GCMS SENSOR RESULT SUMMARY FILE FAILED TO BE CREATED
disp('ERROR: GCMS SENSOR SUMMARY File Failed to be Created!');
disp(' Program erred off in writing summary file in GCMS_SENSORS subroutine.
');
   disp(' Verify GCMS_SENSORS program is correctly creating the summary file.
');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..
;
case 210 % ERROR 210 - RADIATION SENSOR RESULT SUMMARY FILE FAILED TO BE CREATED
disp('ERROR: RADIATION SENSOR SUMMARY File Failed to be Created!');
disp(' Program erred off in writing summary file in RADIATION_SENSORS subroutine.
');
   disp(' Verify RADIATION_SENSORS program is correctly creating the summary file.
');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..
;
case 211 % ERROR 211 - INCLINOMETER SENSOR RESULT SUMMARY FILE FAILED TO BE CREATED
disp('ERROR: INCLINOMETER SENSOR SUMMARY File Failed to be Created!');
disp(' Program erred off in writing summary file in INCLINOMETER_SENSORS subroutine.
');
   disp(' Verify INCLINOMETER_SENSORS program is correctly creating the summary file.
');
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..
;
case 212 % ERROR 212 - IMAGING SENSOR RESULT SUMMARY FILE FAILED TO BE CREATED

disp('ERROR: IMAGING SENSOR SUMMARY File Failed to be Created!');
disp(' Program erred off in writing summary file in IMAGING_SENSORS subroutine.
');
disp(' Verify IMAGING_SENSORS program is correctly creating the summary
file.');</disp>
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..\..;
case 213 % ERROR 213 - REFRACTION SENSOR RESULT SUMMARY FILE FAILED TO BE CREATED
disp('ERROR: REFRACTION SENSOR SUMMARY File Failed to be Created!');
disp(' Program erred off in writing summary file in REFRACTION_SENSORS
subroutine. ');
disp(' Verify REFRACTION_SENSORS program is correctly creating the summary
file.');</disp>
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..\..;
case 214 % ERROR 215 - DATA PROCESSING RESULT SUMMARY FILE FAILED TO BE CREATED
disp('ERROR: DIGITAL SIGNAL PROCESSING SUMMARY File Failed to be Created!');
disp(' Program erred off in writing summary file in DIGITAL_SIGNAL_PROCESSING
subroutine. ');
disp(' Verify DIGITAL_SIGNAL_PROCESSING program is correctly creating the summary
file.');</disp>
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..\..;
case 215 % ERROR 216 - MASS PROPERTY SUMMARY DATA FILE FAILED TO BE CREATED
disp('ERROR: MASS PROPERTIES SUMMARY File Failed to be Created!');
disp(' Program erred off in MASS_PROPERTIES subroutine. ');
disp(' Verify MASS_PROPERTIES program is correctly creating the summary
variables file.');</disp>
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..\..;
case 216 % ERROR 217 - POWER PROPERTY SUMMARY DATA FILE FAILED TO BE CREATED
disp('ERROR: POWER PROPERTIES SUMMARY File Failed to be Created!');
disp(' Program erred off in POWER_PROPERTIES subroutine. ');
disp(' Verify POWER_PROPERTIES program is correctly creating the summary
variables file.');</disp>
disp('*** TERMINATING ISSPO PROGRAM ***');
% Return to Main Directory
cd ..\..\..\..;
otherwise
disp('*** UNKNOWN ERROR DETECTED ***');
disp('Report ERROR and Case Configuration to Developer for Resolution!');
tfprintf(' ERROR CODE: %3d', ERR_Code);
cd ..\..\..\..;
end

% Plays Error Sound Warning
%SoundPath = ['\PRGM_FILES\SOUNDS' ]
if ERR_Code ~= 1
    cd \PRGM_FILES\
end
% % Return to Main Directory
% cd ..\..\..\..;

372
function MASS_PROPERTIES(DataArray, UNITS)
% Load Data files in to program for use
load('Constants_DB.mat');
% Load Final sensor Matrix values into Main Program
for i = 1:size(DataArray,1)
    switch DataArray{i,1}
    case 'ATMOSPHERE'
        load('TEMP_SENSOR_Final.mat')
        load('PRESSURE_SENSOR_Final.mat')
        load('WIND_VELOCITY_SENSOR_Final.mat')
        load('HUMIDITY_SENSOR_Final.mat')
        load('DENSITY_SENSOR_Final.mat')
    case 'ACCELERATION'
        load('ACCELEROMETER_SENSOR_Final.mat')
    case 'ACOUSTICS'
        load('ACOUSTIC_SENSOR_Final.mat')
    case 'EM FIELD'
        load('AC_FIELD_SENSOR_Final.mat')
    case 'GAS ANALYSIS'
        load('GCMS_SENSOR_Final.mat')
    case 'INCLINATION'
        load('INCLINOMETER_SENSOR_Final.mat')
    case 'NEPHELOMETRY'
        load('NEPHELOMETER_SENSOR_Final.mat')
    case 'OPTICS'
        load('IMAGING_SENSOR_Final.mat')
    case 'RADIATION'
        load('RADIATION_SENSOR_Final.mat')
    case 'REFRACTION'
        load('REFRACTION_SENSOR_Final.mat')
    case 'DATA PROCESSING'
        load('DIGITAL_SIGNAL_PROCESSING_Final.mat')
    otherwise
end
end

end

% Load Final sensor Matrix values into Main Program
k = 1;
for i = 1:size(DataArray,1)
    switch DataArray{i, 1}
        case 'ATMOSPHERE'
            if strcmpi(UNITS,'SI') == 1
                MASS_PROPS(k,1:4) = [ MATRIX2{1,5}/gr2kg MATRIX2{1,6}/mm2m MATRIX2{1,7}/mm2m ];
                P_MATRIX6{1,29}/mm2m ];
            % Pressure Sensor - converts gr to kg and mm to meters
            end
            if strcmpi(WVJTYPE,'ANEMOMETER') ^= 1
                MASS_PROPS(k+2,1:4) = [ W_MATRIX3{1,12} ];
                W_MATRIX3{1,16}/mm2m ];
            end
            if strcmpi(WVJTYPE,'DOPPLER') == 1
                MASS_PROPS(k+2,1:4) = [ W_MATRIX3{1,24} ];
            end
            MASS_PROPS(k+3,1:4) = [ H_MATRIX{1,18}/gr2kg H_MATRIX{1,17}/mm2m ];
        end
        case 'ACCELERATION'
            if strcmpi(UNITS, 'SI') == 1
                MASS_PROPS(k,1:4) = [ ACC_MATRIX6{1,20}/gr2kg ACC_MATRIX6{1,21}/cm2m ACC_MATRIX6{1,22}/cm2m ];
            if strcmpi(WVJTYPE,'DOPPLER') == 1
                MASS_PROPS(k+2,1:4) = [ W_MATRIX3{1,12} ];
            end
            end
    end
end
```matlab
k = k + 1;
end
if strcmpi(UNITS, 'British') == 1
    MASS_PROPS(k, 1:4) = [ ACC_MATRIX6{1,20}/oz21bm ACC_MATRIX6{1,21}/in2ft ACC_MATRIX6{1,22}/in2ft ACC_MATRIX6{1,23}/in2ft ];
k = k + 1;
end
case 'ACOUSTICS'
if strcmpi(UNITS, 'SI') == 1
    MASS_PROPS(k, 1:4) = [ ACS_MATRIX4{1,28}/gr2kg ACS_MATRIX4{1,25}/mm2m ACS_MATRIX4{1,26}/mm2m ACS_MATRIX4{1,27}/mm2m ];
end
if strcmpi(UNITS, 'British') == 1
    MASS_PROPS(k, 1:4) = [ ACS_MATRIX4{1,28}/oz21bm ACS_MATRIX4{1,25}/in2ft ACS_MATRIX4{1,26}/in2ft ACS_MATRIX4{1,27}/in2ft ];
k = k + 1;
end
if strcmpi(UNITS, 'British') == 1
    if strcmpi(UNITS, 'SI') == 1
        MASS_PROPS(k, 1:4) = [ ACS_MATRIX4{1,28}/gr2kg ACS_MATRIX4{1,25}/mm2m ACS_MATRIX4{1,26}/mm2m ACS_MATRIX4{1,27}/mm2m ];
    end
end
end
if strcmpi(UNITS, 'British') == 1
    if strcmpi(UNITS, 'SI') == 1
        MASS_PROPS(k, 1:4) = [ ACS_MATRIX4{1,28}/oz21bm ACS_MATRIX4{1,25}/in2ft ACS_MATRIX4{1,26}/in2ft ACS_MATRIX4{1,27}/in2ft ];
    end
end
end
if strcmpi(UNITS, 'British') == 1
    if strcmpi(UNITS, 'SI') == 1
        MASS_PROPS(k, 1:4) = [ ACS_MATRIX4{1,28}/gr2kg ACS_MATRIX4{1,25}/mm2m ACS_MATRIX4{1,26}/mm2m ACS_MATRIX4{1,27}/mm2m ];
    end
end
end
end
```
MASS_PROPS(k, 1:4) = [ GCMS_MATRIX3{1,33} GCMS_MATRIX3{1,25}/in2ft GCMS_MATRIX3{1,26}/in2ft
GCMS_MATRIX3{1,27}/in2ft ];
  k = k + 1;
end

case 'INCLINATION'
  if strcmpi(UNITS, 'SI') == 1
    MASS_PROPS(k, 1:4) = [ INCL_MATRIX5{1,19}/gr2kg INCL_MATRIX5{1,16}/mm2m INCL_MATRIX5{1,17}/mm2m
    INCL_MATRIX5{1,18}/mm2m ];
    k = k + 1;
  end
  if strcmpi(UNITS, 'British') == 1
    MASS_PROPS(k, 1:4) = [ INCL_MATRIX5{1,19}/oz21bm INCL_MATRIX5{1,16}/in2ft INCL_MATRIX5{1,17}/in2ft
    INCL_MATRIX5{1,18}/in2ft ];
    k = k + 1;
  end
end

case 'NEPHELOMETRY'
  if strcmpi(UNITS, 'SI') == 1
    MASS_PROPS(k, 1:4) = [ N_MATRIX3{1,6} N_MATRIX3{1,11}/mm2m N_MATRIX3{1,9}/mm2m N_MATRIX3{1,10}/mm2m
    N_MATRIX3{1,12}/mm2m ];
    k = k + 1;
  end
  if strcmpi(UNITS, 'British') == 1
    MASS_PROPS(k, 1:4) = [ N_MATRIX3{1,6} N_MATRIX3{1,11}/in2ft N_MATRIX3{1,9}/in2ft N_MATRIX3{1,10}/in2ft
    N_MATRIX3{1,12}/in2ft ];
    k = k + 1;
  end
end

case 'OPTICS'
  if strcmpi(UNITS, 'SI') == 1
    MASS_PROPS(k, 1:4) = [ IMG_MATRIX5{1,26} IMG_MATRIX5{1,23}/mm2m IMG_MATRIX5{1,24}/mm2m
    IMG_MATRIX5{1,25}/mm2m ];
    k = k + 1;
  end
  if strcmpi(UNITS, 'British') == 1
    MASS_PROPS(k, 1:4) = [ IMG_MATRIX5{1,26} IMG_MATRIX5{1,23}/in2ft IMG_MATRIX5{1,24}/in2ft
    IMG_MATRIX5{1,25}/in2ft ];
    k = k + 1;
  end
end

case 'RADIATION'
  if strcmpi(UNITS, 'SI') == 1
    MASS_PROPS(k, 1:4) = [ RAD_MATRIX3{1,26} RAD_MATRIX3{1,27}/mm2m RAD_MATRIX3{1,28}/mm2m
    RAD_MATRIX3{1,29}/mm2m ];
    k = k + 1;
  end
  if strcmpi(UNITS, 'British') == 1
    MASS_PROPS(k, 1:4) = [ RAD_MATRIX3{1,26} RAD_MATRIX3{1,27}/oz21bm RAD_MATRIX3{1,28}/in2ft
    RAD_MATRIX3{1,29}/in2ft ];
    k = k + 1;
  end
end

case 'REFRACTION'
  if strcmpi(UNITS, 'SI') == 1
    MASS_PROPS(k, 1:4) = [ REF_MATRIX3{1,7}/gr2kg REF_MATRIX3{1,12}/cm2m REF_MATRIX3{1,13}/cm2m
    REF_MATRIX3{1,14}/cm2m ];
    k = k + 1;
  end
  if strcmpi(UNITS, 'British') == 1
    MASS_PROPS(k, 1:4) = [ REF_MATRIX3{1,7}/oz21bm REF_MATRIX3{1,12}/oz21bm REF_MATRIX3{1,13}/oz21bm
    REF_MATRIX3{1,14}/oz21bm ];
    k = k + 1;
  end
end
function POWER_PROPERTIES(DataArray, UNITS)
% Sensor Power Property Database
% Developed by: Keith Schreck
% Mechanical and Aerospace Engineering
% San Jose State University
% Date: Fall 2007
% This file contains the POWER PROPERTY calculation for the ISSPO program.
% Data here is based on the sensors selected in the ISSPO Case Study and loads
% the power requirement data into a sensor power array and records values
% for all the sensor components chosen in the Sensor Data file for the main program.
% Values are assigned to a variable name and saved to a MatLAB '.mat' file
% and loaded when calling the ISSPO program.
% Cases where Power data is not available and Voltage - Current information is
% available, Power requirements are calculated based on Ohm's Law:
% \[ P = V \times I \] in Watts

D22. Power Properties Program
% Load Data files into program for use
load('Constants_DB.mat');

% Load Final sensor Matrix values into Main Program
for i = 1:size(DataArray,1)
    switch DataArray{i,1}
    case 'ATMOSPHERE'
        load('TEMP_SENSOR_Final.mat')
        load('PRESSURE_SENSOR_Final.mat')
        load('WIND_VELOCITY_SENSOR_Final.mat')
        load('HUMIDITY_SENSOR_Final.mat')
        load('DENSITY_SENSOR_Final.mat')
    case 'ACCELERATION'
        load('ACCELEROMETER_SENSOR_Final.mat')
    case 'ACOUSTICS'
        load('ACOUSTIC_SENSOR_Final.mat')
    case 'EM FIELD'
        load('AC_FIELD_SENSOR_Final.mat')
    case 'GAS ANALYSIS'
        load('GCMS_SENSOR_Final.mat')
    case 'INCLINATION'
        load('INCLINOMETER_SENSOR_Final.mat')
    case 'NEPHELOMETRY'
        load('NEPHELOMETER_SENSOR_Final.mat')
    case 'OPTICS'
        load('IMAGING_SENSOR_Final.mat')
    case 'RADIATION'
        load('RADIATION_SENSOR_Final.mat')
    case 'REFRACTION'
        load('REFRACTI0N_SENSOR_Final.mat')
    case 'DATA PROCESSING'
        load('DIGITAL_SIGNAL_PROCESSING_Final.mat')
    otherwise
    end
end

% Load Final sensor Matrix values into Main Program
k = 1;
for i = 1:size(DataArray,1)
    switch DataArray{i,1}
    case 'ATMOSPHERE'
        if strcmpi(UNITS,'SI') == 1
            POWER_PROPS(k ,1) = [ MATRIX2(1,4) ];
        end
        if strcmpi(WV_TYPE,'ANEMOMETER') == 1
            POWER_PROPS2(1 ,1) = [ 0 ]; % Pressure Sensor - converts V & mA to W
        elseif strcmpi(WV_TYPE,'DOPPLER') == 1
            POWER_PROPS(k+2,1) = [ W_MATRIX3{1,23}/mA2A ]; % Pressure Sensor - converts V & mA to W
        end
    end
end
end
POWER_PROPS(k+1,1) = [ 0 ]; % Humidity Sensor - No Voltage or Current Data
POWER_PROPS(k+4,1) = [ D_MATRIX6{1,14} * D_MATRIX6{1,15}/mA2A ]; % - converts V & mA to W
k = k + 5;
end
if strcmpi(UNITS, 'British') == 1
POWER_PROPS(k ,1) = [ MATRIX2{1,4} ];
POWER_PROPS(k+1,1) = [ P_MATRIX6{1,14} * P_MATRIX6{1,15}/mA2A * BTU_hr2W ]; % Pressure Sensor - converts V & mA to W to BTU/hr
end
if strcmpi(WV_TYPE, 'ANEMOMETER') == 1
POWER_PROPS(k+2,1) = [ 0 ]; % Probe Dimensions - Voltage Only no Current Data
POWER_PROPS2(1 ,1) = [ 0 ]; % Handheld Dimensions
end
if strcmpi(WV_TYPE, 'DOPPLER') == 1
POWER_PROPS(k+2,1) = [ W_MATRIX3{1,21} ];
end
POWER_PROPS(k+3,1) = [ 0 ]; % Humidity Sensor - No Voltage or Current Data
POWER_PROPS(k+4,1) = [ D_MATRIX6{1,14} * D_MATRIX6{1,15}/mA2A * BTU_hr2W ]; % - converts V & mA to W to BTU/hr
k = k + 5;
case 'ACCELERATION'
if strcmpi(UNITS, 'SI') == 1
if ACC_MATRIX6{1,25} ~= 0
POWER_PROPS(k ,1) = [ ACC_MATRIX6{1,25} ]; % W
else
POWER_PROPS(k ,1) = [ ACC_MATRIX6{1,14} * ACC_MATRIX6{1,15}/mA2A ]; % - converts V & mA to W
end
k = k + 1;
end
if strcmpi(UNITS, 'British') == 1
if ACC_MATRIX6{1,25} ~= 0
POWER_PROPS(k ,1) = [ ACC_MATRIX6{1,25} ]; % BTU's/hr
else
POWER_PROPS(k ,1) = [ ACC_MATRIX6{1,14} * ACC_MATRIX6{1,15}/mA2A * BTU_hr2W ]; % - converts V & mA to W to BTU/hr
end
k = k + 1;
end
case 'ACOUSTICS'
if strcmpi(UNITS, 'SI') == 1
POWER_PROPS(k ,1) = [ ACS_MATRIX4{1,13} * ACS_MATRIX4{1,14}/mA2A ]; % - converts V & mA to W
k = k + 1;
end
if strcmpi(UNITS, 'British') == 1
POWER_PROPS(k ,1) = [ ACS_MATRIX4{1,13} * ACS_MATRIX4{1,14}/mA2A * BTU_hr2W ]; % - converts V & mA to W to BTU/hr
k = k + 1;
end
case 'EM FIELD'
% EM FIELD Sensor
if strcmpi(UNITS, 'SI') == 1
POWER_PROPS(k ,1) = [ 0 ]; % Probe Dimensions - Voltage Only no Current Data
POWER_PROPS2(1 ,1) = [ 0 ]; % Handheld Dimensions
k = k + 1;
end
if strcmpi(UNITS,'British') == 1
    POWER_PROPS(k,1) = [ 0 ]; % Probe Dimensions
    POWER_PROPS3(1,1) = [ 0 ]; % Handheld Dimensions
    k = k + 1;
end

case 'GAS ANALYSIS'
    if strcmpi(UNITS, 'SI') == 1
        if strcmpi(GCMSJTYPE,'WAVELENGTH') == 1
            POWER_PROPS(k,1) = [ 0 ]; % Voltage Only no Current Data
            k = k + 1;
        end
        if strcmpi(GCMSJTYPE,'MASS-CHARGE') == 1
            POWER_PROPS(k,1) = [ GCMS_MATRIX3{1,37} ];
            k = k + 1;
        end
    end
    if strcmpi(UNITS,'British') == 1
        if strcmpi(GCMSJTYPE,'WAVELENGTH') == 1
            POWER_PROPS(k,1) = [ 0 ]; % Voltage Only no Current Data
            k = k + 1;
        end
        if strcmpi(GCMSJTYPE,'MASS-CHARGE') == 1
            POWER_PROPS(k,1) = [ GCMS_MATRIX3{i,37} ];
            k = k + 1;
        end
    end
end

case 'INCLINATION'
    if strcmpi(UNITS,'SI') == 1
        POWER_PROPS(k,1) = [ 0 ]; % Voltage Only no Current Data
        k = k + 1;
    end
    if strcmpi(UNITS,'British') == 1
        POWER_PROPS(k,1) = [ 0 ]; % Voltage Only no Current Data
        k = k + 1;
    end
end

case 'NEPHELOMETRY'
    if strcmpi(UNITS,'SI') == 1
        POWER_PROPS(k,1) = [ N_MATRIX3{1,7} ]; % Watts
        k = k + 1;
    end
    if strcmpi(UNITS,'British') == 1
        POWER_PROPS(k,1) = [ N_MATRIX3{1,7} ]; % BTUs/hr
        k = k + 1;
    end
end

case 'OPTICS'
    if strcmpi(UNITS,'SI') == 1
        POWER_PROPS(k,1) = [ IMG_MATRIXS{l,30}/mW2W ]; % mW to W
        k = k + 1;
    end
    if strcmpi(UNITS,'British') == 1
        POWER_PROPS(k,1) = [ IMG_MATRIX5{i,30} ]; %
        k = k + 1;
    end
end
case 'RADIATION'
    if strcmpi(UNITS, 'SI') == 1
        POWER_PROPS(k,1) = [ RAD_MATRIX3{1,30} ]; % Watts
        k = k + 1;
    end
    if strcmpi(UNITS, 'British') == 1
        POWER_PROPS(k,1) = [ RAD_MATRIX3{1,30} ]; % BTUs/hr
        k = k + 1;
    end
end

case 'REFRACTION'
    if strcmpi(UNITS, 'SI') == 1
        POWER_PROPS(k,1) = [ REF_MATRIX3{1,11} ]; %
        k = k + 1;
    end
    if strcmpi(UNITS, 'British') == 1
        POWER_PROPS(k,1) = [ REF_MATRIX3{1,11} ]; %
        k = k + 1;
    end
end

case 'DATA PROCESSING'
    if strcmpi(UNITS, 'SI') == 1
        POWER_PROPS(k,1) = [ DSP MATRIX2{1,16} /mW2W ]; %
        k = k + 1;
    end
    if strcmpi(UNITS, 'British') == 1
        POWER_PROPS(k,1) = [ DSP MATRIX2{1,16} ]; %
        k = k + 1;
    end
end

otherwise
end
end

save('POWER_PROPERTIES_Data.mat');
APPENDIX E

ISSPO Program Error Codes

The following table details the errors that can occur during the execution of the ISSPO tool. The ERROR PRG is triggered when an error is encountered with a single corresponding error code. The codes detail the nature of the error and a possible remedy to the problem.

<table>
<thead>
<tr>
<th>ERROR CODE</th>
<th>SOURCE MODULE</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ISSPO</td>
<td>INPUT_FILE does not exist. Verify correct input file name.</td>
</tr>
<tr>
<td>2</td>
<td>ISSPO</td>
<td>InputDeck file does not exist. InputDeck file not successfully created.</td>
</tr>
<tr>
<td>3</td>
<td>CONSTANTS</td>
<td>CONSTANTS database file not created. Verify Constants database is being created.</td>
</tr>
<tr>
<td>4</td>
<td>PLANETARY_DATABASE</td>
<td>Planetary database file does not exist. Verify the Planetary Database program is correctly making the database file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ERROR CODE</th>
<th>SOURCE MODULE</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>ISSPO</td>
<td>Invalid UNITS variable entered into input file. Verify UNITS is ‘SI’ or ‘British’.</td>
</tr>
<tr>
<td>51</td>
<td>ISSPO</td>
<td>Invalid Print Option variable entered into input file. Verify PRINTFLG is ‘Y’ or ‘N’. Prints out individual program results to main program window.</td>
</tr>
<tr>
<td>52</td>
<td>DATA_TYPE_VERIFIER</td>
<td>Invalid type of science data entered into SENSOR_DATA Array. Verify sensor data types entered into array are valid science data options.</td>
</tr>
<tr>
<td>53</td>
<td>ISSPO</td>
<td>Undefined Mass Limit variable entered into input file. Verify MASS_LIMIT variable is set and defined in user input file.</td>
</tr>
<tr>
<td>54</td>
<td>ISSPO</td>
<td>Undefined Power Limit variable entered into input file. Verify POWER_LIMIT variable is set and defined in user input file.</td>
</tr>
<tr>
<td>55</td>
<td>ISSPO</td>
<td>Undefined Volume Limit variable entered into input file. Verify VOLUME_LIMIT variable is set and defined in user input file.</td>
</tr>
<tr>
<td>56</td>
<td>DATA_TYPE_VERIFIER</td>
<td>Sensor Option Requires additional Input parameters in SENSOR_DATA Variable in Input file. Verify proper data format for selected sensor.</td>
</tr>
<tr>
<td>Page</td>
<td>DATA_TYPE</td>
<td>VERIFIER</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>----------</td>
</tr>
<tr>
<td>58</td>
<td>DATA_TYPE</td>
<td>VERIFIER</td>
</tr>
<tr>
<td>62</td>
<td>DATA_TYPE</td>
<td>VERIFIER</td>
</tr>
<tr>
<td>64</td>
<td>DATA_TYPE</td>
<td>VERIFIER</td>
</tr>
<tr>
<td>65</td>
<td>DATA_TYPE</td>
<td>VERIFIER</td>
</tr>
<tr>
<td>VERIFIER</td>
<td>DATA_TYPE_VERIFIER</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td>'INCLINATION'. Requires Inclination Sensor Range in 2nd column position in SENSOR_DATA variable in Input file. Valid Ranges are 'LOW', 'MEDIUM', 'HIGH'. Verify proper range format for selected sensor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Option 'ACCELERATION' Requires additional Input parameters in SENSOR_DATA Variable in Input file. Verify proper data format for selected sensor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Invalid value in sensor option 'ACCELERATION'. Requires Acceleration Sensor Range in 2nd column position in SENSOR_DATA variable in Input file. Valid Ranges are 'SOFT', 'MEDIUM', 'HIGH', 'IMPACT', 'BALLISTIC'. Verify proper range format for selected sensor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Option 'ATMOSPHERE' - 'WIND VELOCITY' Requires additional Input parameters in SENSOR_DATA Variable in Input file. Verify proper data format for selected sensor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Invalid value in sensor option 'WIND VELOCITY'. Requires Sensor Configuration in next column position in SENSOR_DATA variable in Input file. Valid Types are 'ANEMOMETER', 'DOPPLER'. Verify proper system format for selected sensor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Invalid value in sensor option 'GAS ANALYSIS'. Requires Sensor Configuration in next column position in SENSOR_DATA variable in Input file. Valid Types are 'WAVELENGTH', 'MASS-CHARGE'. Verify proper system format for selected sensor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Invalid value in sensor option 'GAS ANALYSIS' - 'MASS-CHARGE' Option. Requires Sensing Mass Range in 3rd column position in SENSOR_DATA variable in Input file. Valid Ranges are 'LOW', 'MEDIUM', 'HIGH'. Verify proper system format for selected sensor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Option 'ACCELERATION' Requires additional Input parameters in</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA_TYPE_VERIFIER</td>
<td>SENSOR_DATA Variable in Input file. Sensor Power Type value required in 3rd Column. Verify proper data format for selected sensor.</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>Sensor Option ‘ATMOSPHERE’ – ‘TEMPERATURE’ Requires additional Input parameters in SENSOR_DATA Variable in Input file. Sensor Power Type value required in next Column. Verify proper data format for selected sensor.</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>Invalid value in sensor option ‘TEMPERATURE’. Requires Temperature Sensor Power Type in next column position in SENSOR_DATA variable in Input file. Valid types are ‘VOLTAGE’ or ‘RESISTANCE’. Verify proper range format for selected sensor.</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>Sensor Option ‘ACOUSTICS’ Requires additional Input parameters in SENSOR_DATA Variable in Input file. Verify proper data format for selected sensor.</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>Invalid value in sensor option ‘ACOUSTICS’. Requires Acoustic Sensor Type in 2nd column position in SENSOR_DATA variable in Input file. Valid Ranges are ‘SENSOR’, or ‘ARRAY’. Verify proper type format for selected sensor.</td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>Sensor Option ‘ATMOSPHERE’ – ‘DENSITY’ Requires additional Input parameters in SENSOR_DATA Variable in Input file. Sensor Medium Type value required in next Column. Verify proper data format for selected sensor.</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>Invalid value in sensor option ‘DENSITY’. Requires Density Sensor Medium in next column position in SENSOR_DATA variable in Input file. Valid types are ‘GAS’ or ‘LIQUID’. Verify proper operating medium for selected sensor.</td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>Sensor Option ‘DATA PROCESSING’ Requires additional Input parameters in SENSOR_DATA Variable in Input file. Verify proper data format for selected sensor.</td>
<td></td>
</tr>
</tbody>
</table>
Sensor Option ‘DATA PROCESSING’. Requires additional Input parameters in SENSOR_DATA Variable in Input file. Requires Microprocessor Cycle speed in 2\textsuperscript{nd} column position and number of connected devices in 3\textsuperscript{rd} column position. Verify proper data format for selected sensor.

Invalid value in sensor option ‘DATA PROCESSING’. Requires Microprocessor Cycle speed in 2\textsuperscript{nd} column position in SENSOR_DATA variable in Input file. Valid types are ‘LOW’, ‘MEDIUM’, or ‘HIGH’. Verify proper operating medium for selected sensor.

Invalid value in sensor option ‘DATA PROCESSING’. Required number of connected devices in SENSOR_DATA variable in Input file. Non-numeric value detected in 3\textsuperscript{rd} column position. Verify proper operating medium for selected sensor.

<table>
<thead>
<tr>
<th>ERROR CODE</th>
<th>SOURCE MODULE</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>TEMP_SENSORS</td>
<td>TEMP_SENSORS Database file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>101</td>
<td>PRESSURE_SENSORS</td>
<td>PRESSURE_SENSORS Database file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>102</td>
<td>NEPHELOMETERSENSOR</td>
<td>NEPHELOMETER_SENSORS Database file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>103</td>
<td>HUMIDITY_SENSORS</td>
<td>HUMIDITY_SENSORS Database file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>104</td>
<td>DENSITY_SENSORS</td>
<td>DENSITY_SENSORS Database file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>105</td>
<td>ACOUSTIC_SENSORS</td>
<td>ACOUSTIC_SENSORS Database file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>106</td>
<td>AC_FIELD_SENSORS</td>
<td>AC_FIELD_SENSORS Database file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>107</td>
<td>WIND_VELOCITY_SENSORS</td>
<td>WIND_VELOCITY_SENSORS Database file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>108</td>
<td>ACCELEROMETER</td>
<td>ACCELEROMETER_SENSORS Database file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>SENSORS</td>
<td>ERROR CODE</td>
<td>SOURCE MODULE</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>GCMS_SENSORS</td>
<td>109</td>
<td>GCMS_SENSORS Database file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>RADIATION_SENSORS</td>
<td>110</td>
<td>RADIATION_SENSORS Database file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>INCLINOMETER_SENSORS</td>
<td>111</td>
<td>INCLINOMETER_SENSORS Database file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>IMAGING_SENSORS</td>
<td>112</td>
<td>IMAGING_SENSORS Database file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>REFRACTION_SENSORS</td>
<td>113</td>
<td>REFRACTION_SENSORS Database file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>DIGITAL_SIGNAL_PROCESSING</td>
<td>114</td>
<td>DIGITAL_SIGNAL_PROCESSING Database file was not created successfully. Program erred off somewhere in module.</td>
</tr>
</tbody>
</table>

**SENSOR SUMMARY FILES**

<table>
<thead>
<tr>
<th>ERROR CODE</th>
<th>SOURCE MODULE</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>TEMP_SENSORS</td>
<td>TEMP_SENSORS Summary file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>201</td>
<td>PRESSURE_SENSORS</td>
<td>PRESSURE_SENSORS Summary file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>202</td>
<td>NEPHELOMETER_SENSOR</td>
<td>NEPHELOMETER_SENSORS Summary file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>203</td>
<td>HUMIDITY_SENSORS</td>
<td>HUMIDITY_SENSORS Summary file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>204</td>
<td>DENSITY_SENSORS</td>
<td>DENSITY_SENSORS Summary file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>205</td>
<td>ACOUSTIC_SENSORS</td>
<td>ACOUSTIC_SENSORS Summary file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>206</td>
<td>AC_FIELD_SENSORS</td>
<td>AC_FIELD_SENSORS Summary file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>207</td>
<td>WIND_VELOCITY_SENSORS</td>
<td>WIND_VELOCITY_SENSORS Summary file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>208</td>
<td>ACCELEROMETER_SENSORS</td>
<td>ACCELEROMETER_SENSORS Summary file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>Line</td>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>209</td>
<td>GCMS_SENSORS</td>
<td>GCMS_SENSORS Summary file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>210</td>
<td>RADIATIONSENSORS</td>
<td>RADIATIONSENSORS Summary file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>211</td>
<td>INCLINOMETER_SENSORS</td>
<td>INCLINOMETER_SENSORS Summary file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>212</td>
<td>IMAGING_SENSORS</td>
<td>IMAGING_SENSORS Summary file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>213</td>
<td>REFRACTION_SENSORS</td>
<td>REFRACTION_SENSORS Summary file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>214</td>
<td>DIGITAL_SIGNAL_PROCESSING</td>
<td>DIGITAL_SIGNAL_PROCESSING Summary file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>215</td>
<td>MASS_PROPERTIES</td>
<td>MASS_PROPERTIES SUMMARY file was not created successfully. Program erred off somewhere in module.</td>
</tr>
<tr>
<td>216</td>
<td>POWER_PROPERTIES</td>
<td>POWER_PROPERTIES SUMMARY file was not created successfully. Program erred off somewhere in module.</td>
</tr>
</tbody>
</table>